Abstract—Link failure localization has been an important and challenging problem for all-optical networks. The most general monitoring structure, called $m$-trail, is a light-path into which optical signals are launched and monitored. How to minimize the number of required $m$-trails is critical to the expense of this technique. Existing solutions are limited to localizing single link failure or handling only small networks. Moreover, some practical constraints, like lacking of knowledge of the failure quantity, are ignored. To overcome these limitations is prospective but quite challenging. To this end, we provide novel theoretical solution frameworks toward the multi-failure localization problem. On one hand, for small dense networks, we provide a tree-decomposition based algorithm; on the other hand, a random walk based localized algorithm for large scale sparse networks is proposed. In addition, we further adapt these two algorithms to cope with three practical constraints. Theoretical analysis and simulation results are included to prove the correctness and efficiency of the proposed schemes.

I. INTRODUCTION AND RELATED WORK

How to survive from multiple link failures has been a long studied problem for optical networks [1][2], where schemes like redundant capacity, capacity reprovision and p-cycle reconfigurations have been proposed. Beyond mitigating from the damages caused by the link failures, the problem of efficiently localize them in all optical networks, where no electronic switches exists to help discover individual link failures [3], has also seized an increasing attentions[4][5].

However, the researches in this field have been limited to single failure diagnosis. The idea proposed by Zeng et al. [6] of launching probe signals through a set of selected lightpaths and nailing down the faulty links through combinatorial analysis over the probing results, has been further developed by [7][8][9] for single failure localizations. But none of the exiting solutions efficiently tackle the multi-link problem. It is not hard to see that from single-link to multi-link, the localization problem becomes much harder, especially when the exact number of failures is unknown. Therefore, the existing solutions for single link failure localization cannot be simply tuned to address the multiple link problem. What is more, most of the existing single-link solutions involve high computation overhead and thus cannot meet the scalability requirement of nowadays networks.

In this paper, we provide two solutions for localizing multiple link failures, which optimize both the cost of optical monitors and wavelength. Besides, some practical constraints ignored by the existing single failure localization schemes are well investigated. The first and the most important one is that the number of failed links cannot be known beforehand, or tightly upper-bounded. We formulate the problem of estimating this failure quantity as a graph optimization problem, avg-$\beta$-disruptor, which is within the scope of network topology vulnerability assessment [10]. Based on this estimated value, our proposed schemes go through. The second concern is the limited capacity of the WDM (wavelength-division multiplexing) technique as summarized in [11]. Dense-WDM (DWDM) provides up to 128 channels in a single fiber, which requires more power, higher accurate lasers and wave filters, as well as more expensive EDFA’s for amplifiers than Coarse-WDM (CWDM) which has only 18 channels. So taking the set up expense into account, some networks may adopt CWDM and the capacity of each link is limited. Third, some links may cause transmission failures only to a proportion of wavelengths [8], and some performance metrics, like SNR may not always correctly reflect the results of the probes.

To our best knowledge, this is the first attempt to minimize the number of m-trails for multiple link failure localization in various network scenarios. Our contributions are 3-fold: (1) For small networks with an arbitrary density and multiple monitoring locations, we provide a tree-decomposition based method, which is able to catch 90% of the faulty links, and indicates the potential set of links that contain the leftover faulty ones. (2) For large scale networks whose monitors can only be equipped at the transmission terminals (source and destination), we provide a random walk based algorithm, which catches 95% faulty links. (3) For both solutions proposed, we provide further adaption for meet a specific set of constraints and requirements, which frequently occur in real network scenarios.

The rest of this paper is organized as follows: Section II overviews the concept of the $m$-trail solution and the formal definitions of the localization problem. Section III presents the tree decomposition based solution and Section IV shows the random walk based solution. In Section V, simulation results are exhibited. Adaption to practical constraints are included in Section VI and Section VII summarizes this whole paper.

