Fast Failure Localization in All-Optical Networks with Length-Constrained Monitoring Trails

Eva Hosszu*, János Tapolcai*, Lajos Rónyai†, Péter Soproni*, Péter Babarczi*, and Pin-Han Ho‡
*MTA-BME Future Internet Research Group[1], Budapest University of Technology and Economics, Hungary
† Computer and Automation Research Institute Hungarian Academy of Sciences and BME, Hungary
‡ Dept. of Electrical and Computer Engineering, University of Waterloo, Canada
Email: *{hosszu, tapolcai, soproni, babarczi}@tmit.bme.hu, ronyai@sztaki.hu†, p4ho@uwaterloo.ca‡

***Abstract*—Monitoring trails (m-trails) have been extensively studied as an alternative to the conventional link-based monitoring approach by using multi-hop supervisory lightpaths in all-optical networks. However, none of the previous studies have investigated the effect of length constraints upon the m-trail formation, which nonetheless correspond to the failure localization time. This paper addresses the above issue and formulates a new m-trail allocation problem, where the relationship between the number of m-trails versus the maximum hop count is explored. First, the paper investigates the theoretical bounds of allocating m-trails with at most k hops via an optimal group testing construction. Secondly, a novel meta-heuristic approach based on bacterial evolutionary algorithm for solving the length-constrained m-trail allocation problem is introduced. Through extensive simulations the performance gap of the proposed algorithm to the lower bound is presented on a wide diversity of topologies.**

***Index Terms*—monitoring trail; combinatorial group testing; bacterial evolutionary algorithm; length limit**

I. Introduction

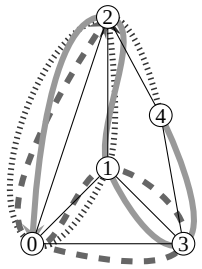
Monitoring and fault localization is a key function in the control and management of all-optical networks. In order to exclude data-plane information from the management design, a widely accepted approach is *out-of-band monitoring*, where a set of supervisory lightpaths (S-LPs) are added into the network exclusively for monitoring purpose. Each S-LP is deployed with devices that monitor the health of the corresponding lightpath, called *monitor*, which generate an alarm if any status change is detected along the S-LP. By collecting the generated alarms – which are broadcasted in the control plane via a link state protocol – each remote routing entity has to be able to unambiguously identify the failed link, called *unambiguous failure localization (UFL)*.

Since each monitor corresponds to a single supervisory lightpath that traverses a set of links, the monitor can instantly obtain the on-off status of these links. The simplest method to achieve UFL is the *link-based* approach, where each link is individually monitored. Obviously it requires $O(|E|)$ monitors, where $|E|$ is the number of links in the network. Even though it is simple, the link-based approach leaves a lot of room for improvement when considering the number of monitors, which not only correspond to additional hardware cost but also cause extra control overhead. As an enhancement, the use of multi-hop supervisory lightpaths is proposed and extensively studied in the past, such as monitoring cycles (m-cycles), m-paths and m-trails [1]–[6]. According to the binary coding mechanism b m-trails can localize any single link failure with up to $2^b - 1$ links. Thus, without topology limitations $b = \lceil \log_2(|E| + 1) \rceil$ m-trails would be sufficient for localizing any single link failure in a network with a set of links $|E|$. The previously reported random code swapping [7] (RCS) scheme for S-LP allocation in the form of m-trails (simple or non-simple paths/cycles) under single link failures can effectively approach this theoretical lower bound at the expense of deploying m-trails with excessive length. It is clear that the long S-LPs not only yield slow failure detection time, but also result in poor signal quality which may cause false detection at the monitor.

Some studies [8] tried to address the above issue and incorporate physical layer constraints in the m-trail allocation problem. However, none of these results are general enough to be efficiently used in a single link UFL m-trail allocation problem with length-constrained m-trails, investigated in this paper. Our contribution is twofold. First, from a theoretical point of view we review the related results published on combinatorial group testing [9]. In particular based on Katona's theorems [10] we introduce a novel algorithm that can intelligently compute a set of non-adaptive group tests for single failure where the maximum size is at most k. This method provides a lower bound on the number of m-trails b in terms of k the length constraint. The presented lower bound is tighter than its widely used entropy-based counterpart. Secondly, a meta-heuristic approach based on bacterial evolutionary algorithm (BEA) for length-constrained m-trail allocation problem is presented. Extensive simulations are conducted to demonstrate that the proposed BEA method approaches the theoretical lower bound in a wide range of topologies.

The rest of the paper is structured as follows. Section II defines the m-trail allocation problem and briefly reviews the related work on the design of heuristics and analytical approaches. Section III presents a lower bound on the number of m-trail for single-link UFL with k hops as the length constraints on the m-trails. Our m-trail allocation heuristic is introduced in Section IV. Simulation results are provided in

[1]This project is partially supported by HSNLab and the grants TÁMOP - 4.2.2.B-10/1–2010-0009, OTKA NK 105645, K77476, and K77778.



(a) A m-trail solution

Link	T_1	T_2	T_3
(0,1)	1	0	1
(0,2)	1	1	1
(0,3)	1	0	0
(1,2)	0	1	1
(1,3)	1	1	0
(2,4)	0	0	1
(3,4)	0	1	0

(b) Alarm code table (ACT)

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

Section V. Finally, Section VI concludes the paper.

