Disjoint-Path Segment Routing: Network Reliability Perspective

Yaser Al Mtawa
The University of Winnipeg, Winnipeg, MB, Canada
y.almtawa@uwinnipeg.ca

Anwar Haque
Western University, London, ON, Canada
ahaque32@uwo.ca

Greg Sidebottom
Juniper Networks, Kanata, ON, Canada
gsidebot@juniper.net

2022 International Wireless Communications and Mobile Computing (IWCMC) | 978-1-6654-6749-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/IWCMC55113.2022.9825466

***Abstract*— Achieving five-nines reliability in communication networks is of paramount importance to network operators. Adopting multipath Segment Routing (SR) in networking offers several key advantages in load balancing, optimal link utilization, reducing congestion, and enhancing reliability. Equal-cost multipath (ECMP) routing falls short to effectively utilize a multipath approach because ECMP may still forward traffic through the specific paths resulting in decreased network reliability. In this paper, we propose a novel two-phase Disjoint-path SR-based reliability framework to enhance the current network reliability. This framework drives network traffic flows through disjoint paths, resulting in effective load balancing and improving network reliability to meet stringent requirements of various services. Our proposed framework includes the following: a) Phase I: Optimization phase uses Mixed-Integer Linear Programming (MILP) formulation to find the optimal multipath that minimizes the maximum utilization of a given network while transmitting traffic between two terminal nodes, and b) Phase II: This phase splits the multipath of phase I into two disjoint multipaths, namely working and backup. The proposed framework is implemented, and its validation and effectiveness are demonstrated through simulation results.**

Keywords—Communication Network, Edge-disjoint MultiPath, Reliability, Segment Routing, Mixed-Integer Linear Programming.

I. INTRODUCTION

In recent years the adoption of segment routing (SR) in communication networks has increased rapidly. This increase is due to SR-enabled networks' features compared to the MPLS/IP-based networks, including the Quality of Service (QoS), network management, maintenance efficiency, and network performance optimization. The key idea of SR is to perform routing using few logical segments to form a source-routed path (SRP) between the source and destination nodes. Each segment has a segment ID (SID). In this way, each intermediate switch only needs to know the next SID in the SID stack and forward packets.

Using Menger's theorem, the number of edge-disjoint paths (i.e., path diversity) between a pair of nodes, s and t, can be calculated as a minimum edge cut set [1]. The application of Ford-Fulkerson (or Edmonds-Karp [2]) can be utilized to generate multiple disjoint paths connecting a pair of terminal nodes. However, this method provides only linearly independent path diversity. Such path diversity will exploit the bandwidth and increase the possibility of overloading. Furthermore, Internet interior gateway protocols (IGPs) such as Open Shortest Path First (OSPF) routes flow on the shortest paths. If there are multi-shortest paths from a source to a destination, then the traffic flow is split equally among all the outgoing links that are part of the shortest paths. This equal splitting of flow is usually called the Equal-Cost Multipath (ECMP) principle. However, OSPF entirely depends on the weights/costs that are assigned to the networks' links. Setting the OSPF weight is an optimization problem. Using the weights according to the link metric of IGP only may cause network congestion as more traffic flows will use only the OSPF paths.

Motivated by the previous-mentioned challenges of current routing solutions, in this paper, we propose a framework that overcomes the linearity of independent disjoint paths and the weight-setting required for OSPF to optimize the objective functions such as congestion and utilization. In this regard, network operators would be interested in utilizing all possible disjoint paths that meet their networks' routing optimization considerations. Having two edge-disjoint mutipaths, working and backup, would benefit network operators to manage their traffic reliably.