II. BACKGROUND AND PROBLEM BRANCHES

A. M-trail Solution for Localizing Single Failure

A $m$-trail is a lightpath which consists of multiple links. The two ends of the $m$-trails are called transmitter and receiver
respectively. Each \( m \)-trail will be equipped with a monitor at the receiver end, which monitors the received wavelength signal after it is injected at the transmitter and traverses through the path, to catch any abnormal signs like high SNR or BER (which we call positive) to indicate if any of the links within the path are failed. Note that each link can be included by multiple \( m \)-trails, and a failed link will poison all the trails containing it, therefore by carefully designing the \( m \)-trails and observing the monitored result of each one, it is possible to catch the failure links via combinatorial methods. An example of this solution can be illustrated by an example in Fig. 1.

As summarized by Doumith et al. in [12], this solution has several variants known as Monitoring cycles, Monitoring trails and Monitoring trees, and their differences lie in the positions of the laser diodes/optical monitors deployed. Although dedicated supervisory channels are required to carry the light-weight probe signals, since the faulty detection procedure is periodically executed or triggered by network traffic exceptions, the network capacity will not be significantly affected by these probe messages. The optimization toward our methodology in this paper, includes both the enhancement of the diagnosis accuracy, and the reduction of the diagnosis cost (the number of laser diodes/optical monitors and supervisory channels).

Given that any \( m \)-trail containing at least one link failure will return a positive monitoring result, and those consisting of operative links will return negative results, the problem of localizing multi-link failures is thus converted to a trail-selection problem, which is abstracted as an optimization problem \( M\text{-LFL} \) (Multi-Link failure localization):

Definition 1 (M-LFL): Given an undirected connected graph \( G = (V, E) \), where \( V \) refers to the vertex set and \( E \) edge set. Assume \( d > 1 \) edges out of \( E \) are failed, unambiguously localize them with the minimized number of \( m \)-trails.

We study basically two branches of this problem, i.e., Multi-LOC and \( st\)-LOC, which respectively refer to the case with unlimited number of monitoring locations and only a node pair \((s, t)\) as the transmitter and receiver, i.e., a single monitor at \( t \). These two branches in fact match different application scenarios: for networks with relative smaller size, it is practical more convenient to place monitors at arbitrary nodes, which will be addressed by our tree-decomposition algorithm. On the other hand, for large scale networks with limited locations available for deploying monitors, we present a random walk based algorithm. Beyond this, we will consider three practical constraints, i.e., upper-bounded fiber link capacity, unknown link failure quantity and partial link failures, in Section VI.

Some notations used throughout the paper are summarized as follows: \( V, E \) - vertex/edge set; \( n, m \) - vertex/edge set size; \( D \) - graph diameter; \( \deg(v) \) - degree of the vertex \( v \); \( d \) - number of failed links.

III. A TREE DECOMPOSITION BASED ALGORITHM FOR MULTI-LOC IN SMALL DENSE NETWORKS

In this branch, we can place a monitor at any node within the graph and want to minimize the number of \( m \)-trails used. The flow of this section is as follows. Firstly, we introduce the subroutine (\( HA \)) proposed by Harvey [7] on localizing single failures on trees. Secondly, we state our algorithm TDL (Alg. 1) to localize multiple failures on general graphs.

1) propose Thm. 1 on a probabilistic size upper-bound of graphs containing at most one failure.
2) propose Alg. 2 to find the \( v \)-Span graph (Def. 2) and \( L \)-degree (Def. 3) of each vertex \( v \).
3) propose Alg. 3 to greedily partition the graph into edge-disjoint tree-subgraphs based on the graph size upper-bound and vertex \( L \)-degree derived above.
4) employ Alg. \( HA \) to localize the single link failures within each tree-subgraph.

Finally, we sketch the process of finding redundant tree-subgraphs to improve the diagnosis accuracy.