II. Background

A. Monitoring Trails

In general, an m-trail solution consists of a set of b m-trails $T_1, T_2, \ldots, T_b$. Upon a link failure, the monitor of any m-trail traversing the failed link generates an alarm, which is sent to the network controller with the highest priority via any possible signaling protocol that supports point-to-point event-driven notification. An alarm code $[a_1, \ldots, a_b]$ can be formed at the network controller by reading the status of each m-trail, where $a_l = 1$ means that the monitor on m-trail T_l alarms and $a_l = 0$ otherwise. Fig. 1(a) shows a solution with three m-trails T_1, T_2, T_3 under single link failures. If link $(0, 1)$ fails, the monitors on T_1 and T_3 will alarm to produce alarm code $[1, 0, 1]$ at the network controller. Similarly, if link $(0, 2)$ fails, the monitors on all the three m-trails will alarm with the resulting alarm code $[1, 1, 1]$. The alarm code table (ACT) in Fig. 1(b) is available at the network controller, which maintains all the legitimate alarm codes corresponding to each failure state. Thus, the network controller can unambiguously localize a particular single link failure by matching the alarm code in the ACT.

B. Problem Definition - Deployment of M-trails for UFL with Length Limit

The input is a graph denoted as $G(E, V)$ with $|E|$ links and $|V|$ nodes, and a length limit $k \geq 1$. We are interested in the minimal number b, for which single link UFL is solvable. We need to satisfy all of the following three constraints:

- *(i) UFL constraint:* each link e must be assigned with a unique binary non-zero alarm code $c(e) = A_e = [a_1^e, a_2^e, \ldots\ldots a_b^e]$, where b is the length of alarm code, and a_l^e is a binary digit, which is 1 if the l^{th} m-trail, denoted by T_l, traverses through this link and 0 otherwise.
- *(ii) Trail shape constraint:* the links with $a_l^e = 1$ must form a trail (connected subgraph) for $l = 1, \ldots, b$. This is necessary for an m-trail to be able to traverse with a single lightpath through every link with bit 1 in the corresponding position l of alarm code.
- *(iii) Length constraint:* each m-trail can traverse at most k links.

To further explain, to satisfy the *(i)* UFL constraint, the theoretical lower bound on b is $b \geq \lceil \log_2 (|E| + 1) \rceil$. On the other hand, the *(ii)* trail shape constraint have to be considered as well, i.e. each bit position $1 \leq i \leq b$ (or column l of the ACT) have to form a connected subgraph of the underlying topology. Thus, the topology properties (e.g. nodal degree) influences the UFL monitoring lower bound. Furthermore, the lower bound on the number of m-trails is influenced by the maximal *(iii)* length constraint on the m-trails, i.e. by the maximal k hop count as well.

C. Related Work

Most of the prior art considered the *(i)* UFL and *(ii)* trail shape constraint only, and to the best of our survey none of the previously reported studies has considered the length limitation for each m-trail. The first heuristic that could achieve $\lceil \log_2 (|E| + 1) \rceil$ m-trails in typical backbone topologies was proposed in [11]. It is composed of two steps, in the first, called random code assignment (RCA), unique alarm codes of length $\lceil \log_2 (|E| + 1) \rceil$ are assigned to the links of the network in random order. In such a way UFL constraint is satisfied; however, probably it is not a valid m-trail solution, because there is a position i where the links with $a_{e,i} = 1$ do not satisfy the *(ii)* shape constraint. On the other hand, at the extent of increasing the number of m-trails, such a solution can always be transformed to a valid m-trail solution by dividing the links with $a_{e,i} = 1$ into a set of m-trails for each position i. This allows defining the cost for invalid alarm code assignments as well. In the second step, called random code swapping (RCS), the alarm codes of two links are swapped if the swapping can reduce the cost. For fast implementation, code swapping is allowed among links whose codes are the same except at one position. Swapping a code with an unassigned binary code is also possible. The running time of the heuristic is further improved with some meta heuristic approaches, such as Tabu-Search [12], and simulated annealing [13]. Recently several more heuristics have been proposed in [14]–[17].

Essentially optimal construction with $\lceil \log_2(|E| + 1) \rceil + O(1)$ m-trails for UFL of single link failure were given for several special topologies: for complete graphs [7] with $4 + \lceil \log_2(|E| + 1) \rceil$ m-trails, for a 2D grid network with two rows and arbitrary many columns [18] with $b = \lceil 0.42 + \log_2 (|E| + 1) \rceil$ m-trails.

A physical layer analysis of the degradation of signal quality for in-band and out-of band monitoring was given in [8]. The Integer Linear Program for m-trail allocation problem was extended with constraints on the physical-length of m-trails. We claim this is the first analytical work, where the physical layer constraints are considered in the optimization of monitoring trail allocation problem.

III. Theoretical Lower Bound on the Number of Length-Constrained M-trails

In this section we investigate the length-constrained m-trail allocation problem using a combinatorial group testing [9] (CGT) approach, introduced in Section III-A. First

the mathematical problem is formulated in Section III-B, then we propose a novel algorithm to obtain a lower bound in Section III-C solving the *(i)* UFL and *(iii)* length constraints simultaneously.