In this paper, we enable network operators to exploit the features and capabilities of SR, take advantage of the equal-cost multipaths (ECMP), and seek load balancing within a network. The main contributions of this paper are summarized as follows:

1) Propose a two-stage framework for the edge-disjoint multipaths problem. We formulate a MILP problem to optimize the traffic routing through a multipath such that the maximum utilization of links in a network is minimized.
2) We design a splitting algorithm called SiT. While the MILP generates single multipath, SiT splits this multipath into two edge-disjoint multipaths: working and backup.
3) We conduct extensive experiments on various types of network topologies to show the validation and effectiveness of our framework and splitting scheme. We employ networkwide cost and reliability as key indicators of the effectiveness of our proposed framework.

The remainder of this paper is structured as follows: Section II presents related work. Network modeling and problem formulation will be addressed in Section III. Section IV is devoted to providing our proposed SR-based reliability framework. Experimental results that contain our framework's validation and effectiveness will be conducted in Section V. We conclude the paper in Section VI.

II. RELATED WORK

The problem of finding edge-disjoint paths (EDP) is different from calculating the *maximum number* of EDP between a pair of terminal nodes. **Menger's** theorem [1] **Edmonds** et al. [2] enable finding the maximum number of edge-disjoint paths. On the other hand, **Yen's** algorithm [3] computes single-source *K*-shortest paths for a graph of

nonnegative link cost. It is an iterative algorithm that employs any shortest path algorithms such as the Dijkstra algorithm [4] to find the shortest/optimal path. In the next iterations, it finds the subsequent K-1 deviations of the best path.

Similarly, Open Shortest Path First (**OSPF**) finds all shortest paths between two terminal nodes. However, OSPF applies the ECMP principle to split the flow evenly at outgoing links that belong to the shortest paths. Both Yen's and OSPF do not provide edge-disjoint paths, and they require optimizing the weight values of the network links to achieve the application's objective function.

Finding the maximum EDP problem is known to be an NP-hard optimization problem [5]. Martin et al. [5] **combined** ILP and an evolutionary algorithm to deal with the hardness of this problem. Most recently, Pereira et al. [6] proposed an Evolutionary Computation approach to encode routing paths using only three segments. Similarly, Li et al. [7] provided MILP to transmit traffic load between two terminal nodes using paths with no more than K segments to minimize the maximum link utilization in the topology. Unlike [7], which finds multiple paths with traffic splitting, [8] and [9] focused on finding one K-segment routing for each flow with no traffic splitting.

Note that the reliability framework proposed in our paper is different from [6]–[9] as these works aim at improving network performance by optimizing SID's stack size. The objective is to maximize the number of node SIDs over the number of adjacent SIDs. However, this encoding-based approach has a drawback that reduces the number of paths in the generated multipath. On the other hand, our framework's objective is to optimize the link utilization without limiting the size of the generated multipath. Furthermore, the works in [7]–[9] do not address the edge-disjointness of the multipath of SRPs, while our paper also focuses on network reliability where we design an algorithm to split the MILP-generated multipath into two edge-disjoint multipaths: working and backup.

III. Network Modeling and problem formulation

A communication network is mathematically modeled as a graph $G = (N, L)$, where N and L represent the sets of vertices/nodes and edges/links in a graph G. L describes the connectivity between the network's nodes. $L \subseteq \{\{i,j\} \mid i,j \in N \text{ and } i \neq j\}$, i.e., undirected graph. Link direction is modeled as an ordered pair of nodes as follows LD = $\{(i,j),(j,i) \mid \{i,j\} \in \mathrm{L}\}$. Directed link (i,j) (or $e_{i,j}$) is also called an arc over which data traffic flows from i *to* j. Each arc $e_{i,j}$ has a traffic engineering (TE)-numerical metric value $a_{i,j}$ which is a real number. Let $n = |N|$ and $m = |L|$ be the number of nodes and links, respectively. In this paper, we focus mainly on links as elements of L that could fail. A simple path between two nodes s and t is defined as a set of adjacent links that connect s with t with no repetition of nodes. Multiple paths with different costs/lengths could be available to connect s and t. Multipath between (s,t) is defined as an abstraction of the fundamentals of a segment and how it is used in a network. Each arc of the multipath has a bandwidth fraction (BWFrac) and a capacity.