A. Algorithm \( HA \) on tree graphs

For each edge \( e \), \( HA \) defines two metrics: depth refers to the number of nodes in the path from \( e \) to the root; light-depth refers to the number of light edges on that path. By defining the weight of node as its number of descendants (nodes in the subtree rooted at this node), a path from a node to its heaviest child is a heavy edge, otherwise light edge. All the edges of the same depth or light-depth is included in a subtree traversed by a trail. So with all these subtrees as trails for depth \([0, D(G) - 1]\) and light-depth \([0, \log_2(|V(G)|))]\), then depth and light-depth of any single failed edge can be obtained, and further localized using at most \( \log_2(|V(G)|) \) trails. A theoretical bound over the number of trails \( L(G) \) was derived in [7]

Lemma 1: [7] For any tree \( G \), we have the minimum number of trails \( L(G) \) as

\[ L(G) \leq \min(\Omega(D + \log n), \min(O(D \log n), O(D + \log^2 n))] \]

where \( D = D(G) \) and \( n = |V(G)| \).

[7] mentions that if the graph contains \( d + 1 \) edge-disjoint spanning trees, then the problem can be solved for each individual tree. However, no hints of how to decompose the graph has been investigated. Since it is proved that only a \( 2(d + 1) \)-edge-connected graph has \( d + 1 \) edge-disjoint spanning trees, this scheme is inapplicable to general graphs.

---

![Diagram](image_url)
B. Tree-decomposition

In light of the localization algorithm on trees, we aim to decompose the graph into multiple edge-disjoint tree-subgraphs (not spanning tree), and localize the failure in each subgraph in parallel. The reason why we choose tree as the structure of the subgraphs is two-fold: on one hand, HA gives performance guarantees on trees; on the other hand, it is either difficult to decompose a general graph into other structures, like ring backbone adopted in [9], or expensive to localize the failure in the subgraph, like star structures.

Our tree decomposition based localization (TDL) is included in Alg. 1. Within each tree-subgraph, we execute HA algorithm. When there is zero or exactly one link failure in a tree-subgraph $T_i$, respectively no link or this failed one can be correctly returned by the algorithm $HA(T_i)$. Otherwise, the set of suspect links of the tree will be recorded. The random algorithm in [9] could also work as a subroutine in this step, but since there is no guarantee over the expense of their code swapping step, we prefer this deterministic one. The two leftover problems for this solution are: (1) how large should each tree-subgraph be to make each subgraph only contain a single failure with high probability? (2) how to partition the graph into tree sub-graphs to get the lowest overall expense, i.e., the minimum total number of m-trails?

**Algorithm 1** TDL (tree-decomposition based localization)

1: **Input**: Graph $G=(V, E)$ and constant $d$.
2: **Output**: The set of failed links $F$ and the set of suspect links $H$.
3: 4: **Offline stage**
5: Run Alg. 3 to divide the graph $G$ into $r \geq d$ edge-disjoint tree-subgraphs $T_1, T_2, \ldots, T_r$.
6: 7: **Online stage**
8: Let $F \leftarrow \emptyset$ be the set of failed links.
9: Run $HA(T_i)$ for all $i=1, \ldots, r$ to localize the single link failure in each $T_i$, in parallel.
10: 11: if $HA(T_i)$ returns one failed link $e$ then
12: $F \leftarrow F \cup \{e\}$;
13: else
14: if $HA(T_i)$ fails at depth $h$ then
15: Put all the links with lower depth($\leq h$) in to the suspect link set $H$.
16: end if
17: return Failed link set $F$

We assume that the link failures fall into a Bernoulli distribution $[5]$, where each edge $e$ has a probability $f_e = p$ to be failed for some constant $p \in (0, 1)$. An estimation over the size upper-bound of any subgraph which contains most one failed link, is obtained in Theorem 1.

**Theorem 1**: Given that at most $d$ out of $m$ links are failed in the whole network, any subgraph with more than $\left\lceil \frac{m(1 - d/(1 - \sqrt[1/d]{1 - m(1 - p)}) \cdot m)}{m(1 - 1/(1 - \sqrt[1/d]{1 - m(1 - p)}) \cdot m)} + 1 \right\rceil$ edges will contain at most one failed link with a high probability $\alpha$, where $\alpha \in (0, 1)$ can be arbitrarily large.