A. Combinatorial Group Testing Formulation

The input of the investigated non-adaptive CGT problem is a positive integer parameter k and a set of items denoted by $E = \{e_1, \dots, e_n\}$, where $n = |E|$ is the number of items in set E. A group test consists of a set of items, and the i-th group test is denoted by T_i for $i = 1, \dots, b$, where b denotes the number of tests. The cardinality of group test T_i is denoted by $|T_i|$ and is equal to the number of items in the group test. The group tests are launched parallel, without knowing the outcome of the others. Every group test may have two outcomes, positive if there is a faulty item involved and negative otherwise. We search for the minimal b such that there exists a set of group tests $T_1, \dots T_b$ with cardinality at most $|T_i| \leq k$ for $i = 1, \dots, b$, such that up to d faulty items can be unambiguously identified according to the results of the group tests.

A similar concept was introduced as *separating systems* in [10]. Let $H = \{1, 2, \dots, n\}$ be a finite set and $A_1, A_2, \dots, A_m$ subsets of H. We call a system $\mathcal{A} = \{A_1, A_2, \dots, A_m\}$ separating system, if for any two distinct elements x and y in H there exists an $A_i, 1 \leq i \leq m$ such that it contains exactly one of x and y. If we also presume that for every i the cardinality of set A_i is at most k (i.e. $|A_i| \leq k$) we have a similar problem to the length-constrained m-trail problem with the exception that in m-trail formulation we also have to account for the no-failure state. Note that if $k = n$ we have the traditional non-adaptive CGT problem.

For length-constrained m-trails, where we measure length by hop-count we have the following: b is the number of m-trails, k is the maximum length of an m-trail, and r is an auxiliary variable for computational purposes.

Let $S(n,_{=} k)$ denote the minimum number of tests needed when each test contains exactly $k \leq \frac{n}{2}$ elements. It is easy to prove that $S(n,_{\leq} k) = S(n,_{=} k)$. Note that this result can be extended for $k > \frac{n}{2}$, by replacing the tests with $|T_i| > \frac{n}{2}$ with its complementer set.

It can be shown, that the following theorem is equivalent to [10, Theorem 3]:

Theorem 1: $S(n,_{\leq} k)$ equals to the minimal integer b that satisfies the following two inequalities

$$bk \geq \sum_{i=0}^{r} i\binom{b}{i} + (r+1) \cdot \left[n - \sum_{i=0}^{r} \binom{b}{i} \right] \quad , \tag{1}$$

where r is the unique positive integer[1] such that

$$\sum_{i=0}^{r+1} \binom{b}{i} \geq n > \sum_{i=0}^{r} \binom{b}{i}. \tag{2}$$

In Section III-B the mathematical program to calculate b is presented.

[1]The index r is used in a slightly different way than in [10, Theorem 3].

TABLE I: Modified Pascal's triangle.

$b = 0$:				1				
$b = 1$:			1		2			
$b = 2$:		1		3		4		
$b = 3$:	1		4		7		8	

B. Mathematical Program Formulation

To obtain b a mathematical problem should be solved. In this subsection we investigate how to solve the mathematical problem fast. Unfortunately there is no useful closed form for the sum of the first r binomial coefficients for fixed b. Thus let us introduce the following notation

$$\left\langle {b \atop r} \right\rangle = \sum_{i=0}^{r} \binom{b}{i},$$

where b and r are positive integers such that $0 \leq r \leq b$. Using $\left\langle {b \atop r} \right\rangle$, Constraint (2) can be written as

$$\left\langle {b \atop r+1} \right\rangle \geq n > \left\langle {b \atop r} \right\rangle.$$

First let us prove the following property of the function $\left\langle {b \atop r} \right\rangle$, which will be used throughout the algorithm.

Lemma 1: For $b \geq r \geq 1$ we have

$$\left\langle {b-1 \atop r-1} \right\rangle + \left\langle {b-1 \atop r} \right\rangle = \left\langle {b \atop r} \right\rangle. \tag{3}$$

The proof is shown in Appendix A. As a corollary, it results in a similar construction as Pascal's triangle by replacing the 1s along the right-hand side of Pascal's triangle by the powers of 2, and use the standard recursive sum rule to generate the rest of the triangle, we get a triangle with $\left\langle {b \atop r} \right\rangle$ as shown on Table I.

Let us define $0 \leq r_0 < 1$ so that n is the weighted sum of $\left\langle {b \atop r+1} \right\rangle$ and $\left\langle {b \atop r} \right\rangle$, formally

$$n = (1 - r_0)\left\langle {b \atop r} \right\rangle + r_0 \left\langle {b \atop r+1} \right\rangle. \tag{4}$$

Next we show that Eq. (1) can be rewritten as

$$k \geq (1 - r_0)\left\langle {b-1 \atop r-1} \right\rangle + r_0 \left\langle {b-1 \atop r} \right\rangle. \tag{5}$$

First we have $i\binom{b}{i} = b\binom{b-1}{i-1}$ for $b \geq i \geq 1$ thus the first sum in Eq. (1) can be written as

$$\sum_{i=0}^{r} i\binom{b}{i} = b\sum_{i=1}^{r} \binom{b-1}{i-1} = b\sum_{i=0}^{r-1} \binom{b-1}{i} = b \cdot \left\langle {b-1 \atop r-1} \right\rangle.$$