Given a graph G that represents a communication network. Let $(s_k, t), 1 \leq k \leq n$ be k-terminal pairs. Through these pairs, traffic flows are expected to flow from s_k to t. Let $\aleph = \{p_1, p_2, \ldots, p_r\}$ denote the set of all possible paths p_i that connect (s_k, t). p_i is sorted by a length in ascending order such as $p_1 \leq p_2 \leq \cdots \leq p_r$, where $1 \leq r \leq \sum_{j=0}^{n-2}(n-2\,j\,)j!$. The upper bound of r represents the size of set $\aleph$ when G is a complete graph of n vertices. The goal is to find a single set of p_i so that the maximum link utilization is minimized, then split this set into two edge-disjoint multipaths, namely working and backup. The larger and comparable sizes of these two multipaths, the better.

IV. The Proposed Reliability Framework

MILP solvers such as CPLEX [12] can solve some difficult combinatorial problems with limited size. However, the solver often falls into computational intractability for large-scale problems, and no optimal solution is produced. Therefore, our proposed framework effectively combines two approaches (i.e., optimization and heuristic techniques) to overcome this hardness.

A. Phase One: MILP Formulation for EDP

In this phase, we formulate MILP to find single multipath that flows traffic from k −source nodes to a destination node so that the maximum link utilization is minimized.

- *Input to the optimization problem*

The arc data structure includes four members: fromNode, toNode, capacity, and cost. The bandwidth available/capacity on an arc (i,j) is referred to by $c_{ij} \in \mathbb{R}^+ \cup \{0\}$. We further refer to the cost using an arc (i,j) by $d_{ij} \in \mathbb{R}^+ \cup \{0\}$. Furthermore, D represents a demand matrix such that the element $D(s,t)$ provides the demand in traffic flow between a pair $(s,t) \in N \times N$.

- *Decision Variables*

$f^t_{(i,j)}$ represents how much traffic passes over arc (i,j) towards node t. We use u_{ij} to denote the sum of traffic loads overall demands go over (i,j). The Boolean decision variable $\beta^t_{(i,j)}$ indicates whether arc (i,j) will carry any traffic towards node t. Further, the length in TE metric cost from a node j to another node t is given by l^t_j. $z \in \mathbb{R}^+ \cup \{0\}$ represents a maximum utilization of links networkwide.

- *Validity and Continuity*

Conservation of traffic flow to ensure the desired traffic demand from s to t is guaranteed through the following constraints. $\forall\, i,t \in N$:

$$\sum_{\substack{(i,j)\in LD \\ i=t}} f^t_{(i,j)} - \sum_{\substack{(j,k)\in LD \\ k=t}} f^t_{(j,k)} = -\sum_{k\in N} D(k,t) \quad (1)$$

$$\sum_{(i,j)\in LD} f^t_{(i,j)} - \sum_{(j,i)\in LD} f^t_{(j,i)} = D(i,t)\,, i \neq t \quad (2)$$

Another validity and continuity constraint is to keep the amount of flow transmitted over any arc is less than its capacity (i.e., available bandwidth). We formulate this constraint as follows.

$$\sum_{t\in N} f^t_{(i,j)} \leq c_{ij}, \forall (i,j) \in LD \quad (3)$$

- *Bandwidth and Length Constraints*

$$f^t_{(i,j)} \leq \sum_{k\in N} D(k,t) \;\; if \beta^t_{(i,j)} = 1, t \in N, (i,j) \in LD \quad (4)$$

This constraint reads as the amount of *traffic load* passes to t and ends up on link (i, j) is less than the sum of demands towards t. Given that arc (i, j) was chosen for traffic delivery. To keep the candidate paths of multipath as shortest as possible, we have the following constraints.

$$0 \leq l_j^t + d_{ij} - l_i^t \quad (5)$$
$$l_j^t + d_{ij} - l_i^t \leq (1 - \beta_{(i,j)}^t) \quad (6)$$
$$(1 - \beta_{(i,j)}^t) \leq l_j^t + d_{ij} - l_i^t \quad (7)$$

- *Utilization*

$$f_{(i,j)}^t \leq z * c_{ij}, \forall (i,j) \in LD, t \in N \quad (8)$$

This constraint maximizes link utilization.