**Proof**: Let $f(s_i, d_i)$ denote the probability that there are at most $d_i$ failed links in a subgraph containing in total $s_i$ edges, then we have

$$f(s_i, d_i) = \sum_{x=1}^{d_i} \binom{s_i}{x} p^x (1 - p)^{s_i - x}$$

hence we need first to bound $p$ from above to have $f(m, d) > \alpha$. Let $p = \frac{d}{(2\exp - 1)}$ with a constant $\delta \in (0, 2\exp - 1)$, according to

**Chernoff bound**, we need to have

$$f(m, d) > \exp \left( -\frac{mp\delta^2}{4} \right) > \alpha$$

Then if we need to have $d_i \leq 1$ with probability $\alpha$, i.e.,

$$\alpha \leq (1 - p)^{-1}(1 + (m_i - 1)p)$$

The proof completes by strengthening this inequality.

With regard to the graph partition process, two core concepts $v$-Span graph and $L$-degree are proposed.

**Definition 2**: ($v$-Span graph) Given graph $G = (V, E)$ and constant $d \ll |E|$, for any vertex $v \in V$, a $v$-Span graph $\Psi(v)$ is constructed by Alg. 2.

**Algorithm 2** $v$-Span Subgraph Generation

1: **Input**: Graph $G=(V, E)$ and vertex $v \in V$.
2: **Output**: $v$-Span graph $\Psi(v)$.
3: $\Psi(v) \leftarrow \{v\}$
4: while $E(\Psi(v)) \leq \left\lceil \log \frac{\ln 1/p}{\ln 2} \right\rceil + 1$ do
5: do BFS (breadth-first-search) starting from $v$;
6: include any vertex reached by BFS into $\Psi(v)$;
7: if any edge $e$ reached by BFS introduces a cycle, eliminate the endpoint with smaller degree from $\Psi(v)$, as well as all its incident edges.
8: end while
9: return $\Psi(v)$

**Definition 3**: For any vertex $v$, the $L$-degree of $v$ is

$$L(v) = \min \left\{ \frac{|D(\Psi(v))| \cdot \log |\Psi(v)|}{|E(\Psi(v))|}, \frac{D(\Psi(v)) + \log^2 |\Psi(v)|}{|E(\Psi(v))|} \right\}$$

$L$-degree is formulated based on the upper-bound of $m$-trails in Lemma 1. The denominator $|E(\Psi(v))|$ is to avoid selecting trees with singleton edge, which results in too many tree-subgraphs and overall m-trails.

In principle, the graph is decomposed into multiple edge-disjoint $v$-Span graph greedily based on the vertex $L$-degree, as shown in Alg. 3. There are two issues to be noticed in this scheme. One is the cycle-break operation in Step 7 of Alg. 2, which is used to avoid isolated edge. An isolated edge refers to the one all of whose neighboring edges (those sharing endpoints with this edge) have been included in some tree-subgraph already in the decomposing process, in which case we will have to use another tree-subgraph to cover this singleton edge and a m-trail only consisting of it, which is costly. The other one is the greedy strategy based on $L$-degree for selecting tree-subgraphs. As shown from Step 7 to 10 in Alg. 3, in order to save the localization expense in each tree, we choose the vertex with the smallest $L$-degree to span the
tree-graph. However, this greedy selection only takes place within a set $V_s$ of vertices, instead of the whole vertex set $V(G)$. $V_s$ are those vertices which have been visited by the spanning process of previously chosen trees, but not included. In other words, they are involved with cycles and ruled out by the cycle-break operation. It is easy to see that normally vertex with small node-degree has a large $L$-degree, therefore these abandoned vertices gets less likely to be chosen for spanning a tree-subgraph as more and more edges are removed from the graph. They will become small leftover fragments if we only span the tree-subgraph from vertices with smallest $L$-degree.