The r_0 weighted sum of $\left\langle {b-1 \atop r-1} \right\rangle$ and $\left\langle {b-1 \atop r} \right\rangle$ equals to

$$\begin{aligned}(1 - r_0)\left\langle {b-1 \atop r-1} \right\rangle + r_0 \left\langle {b-1 \atop r} \right\rangle = \\ \left\langle {b-1 \atop r-1} \right\rangle + r_0 \left(\left\langle {b-1 \atop r} \right\rangle - \left\langle {b-1 \atop r-1} \right\rangle \right) = \\ \left\langle {b-1 \atop r-1} \right\rangle + r_0 \binom{b-1}{r}.\end{aligned}$$

Thus, to prove Eq. (5) we need to show

$$(r+1)\cdot\left[n-\sum_{i=0}^{r}\binom{b}{i}\right]=b\cdot r_0\cdot\binom{b-1}{r},$$

which is equivalent to

$$n-\sum_{i=0}^{r}\binom{b}{i}=r_0\cdot\binom{b}{r+1},\text{ or }n=\left\langle{b\atop r}\right\rangle+r_0\cdot\binom{b}{r+1}.$$

The latter is the definiton in Equation (4). Therefore with function $\left\langle{b\atop r}\right\rangle$ the mathematical program in Theorem 1 can be written as

$$\begin{aligned}&\min b \qquad (6)\\ k &\geq (1-r_0)\left\langle{b-1\atop r-1}\right\rangle+r_0\left\langle{b-1\atop r}\right\rangle\\ n &= (1-r_0)\left\langle{b\atop r}\right\rangle+r_0\left\langle{b\atop r+1}\right\rangle\\ &\quad 0\leq r_0<1, b\in N^+, r\in N^+, 0\leq r<b.\end{aligned}$$

C. Algorithm to Obtain the Minimal Test Number

In this section we show how to solve the mathematical program without the knowledge of the modified Pascal's triangle, which would introduce huge computational overhead. Although Katona gave the following lower bound on b in [10]:

$$b_k\geq\frac{\log_2(n)}{\frac{k}{n}\log_2(\frac{n}{k})+\frac{n-k}{n}\log_2\frac{n}{n-k}},\qquad(7)$$

it requires the the evaluation of the modified Pascal's triangle in Table I, which can be done dynamically; however, it takes a lot of operations. In the Katona algotithm $b:=b_k+1$ is taken and the column r is found in the triangle where $\left\langle{b\atop r}\right\rangle\geq n$. The obtained (b,r) integer pairs are tested, and this procedure is repeated with $b:=b+1$ until the test is positive.

On the other hand, we present a novel method in Algoritm 1 for computing b, for which we don't need ot evaluate the modified Pascal's triangle. Furthermore, our algorithm improves the Katona lower bound in Equation (7), and makes our method applicable as a benchmark on the minimal number of S-LPs in the m-trail allocation problem.

The algorithm starts with initializing b with Eq. (7), and the auxiliary variables x_1 and y_1 with 1 and 0 in Step (1)-(2), respectively. In the loop of Step (3)-(6) in the $\lceil b\rceil$th row we find an element, such that $\left\langle{b\atop r}\right\rangle<k\leq\left\langle{b\atop r+1}\right\rangle$. In Step (7)-(13) we compute b and r such the mathematical program formulated in Eq. (6) is satisfied by using the auxiliary variables x_0,x_1,y_0 and y_1 and the properties of the function $\left\langle{b\atop r}\right\rangle$ proved in Subsection III-B. As a result of this procedure, we obtain a proper lower bound b on the minimal number of m-trails satisfying *(i)* UFL and *(iii)* length constraints, which is used in the simulations in Section V.

Algorithm 1: Trail Shape Constraint Unaware Lower Bound (TSCU Bound)

Input: the number of items n; maximum test size k
Result: b
begin
$b:=\left\lceil\frac{\log_2 n}{\frac{k}{n}\log_2\frac{n}{k}+\frac{n-k}{n}\log_2\frac{n}{n-k}}\right\rceil$ // by [10]
1 $x_1:=1$, $y_1:=0$
for $r=1$ **to** $\lceil b\rceil$ **do**
3 $x_0=x_1$; // variable for $\binom{b}{r}$
4 $x_1:=\frac{b+1}{b-r+1}\cdot x_0$; // variable for $\binom{b}{r+1}$
5 $y_0=y_1$; // variable for $\left\langle{b\atop r}\right\rangle$
6 $y_1:=y_0+x_1$; // variable for $\left\langle{b\atop r+1}\right\rangle$
if $y_1\geq k$ **then** break
repeat
7 $b:=b+1$
8 $y_1=y_0+y_1$; // $\left\langle{b+1\atop r+1}\right\rangle:=\left\langle{b\atop r}\right\rangle+\left\langle{b\atop r+1}\right\rangle$
9 $x_1=x_0+x_1$; // $\binom{b+1}{r+1}:=\binom{b}{r}+\binom{b}{r+1}$
10 $y_0=2y_0-x_0$; // $\left\langle{b+1\atop r}\right\rangle:=2\left\langle{b\atop r}\right\rangle-\binom{b}{r}$
11 $x_0=x_0(1+\frac{r}{b-r+1})$;
// $\binom{b+1}{r}:=\binom{b}{r}\left(1+\frac{r}{b-r+1}\right)$
12 **if** $x_0\geq n$ **then**
$\binom{b+1}{r+2}:=\binom{b}{r+1}\left(1+\frac{b-r-1}{r+2}\right)$
$\left\langle{b+1\atop r+2}\right\rangle:=2\left\langle{b\atop r+1}\right\rangle+\binom{b}{r+2}$
$r:=r+1$
$r_0:=\frac{n-\left\langle{b+1\atop \lfloor r\rfloor}\right\rangle}{\left\langle{b+1\atop r+1}\right\rangle-\left\langle{b\atop r}\right\rangle}$ 13
until $k\geq(1-r_0)\left\langle{b\atop r-1}\right\rangle+r_0\left\langle{b\atop r}\right\rangle$
return $b-1$
end