- *Complete Optimization Formulation*

We summarize below the complete MILP formulation as follows: $min\ z$

$$subject\ to\colon\ (1) - (8)$$

B. Phase Two: Finding Working and Backup Multipaths

Splitting the result of our MILP into two multipaths is not straightforward as we seek to maximize the size of the two multipaths. In this subsection, we propose an algorithm called SiT that divides the MILP-generated multipath into two edge-disjoint multipaths. The basic idea of SiT algorithm is two-fold: distribute the links incident to source nodes between working and backup multipaths, and then resolve the conflict between working and backup paths over shared links.

Algorithm 1: Splitting Single-Multipath into Two-Disjoint Multipaths (SiT)

Input: single multipath G' from phase one, src, and des.
Output: two disjoint multipaths: working and backup

```
1  all_freq = []
2  src_incident_links = incident (src)
3  for each e in src_incident_links
4      freq_e = find_freq(e, G')
5      all_freq = [all_freq freq_e]
6  end
7  sorted_all_freq = sort(all_freq, 'descend')
8  working_e = get_odd_indices(sorted_all_freq)
9  backup_e = get_even_indices(sorted_all_freq)
10 working = get_paths_including(working_e)
11 backup = get_paths_including(backup_e)
12 intersection = find_intersection(working, backup)
13 sorted_pths = sort(intersection, 'desc')
14 for path in sorted_pths
15   if links(path) in working  > links(path) in backup
16     remove(path, backup)
17     if no_conflict_with(backup, path)
18         add(path, working)
19     end
20   else
21     remove(path, working)
22     if no_conflect_with(working, path)
23         add(path, backup)
24     end
25   end
26 end
```

The first step distributes the first-mile links among the two multipaths. For better fairness in this distribution, we find the appearance's frequency of each link in MILP G', sort it, and distribute it one by one among these two multipaths, as shown in lines 3-9. In Lines 10-11, we assign the G' paths between working and backup sets according to their connectivity to the links incident to sources which we already distributed in the previous step. In the second fold/step, SiT disengages the conflict over shared links. *find_intersection* function in Line 12 identify the links that are shared between working and backup multipaths. This function includes multiple loops to iterate over every path in working and backup multipaths to check their overlapping. Two paths in working and backup have a common link are called conflicting paths. Such conflict needs to be resolved by removing either path. The removal of which path depends on how many other paths conflict with this particular path. That is, the higher the path in conflict with other paths (from the other set), the more probability to be removed. This is shown in lines 13-26.

SiT finds link-disjoint working and backup multipaths successfully using a two-step strategy. However, the number of paths in these two multipaths can be further improved. In the following subsection, we propose Enhanced SiT (E-SiT) and illustrate how this would elevate the quality of results regarding the number of paths included in both sets.

- E-SiT Methodology

Maximizing the number of paths in working and backup requires evaluating the loss and gain for any possible removal of conflicting paths. We propose a sophisticated method that employs release (r), sacrifice (s) and keep (k). The key indicators (KIs) r, s, and k refer to the number of **removed** conflicting paths, the number of **links** that are **not shared** anymore, and the number of **saved** paths that are not any more conflicting, respectively. We call these KIs a RSK. The following formula defines RSK to determine which paths should be removed first.

$$RSK = \frac{r + k}{s} \quad (9)$$

A higher RSK value means an increased probability to remove its associated conflicting paths. Next, we demonstrate how RSK works. Fig. 1 shows the single multipath with 28 individual paths after the SiT/E-SiT assignment of links incident to source P2 to either working (in blue) or backup (in red) multipaths. Dashed links are not part of these multipaths.