C. Redundant Tree-Subgraphs

As is shown, if each of the tree-subgraph contains at most one failed link, the localization can be completed within a single iteration. Otherwise, the multiple-link failures contained by a single subgraph become false negative ones (positive items diagnosed as negative). It is natural to further diagnose these subgraphs after the first iteration of localization. However, for networks with long geographic distances between node pairs, and thus large transmission latency, the tradeoff in time complexity of multi-iteration localization is not worthwhile.

We resort to redundant tree-subgraphs to address this problem. The subgraphs selected by Alg. 3 are edge-disjoint from each other, while redundant subgraphs share edges with them. In other words, we redo Alg. 3 to generate more $\nu$-Span graphs. Since the most inefficient part of the tree-decomposition scheme lies in those isolated edges or small trees with few edges, we select a predefined number of tree-subgraphs, which are spanned from vertices with small node-degree. It is straightforward that based on the $L$-degree we used, the tree-subgraphs selected by TDL are almost rooted at the ones with large node-degree, which means the redundant tree-subgraphs have small overlapping with them.

IV. A RANDOM WALK BASED ALGORITHM FOR ST-LOC IN LARGE-SCALE TRANSPARENT NETWORKS

In large-scale optical networks, due to the transparency requirement of the fiber data-connections, monitors may only be placed at specific locations. Moreover, most of existing failure localization schemes have high computation overhead or require the underlying network topology to be specific structured, thus impractical for industral implementation and maintenance. To this end, we introduce a random walk [13] based algorithm, using only a single pair of transmitter/receiver.

Unambiguous multi-failure localization with restricted monitoring locations is intrinsically difficult. When both ends of every m-trail are fixed, even $O(|E|)$ trails cannot provide reliable results. Moreover, without the knowledge of the network topology, methods based on connected subgraph [8] or integer programming [14] cannot go through.

Our algorithm leverages a random walk based $d$-disjunct matrix construction proposed in [15]. A $d$-disjunct matrix is a binary matrix $M$ where within any combination of $d + 1$ columns, for each single column $c_0$ and the other $d$ columns $c_1, \cdots, c_d$, there is at least one row $r$ with $M[r, c_0] = 1$ and $M[r, c_i] = 0$ for all $i = 1, \cdots, d$. The concept of $d$-disjunct matrix originates from the group testing theory and has been widely applied to the anomaly localization out of various large-scale instances, due to its efficient decoding method. In the context of link failure localization, assuming the number of link failures is upper bounded by $d$, we let each column of the matrix represent a link, and each row represent a set of links (a 1 entry means the corresponding row m-trail contains the corresponding column link, 0 otherwise). If each row forms a m-trail, since any good link will exist in at least one m-trail which successfully delivers the probe signals, guaranteed by the property of aforementioned $M$, by eliminating all the links contained by negative m-trails, we can identify all the leftover links as faulty ones. Fig. 2 illustrates an example of using a 2-disjunct matrix, and the literature of $d$-disjunct researches is summarized by Du, et al. in [16].

Nevertheless, the implementation of this idea is non-trivial. The construction of $d$-disjunct matrix is quite challenging [16], and becomes much more harder when each matrix row (mapped to trails) needs to be a connected set of the columns (mapped to links). There are few work over the graph-constraint $d$-disjunct matrix until recently, in [15], the authors constructed each matrix row by a random walk on the graph and theoretically proved the probabilistic disjunctness of the resulted matrix, when the number of rows is larger than specific lower bound. We leverage this construction to our localization problem and state the algorithm as Alg. 4.

Algorithm 4 RWL (Random Walk)

1. Input: Network $G = (V, E)$, with a transmitter $s$, receiver $t$. A predefined number $R_0$.
2. Output: The list of failed links.
3. 4. //offline stage
5. Form a set of $R_0$ lightpaths by simulating random walks starting from $s$ till $t$ on $G$.
6. 7. //online stage
8. Launch $R_0$ probe signals in parallel through these lightpaths and check the received ones at $t$.
9. For all the lightpaths successfully delivering the signals, their member links are diagnosed as functional, while the other links are all failed ones.
10. Return all the failed links.