IV. Heuristic Approach for Length-Constrained M-trail Allocation

In this section we first provide an approach which can convert any m-trail solution with arbitrary m-trail lengths into a solution with m-trails of length at most k to satisfy *(iii)* length constraint in Section IV-A. In order to obtain a valid length-constrained single-link UFL m-trail solution, we propose a novel bacterial evolutionary algorithm solving the *(i)* UFL and trail shape constraint *(ii)* simultaneously in Section IV-B using the length-constrained m-trails obtained with the algorithm in Section IV-A to satisfy *(iii)*.

A. M-trail Slicing Algorithm

In a nutshell, the proposed method uses a greedy approach that can slice the long m-trails in a graceful manner as a result of a constructive proof of Theorem 2. For this, we first give the following definition:

Definition 1: Let T_i denote an m-trail. We say that $T_i^j\subseteq T_i$ is an *m-trail chunk*, if T_i^j is a connected part of T_i.

In [19] it was shown that if a T_i m-trail distinguishes between two link failures f_1 and f_2, than after slicing there is at least one T_i^j m-trail which distinguishes f_1 from f_2. Thus, if we have an m-trail solution for UFL, slicing the m-trails to k-length chunks and adding the T_i^j chunks to the link code matrix instead of T_i as independent m-trails, the new ACT remains an UFL solution.

Lemma 2: In a finite, undirected, connected graph $G=(V,E)$ with all δ nodal degrees at least 2 ($\forall v\in V:\delta(v)\geq 2$)

there exists an $e \in E$ edge which can be removed, while $G = (V, E)$ remains connected.

Proof: As there are no $\delta(v) = 1$ nodes, $G = (V, E)$ is not a tree. Thus, it contains at least one cycle C. An arbitrary $e \in C$ edge can be removed, while $G = (V, E)$ remains connected. ■

Theorem 2: Given an arbitrary $T_i = (V_i, E_i)$ m-trail, it is possible to slice T_i in at least $\lceil \frac{k}{2} \rceil$ long chunks (except for the last chunk).

Proof: Our proof is constructive, and presents our slicing algorithm for m-trail $T_i = (V_i, E_i)$, which is, remember, a connected subgraph of $G = (V, E)$. In the algorithm, at each node v an auxiliary array $A[v]$ is maintained, which in each step contains a single connected m-trail chunk, and it is initialized to $\forall v \in V : A[v] := \emptyset$.

If $\exists e = (u, v) \in E_i : \delta(u) = 1, \delta(v) \geq 1$, then remove $E_i := E_i \setminus \{e\}$, $V_i := V_i \setminus \{u\}$ and set

$$A[v] := A[v] \cup A[u] \cup e. \tag{8}$$

Note that $T_i = (V_i, E_i)$ remains connected after the edge and node deletion.

If $\forall v \in V_i : \delta(v) \geq 2$, then $T_i = (V_i, E_i)$ satisfies the conditions of Lemma 2, thus, there exists e which can be removed, while $T_i = (V_i, E_i \setminus \{e\})$ remains connected. Select such an $e = (u, v)$ connected to a node v with minimal nodal degree, let $E_i := E_i \setminus \{e\}$ and set

$$A[v] := A[v] \cup e. \tag{9}$$

The algorithm terminates when $E_i = \emptyset$.

The m-trail chunks are stored in $A[v]$, and a new edge is added to a chunk only in Equation (8) and Equation (9). If $|A[v]| = k$ after the new edge e is added in Equation (9), then $|A[v]|$ is a connected k-length m-trail chunk, thus, it is added to the solution set and $A[v] := \emptyset$. Unfortunately, in Equation (8) it could happen that without loss of generality $|A[v]| = \lceil \frac{k}{2} \rceil$ and $|A[u]| = \lfloor \frac{k}{2} \rfloor$ before the addition. After we remove $e = (u, v)$, the size of $|A[v]| = \lceil \frac{k}{2} \rceil + \lfloor \frac{k}{2} \rfloor + 1 = k+1$. In this case, this new m-trail chunk would not satisfy the k length constraint. Thus, we have to add a new chunks to the solution, namely the longer one of $A[v]$ and $A[u] \cup e$, and set $A[v]$ to be the shorter one.

Finally, the last m-trail chunk is added to the solution, if $E_i = \emptyset$ after the iteration. ■

The presented algorithm in Theorem 2 will be used in our heuristic approach to satisfy the *(iii)* length constraint in an UFL m-trail solution satisfying *(i)* and *(ii)*.

B. Bacterial Evolutionary Algorithm for M-trail Allocation

There exist various optimization algorithms inspired by natural selection processes which are able to solve and to quasi-optimize problems having non-linear, high-dimensional, multi-modal and discontinuous characteristics. The original Genetic Algorithm (GA) was developed by Holland [20] and based on the process of evolution of biological organisms. A slightly different evolutionary technique is called Bacterial Evolutionary Algorithm (*BEA*), which was inspired by the microbial evolution phenomenon [21]. We found that the m-trail allocation problem can be efficiently formulated as a bacteria, thus, we propose a bacterial meta-heuristic approach for the length-constrained m-trail allocation problem.

The *BEA* works on a given population of bacteria. Every bacterium represents a solution for the problem, i.e. a possible m-trail allocation which satisfies *(i)-(iii)*. The used notions in the algorithm are the follows:

- *The size of the population (number of bacteria):* candidate length-constrained UFL m-trail solutions.
- *Each bacterium has a genotype (knowledge of the bacteria):* a $\underline{\underline{B}} \in \{0,1\}^{|E|\times|E|}$ quadratic binary matrix and a $\underline{\sigma_i} = (\sigma_1, \sigma_2, \ldots, \sigma_{|E|})$ permutation of the rows.
- *Each bacterium has a phenotype (specific solution):* alarm code table, ACT.
- *Fitness function (measures the quality of the solution in the phenotype):* The fitness of an m-trail solution (*(ii)* is always satisfied) is evaluated based on the following four aspects (in a descending order of importance): (1) alarm codes in the alarm code matrix are unique (i.e. *(i)* UFL is satisfied); (2) the m-trails are shorter than a given parameter k (*(iii)* is satisfied); (3) the m-trail number is low; (4) the total number of hops covered by m-trails of the solution is low.

Obviously, a solution (alarm code matrix or equivalently the m-trails) with UFL has a higher fitness value than a solution with code ambiguity. To get the ACT (phenotype) from the genotype, we take the columns of $\underline{\underline{B}}$ one-by-one in the order of $\underline{\sigma_i}$ until the UFL property is ensured in the ACT, or there are no more columns in $\underline{\underline{B}}$. In the ith iteration, we identify the connected subgraphs (m-trails) on the links containing 1 in the σ_i position, and investigate m-trail T_{i_k}. If T_{i_k} makes some link failures unambiguous, it is added to the ACT, else it is discarded. Apply the m-trail slicing algorithm in Theorem 2 if necessary to the added T_{i_k} to get the length-constrained $T_{i_k}^j$ m-trails. In this way, we have a phenotype (ACT) which satisfies all required properties *(i)-(iii)* to have a length-constrained UFL m-trail solution.

There are two operators in the *BEA* (show in Figure 2):

- *The bacterial mutation (optimizes the bacteria in the population individually):* During this operation one part of the alarm code table is selected and a given number of clones are generated. A clone differs from the original bacterium only in the currently examined part of the gene

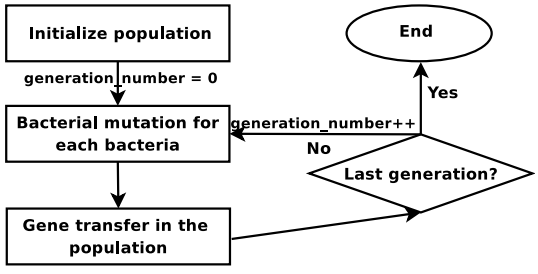


Fig. 2: State-chart of BEA

sequence (ACT). As a result of mutation the clone (or the original bacteria) with the best fitness value is selected.

- *Gene transfer (transfers information between different bacteria within the population):* The bacteria are ordered based on their fitness value. The goal of the gene transfer is to transfer knowledge from a random bacteria from top half of the list (with a good fitness) to a random bacterium from the worst half. If their fitness values are not equal, then the better one gives a part of its gene sequence ($\underline{\underline{B}}$ and $\underline{\sigma}$) to the worse one. The length of the transferred gene sequence is an input parameter for the algorithm as well as the number of gene transfers during a bacterial cycle.

In order to ensure that a valid UFL solution is available at the first generation, we have at least three bacteria. First, an $\underline{\underline{I}}$ identity matrix is the genotype of the first bacteria (this corresponds to the case when each link is monitored individually). The second is $\underline{\underline{1}}-\underline{\underline{I}}$ matrix, where $\underline{\underline{1}}$ is an all-one matrix. The third bacteria has RCS as phenotype extended with random element to a quadratic matrix, and the permutation is set to the RCS part of the matrix. The genotype of the other bacteria are all random 0-1 matrices with random permutation of the rows.