The correctness of this algorithm is ensured by the $d$-disjunct property. If we regard each of these $R_0$ m-trails as a row, they will form a $d$-disjunct matrix with a high probability. Based on the definition of $d$-disjunct matrix, each good link will be traversed by at least one m-trail, which does not go through any of the $d$ failed links. Therefore, all good links will be exempted from the failed link list.
random walks, the constructed matrix $M$ is $d$-disjunct with probability $1 - o(1)$. By roughly taking $(\frac{1}{d^2})^2$-mixing time as $O(\log n)$, $R = O(d^3 \log^4(|E|) \log(|E|/d))$.

We can furthermore improve this algorithm by a Local Rarest First strategy. The idea is to setup a counter $\rho(e)$ for each edge $e$, which stands for the number of different walks that have traversed it. It gives a bias to the random walks, i.e., instead of a random neighbor, the one with the minimum counter value will be picked to proceed the walks.

The purpose of this scheme is to span the walks over the whole networks in a minimum latency, and decrease the length of each walk. In fact, this adaptation to the algorithm RWL still keeps the upper-bound in Lemma 2, though unnecessarily tight, as shown later in the simulation section (Fig. 3(e)).

V. Simulation Results

For the purpose of experimental evaluation, we present the performance of our solutions for different network scenarios in terms of network sizes, topologies and link failure rate. No comparisons with existing single link failure location algorithms since (1) The tree decomposition localization (TDL) uses one of them as a subroutine. When there is only link failure, TDL itself will degenerate to the subroutine. As mentioned earlier, TDL can also use other single failure algorithms as its subroutine, but comparisons between TDLs using different subroutines are again repeating work of the comparisons between existing single link failure solutions. (2) The random walk localization (RWL) enforces the constraints of using only a single pair of monitors for very large networks, and we are not aware of any of the existing single link failure solutions can be fit into this problem setting.

A. The Tree Decomposition based Algorithm

TDL is designed for networks with small size, therefore we adopt one real network ARPANET with $n = 20$ vertices, $m = 32$ edges, as well as three generated networks with $n' = 100$ vertices, $m' = 1000$ edges, which follow Erdos-Renyi, Power-Law [17] and Waxman [18] models respectively. The results are included in Fig. 3(a) and 3(b).

The x-axis of both Fig. 3(a) and Fig. 3(b) is the failure density $d/m$ which increases from 1% to 9%. The y-axis of them is respectively the trail-link ratio $\alpha = \frac{T}{k}$ i.e., number of m-trails over number of links for Fig. 3(a), and the ratio of correctly localized failed links $\beta$ for Fig. 3(b). From these results we can derive the following effects of two factors: link failure rate and underlying topology.

Effect of Link Failure Rate. An increasing number of link failures brings up the difficulty of the localization, however, we conclude that the localization complexity is almost stable as $d/m$ increases from the experimental results. As shown in Fig. 3(a) and Fig. 3(b), the number of m-trails keeps below $0.6m' = 30$ with the $\beta$ greater than 90%, as the failure rate increases from 0.01 to 0.1.

B. The Random Walk based Algorithm

To investigate the performance of the localized algorithm, we test it on large-scale random networks, which are generated following the Sparse Power-law model with average node-degree 2 and 3. Besides the number of m-trails, we also check the average length of each negative m-trail (i.e. no failed links contained), which helps the receiver to decide how long it should wait for a signal to arrive, or regard it as a failure. Therefore, it reflects the online time complexity of this algorithm. We investigate the robustness of the algorithm to increasing network size in terms of the number of links $m$. For each network instance, we fix two vertices with the largest node-degree as the transmitter and receiver.