V. Simulation results

Simulations on several reference network topologies taken from [22] and on randomly generated planar 2-connected backbone network topologies via LEMON [23] were conducted. The performance metrics of interest is the number of m-trails for UFL with length limit (satisfying *(i)-(iii)*) compared to the TSCU Bound in Algorithm 1 (satisfying only *(i)* and *(iii)*). Note that in the TSCU algorithm the *(ii)* trail shape constraint is ignored, thus, our main investigation is the effect of the trail shape constraint respect the network topology. The values of the length constraint are running from 1 to $\lfloor \frac{|E|}{2} \rfloor$, as the Katona theorem used in the TCSU algorithm only gives the lower bound in the range.

In the simulations the number of bacteria in the BEA algorithm is set to 10, i.e. we have $\underline{\underline{I}}$, $\underline{\underline{1}}-\underline{\underline{I}}$, a matrix initialized with the result of the RCS algorithm [11] and 7 random matrices. In order to find a compromise between the solution quality and running time, we found that that the number of m-trails only slightly decreases above 10 generations with 10 clone and mutation number, with 10 gene transfer and with 0.2 transferred gene sequence. Thus, we used these settings in the simulations.

A. Effect of Topology Density

Fig. 3 shows the number of m-trails with different length constraint k on randomly generated networks with different density. All the random networks have 55 edges and different number of nodes. The average nodal degree of each random graph is written in the legend of the chart. All the networks are planar and two connected, except the full mesh.

One can observe that higher average network density results in less length-constrained m-trails, which meets our expectations. Note, that the BEA approach, which considers the *(ii)* trail shape constraint in the m-trail design, requires only a slightly more m-trails in the full mesh network as the TCSU lower bound, which completely ignores *(ii)*. Furthermore, it is interesting to observe that even in networks around 3 average nodal degree the BEA algorithm provides a solution close to the theoretical lower bound, which does not consider the topology limitations. The effect of the topology on the number of m-trails can be observed with the increased m-trail number in sparse networks with lower nodal degree than 2.5.

B. Performance of Real Network Topologies

Fig. 4 shows the number of m-trail with different length constraint k on real network topologies. For small and large k values the proposed BEA algorithm approaches the theoretical lower bound in all networks, and requires only $0-3$ more m-trails. However, for k values around the network diameter, the performance gap is larger. Our observation is that the typical m-trail length falls within this range. Thus, with the greedy m-trail slicing algorithm in Section IV-A these m-trails are sliced into two, in a k length one, and a second short one. However, the number of m-trails is larger only with $7-8$ than the lower bound. Considering that even in full mesh networks the lower bound can not be reached owing to the topology limitation on the shape of the trails, we can conclude that even this simple greedy slicing performs efficiently in real networks.

VI. Conclusions

In this paper we investigated the monitoring trail (m-trail) allocation problem in all-optical WDM networks for single-link failure localization with limits on the maximum length of m-trails. We provided lower bounds on the number of m-trails depending on the length limit using the results of combinatorial group testing. We provided a yet efficient meta-heuristic approach based on bacterial evolutionary algorithm that can efficiently solve the length-constrained monitoring

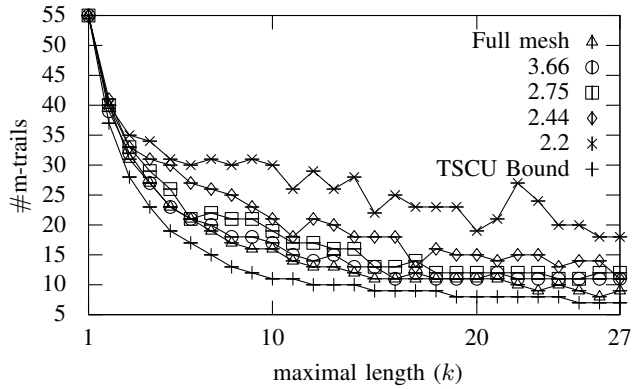
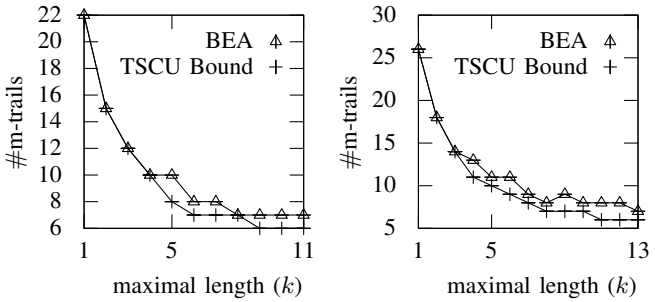
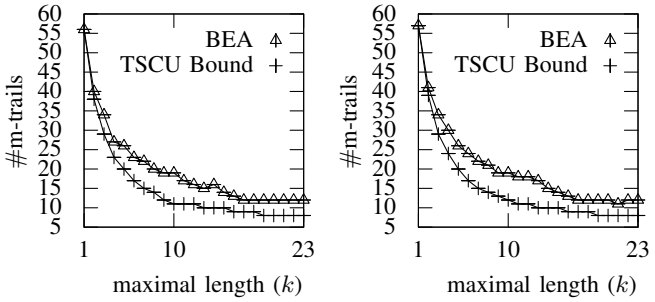


Fig. 3: The performance of the proposed BEA algorithm in comparison with the TSCU Bound on random network topologies with 55 edges.



(a) European core topology (16 node, 22 edge)

(b) German (17 node, 26 edge)



(c) Italian (33 node, 56 edge)

(d) European base topology (37 node, 57 edge)

Fig. 4: Number of m-trails for UFL in real network topologies.

trail allocation problem. Through simulations we showed that the performance of the proposed heuristic algorithm is near optimal in a wide variety of topology density and size.

APPENDIX

A. Proof of Lemma 1

Proof: In the proof we use the following well known formula for binomial coefficients

$$\binom{b-1}{i} + \binom{b-1}{i+1} = \binom{b}{i+1} \tag{10}$$

Equation (3) holds because

$$\begin{aligned}
\left\langle {b-1 \atop r-1} \right\rangle + \left\langle {b-1 \atop r} \right\rangle &= \sum_{i=0}^{r-1} \binom{b-1}{i} + \sum_{i=0}^{r} \binom{b-1}{i} = \\
&\sum_{i=0}^{r-1} \binom{b-1}{i} + 1 + \sum_{i=1}^{r} \binom{b-1}{i} = \\
&\sum_{i=0}^{r} \binom{b-1}{i} + 1 + \sum_{i=0}^{r-1} \binom{b-1}{i+1} = \\
&1 + \sum_{i=0}^{r-1} \left[\binom{b-1}{i} + \binom{b-1}{i+1} \right] = \\
&1 + \sum_{i=0}^{r-1} \binom{b}{i+1} = \sum_{i=0}^{r} \binom{b}{i} = \left\langle {b \atop r} \right\rangle
\end{aligned} \tag{11}$$

■

REFERENCES

[1] C. Assi, Y. Ye, A. Shami, S. Dixit, and M. Ali, "A hybrid distributed fault-management protocol for combating single-fiber failures in mesh based DWDM optical networks," in *Proc. IEEE GLOBECOM*, 2002, pp. 2676–2680.

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

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

[4] C. Mas, I. Tomkos, and O. Tonguz, "Failure location algorithm for transparent optical networks," *IEEE J. Select. Areas Commun.*, vol. 23, no. 8, pp. 1508–1519, 2005.

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

[6] Y. Wen, V. Chan, and L. Zheng, "Efficient fault-diagnosis algorithms for all-optical WDM networks with probabilistic link failures," *J. Lightwave Technol.*, vol. 23, pp. 3358–3371, 2005.

[7] J. Tapolcai, P.-H. Ho, B. Wu, and L. Rónyai, "A novel approach for failure localization in all-optical mesh networks," *IEEE/ACM Trans. Networking*, 2011.

[8] E. Moghaddam, J. Tapolcai, D. Mazroa, and . Hosszu, "Physical impairment of monitoring trails in all optical transparent networks," in *Int. Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, 2011.

[9] D. Du and F. K. Hwang, *Combinatorial Group Testing and Its Applications*. World Scientific, 2000.

[10] G. Katona, "On separating systems of a finite set," *Journal of Combinatorial Theory*, vol. 1, no. 2, pp. 174–194, 1966.

[11] 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.

[12] A. Haddad, E. Doumith, and M. Gagnaire, "A meta-heuristic approach for monitoring trail assignment in WDM optical networks," in *Int. Workshop on Reliable Networks Design and Modeling (RNDM)*. IEEE, 2010, pp. 601–607.

[13] E. A. Doumith, S. A. Zahr, and M. Gagnaire, "A novel meta-heuristic approach for optical monitoring-tree design in wdm networks," in *16th International Conference on Optical Networking Design and Modeling, ONDM 2012*, 2010, pp. 1–6.

[14] K. Maamoun and H. Mouftah, "Using monitoring trails (m-trails) with established lightpaths to perform fault localization in all-optical networks," in *Computer Engineering Conference (ICENCO), 2010 International*. IEEE, 2010, pp. 68–71.

[15] ——, "Novel techniques for deploying monitoring trails (m-trails) for fault localization in all-optical networks," in *High-Capacity Optical Networks and Enabling Technologies (HONET), 2010.* IEEE, 2010, pp. 26–32.

[16] Y. Zhao, S. Xu, X. Wang, and S. Wang, "A new heuristic for monitoring trail allocation in all-optical wdm networks," in *Proc. IEEE GLOBECOM*, dec. 2010, pp. 1 –5.

[17] N. Ogino and H. Nakamura, "All-optical monitoring path computation based on lower bounds of required number of paths," in *Communications (ICC), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–6.

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

[19] P. Babarczi, J. Tapolcai, and P.-H. Ho, "Srlg failure localization with monitoring trails in all-optical mesh networks," in *Proc. International Workshop on Design Of Reliable Communication Networks (DRCN)*, Krakow, Poland, 2011, pp. 188–195.

[20] J. Holland, *Adaptation in natural and artificial systems*. MIT press Cambridge, MA, 1992.

[21] N. Nawa and T. Furuhashi, "Fuzzy system parameters discovery by bacterial evolutionary algorithm," *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 5, pp. 608–616, 1999.

[22] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0 – survivable network design library," *Networks*, vol. 55, no. 3, pp. 276–286, 2010.

[23] "LEMON: A C++ Library for Efficient Modeling and Optimization in Networks," http://lemon.cs.elte.hu.