We generate networks with a range of 500 to 2500 nodes, respectively 1000 to 5000 links. The failure rate $d/m$ is set to 5% which is larger than normal cases. RWL is different from TDL in that the number of m-trails is pre-defined, so we vary this value and check the resulted localization success rate. As shown in Fig. 3(c), as the network size increases, the required number of m-trails to localize the link failures is increasing as well. This is inevitable, since the more link failures we have, the harder the localization is. However, from Fig. 3(d), we can see that the expense is less than $d^2 \log m$, which is the theoretical upper-bound [16] of the number of rows of the $d$-disjunct matrix, i.e. the number of trails to locate $d$ failures without the graph constraint. Furthermore, from Fig. 3(e), we can obtain a rough upper-bound as $cd \log m$ where the constant factor $c$ is less than 5 for a network with up to 5000 links and 250 failed ones.

Average length of each negative m-trail. On one hand, as shown in Fig. 3(f), the average negative trail length increases from around 30 to 100 as the network size increases from 1200 to 5000 links. Since we only have a pair of transmitter and receiver, their diameter can be as large as the diameter of the graph, which increases as the graph gets larger. Based on this, the transmitter can use the maximum transmission time of 100 hops as the deadline for the WDM signals.

Effect of Topology Variation. The results over networks generated from the three random network models almost the the similar trend as link failure rate increases, i.e., trail-link ratio increases from 0.2 to 0.6 and stays stable around 0.5, while success ratio keeps stable around 90%.

VI. Adaptation to Additional Constraints

In this section, we consider three scenario types, $k$-par, $q$-fail and $d$-unknown, as mentioned in the introduction section.

- $k$-par: each fiber link can only have up to $k$ wavelengths(traversed by $k$ m-trails).
• $q$-fail: failed fiber links can still deliver signals of up to $q$ wavelengths, which takes places in Optical cross-connect port intrusions [8].

• $d$-unknown: no knowledge over the upper bound of failure quantity is obtained, which happens frequently for large-scale networks with complicated structures.

A. Case 1: $k$-par

We adapt both proposed algorithms to tackle this.

For TDL, after locating the $v$-Span graph for each vertex $v$, since the m-trails on each tree are deterministic, the load of each link (the number of trails traversing it) is also known, and we can label the links with heavy load (> $k$). By avoiding selecting these tree-subgraphs with heavy load links, the algorithm is feasible. On the other hand, for RWL, since the sole transmitter and receiver have inevitable transmission load, we do not consider this constraint on the edges outgoing from the transmitter or entering the receiver, but it applies to all the other edges. Threshold on the trail counter used for our local rarest first strategy addresses this problem.

As the simulation results in Fig. 4(a) show, in a Power-Law network with average degree 2, as the link failure rate ranges from 1% to 4.5% when restricting the maximum number m-trails traversing each link as $k = 18$ (according to the standard of CWDM), the required number of m-trails keeps almost the same with the case of un-restricted $k$. When the failure rate grows bigger than 4.5%, we require more m-trails since the constraint on $k$ starts to disqualify some random walks.

B. Case 2: $q$-fail

In the case that the wavelength signals of some m-trails can still pass some failed links, false negative rate increases. Assume that each failed link can never support more than $q$ wavelengths, then the certificate of a good link is to appear in at least $q + 1$ negative m-trails. Therefore, for TDL, by duplicating each m-trail by $(q + 1)$ different wavelength signals, i.e. with $(q + 1) \min((D(G) \log m), (D(G) + \log^2 m))$ m-trails, all the failures can be unambiguously localized.

On the other hand, for RWL, the error-tolerant $(d, q)$-disjunct [16] matrix which can be used to localize $d$ positive items in the presence of $q$ error tests, comes into picture. More formally, it is required that each column $C_i$ has at least $q + 1$ different rows $r_j$ where $M[r_j, c_i] = 1$ and $M[r_j, c_k] = 0$ for any other $d$ columns $c_k$ with $k \neq i$. [15] states that with the same random walk construction, the resulted matrix with $O(c^2d^2T^4(m)\log(m/d)/(1-p)^2)$ rows has an overwhelming probability to be $(d, q)$-disjunct matrix, where $q = \Omega(pd\log(m/d)/(1-p)^2)$ for arbitrary $p \in [0, 1]$. Therefore, through some calculations, the RWL algorithm for handling this scenario requires $\frac{2q^2d^2\log^2(m)}{1-2\sqrt{-\log_{\log_{d/q}}\left(1+2q\right)}}$, i.e., $O(qd^2 \log^3 m)$, m-trails in the worst-case. In our simulation (Fig. 4(b)), when $q$ increases from 1 to 5, the cost increases at most by 4 times (when $d/m = 7\%$), which matches the derived bound. However, the curve for $q = 10$ almost coincides with that for $q = 5$, so when $q$ is bigger enough, its influence over the number of trails is negligible. Therefore, RWL is feasible for various values of $q$.

C. Case 3: $d$-unknown

When the number of faulty links is unknown, all the existing solutions fail at the first step. Therefore, how to estimate this value has to be solved beforehand. Trail-and-error methods are inapplicable since the long localization latency is not affordable, while guesses over $d$ may result in an unnecessary large value, which draws too many m-trails.

Since the failure localization procedure will not keep running all the time, the event that triggers it could be some anomalies detected at the network, for example, the throughput or PDR (packet delivery ratio) between specific node-pair sharply decreases. We simply denote this measure for the global network $G$ as $M(G)$, and assume that the localization procedure is triggered when $M(G)$ falls below specific threshold. Apparently, $M(G)$ is related with some function regarding with the graph topology. Therefore, it is possible to
investigate the graph topology in the offline step to provide an estimation over $d$ corresponding to the $\mathcal{M}(\mathcal{G})$ value. Since our work is orthogonal with this detection phase, but focuses on the localization step, for simplicity, we formulate a function \( \text{pairwise routing connectivity} \ CONN(G) = \sum_{s,t \in V} \zeta(s,t) \) where the routing connectivity \( \zeta(s,t) \) between any node pair \((s,t)\) is 1 if there is at least one operational path from \( s \) to \( t \), 0 otherwise. Therefore, \( CONN(G) \) refers to the communication ability between any node pair.

Based on this observation, we resort to an avg-$\beta$-disruptor problem, which finds an average size of edge set whose deletion from the graph makes \( CONN(G) \leq \beta \), to approach the size of the real link failures. Specifically, for a network with \( CONN(G) \leq \beta \), we first guess a value \( d_x \) as the number of failures, randomly choose \( d_x \) links as failures and check if the corresponding \( CONN(G) \leq \beta \), if not, increase \( d_x \) till this is satisfied. For each network instance, we obtain an estimated \( d_x \) by averaging the results of 100 executions. We investigate its performance on a Power-Law network with \( m = 1000 \) links and an increasing failure ratio from 1% to 9%, using the RWL algorithm. As shown in Fig. 4(c), \( 5d_x \log m \) m-trails is enough for the networks with the same size, topology density and failure rate, so we use \( t = 5d_x \log m \) m-trails to check the localization error rate and the cost ratio over the theoretical bound. In our simulation, at least 90% failed links can be correctly localized.

VII. CONCLUSIONS

In this paper, we present two efficient \( m \)-trail algorithms to tackle the multiple link failure localization problem in all-optical networks. The tree-decomposition base algorithm catches most of the link failures for small dense networks with un-restricted monitoring locations, and the random walk based algorithm handles large scale sparse networks with single-pair monitoring location. For implementation purpose, we also consider various practical constraints and include adaptations to our scheme correspondingly. These adaptations, especially the one for unknown failure quantity, provide a general solution framework associated with network vulnerability assessment, which could enlighten the further researches and developments. Since the principle of the provided solutions matches the group testing theory, they are also capable for anomaly detection problem in various network types, not limited in all-optical networks.

REFERENCES