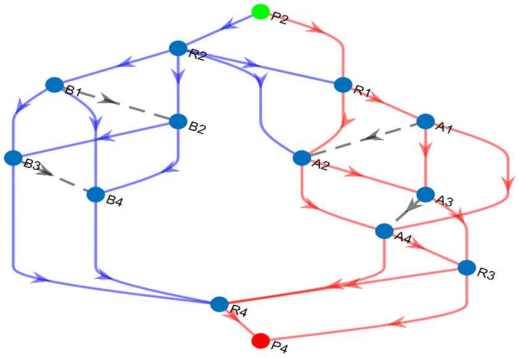

Fig. 1. An illustration of the MILP single multipath after the first stage of SiT/E-SiT.

Table I shows all conflict links and the conflicting paths in working and backup multipaths in Fig. 1. Note that each multipath has its own indexing for its paths. For example, path index 1 in column two is different from path index 1 in column three as the former belongs to the working multipath while the latter belongs to the backup counterpart.

TABLE I: RSK required parameters- the number of shared links, the conflicting working paths, and the conflicting backup paths for each shared link.

Links of conflict (Index, Name)	Conflicting **working** paths (Path index)	Conflicting **backup** paths (Path index)
7, R1-A2	1	1, 2
8, R1-A1	2	3, 4
11, A2-A4	1, 3	2
13, A1-A4	2	4

For better visualization of the proposed method, the actual sequence of each conflicting path is included in Table II.

TABLE II: The indices and actual conflicting paths for both working and backup multipaths.

Path index		Path
Working	1	P2 - R2 - R1 - A2 - A4 - R4 - P4
	2	P2 - R2 - R1 - A1 - A4 - R4 - P4
	3	P2 - R2 - A2 - A4 - R4 - P4
Backup	1	P2 - R1 - A2 - A3 - R3 - P4
	2	P2 - R1 - A2 - A4 - R3 - P4
	3	P2 - R1 - A1 - A3 - R3 - P4
	4	P2 - R1 - A1 - A4 - R3 - P4

We define the contribution of a conflicting path of particular multipath towards a shared link (i, j) as the number of paths that belong to the counterpart multipath and share (i, j). As a concert example, in Table I, a link with index 8 (i.e., R1-A1) is included in three conflicting paths: a path of index 2 in working multipath and paths 3 and 4 in backup multipath. Thus, the contribution of the path of index 2 in working multipath towards the conflict over link R1-A1 is two because there are two paths in the backup counterpart. However, the contribution of paths 3 and 4 towards the conflict over the same link is only one.

Next, we calculate the r, s, and k variables to determine the RSK value for each possible path's removal. Table III shows the effect of the removal order of conflicting paths for resolving the conflict. In this Table, we calculate RSK variables based on Table I. The removal order approach focuses on each shared link, calculates the r, s, and k variables for two cases: 1) what if working conflicting paths are removed, and 2) what if backup paths are removed. In either case, we use the naming convention W- or B- to indicate the working and backup multipaths, respectively, and then followed by the paths' indices. For example, Table III shows two cases to disengage the conflict over the link A2-A4: W-1,3 which means removing working paths of indices 1 and 3, while the second case is B-2 which indicates removing backup paths of index 2. We proceed with this example to calculate RSK.

Case1 (W-1,3): removing conflicting working paths 1 and 3 will sacrifice two paths (1 and 3 themselves). Thus, s is equal to 2. The released shared links by this removal is 2 because, besides link A2-A4, link R1-A2 will also be un-shared as working path 1 as shown in row one of Table III. Hence, $r = 2$. Finally, $k = 2$ because paths 2 and 1 will be saved in the backup multipath. We follow the same process with all other rows.

TABLE III: Calculating the RSK variables out of TABEL I.

Variable/ Path Removal		Release (r)	Sacrifice (s)	Keep (k)	RSK
R1-A2	W-1	1	1	2	3
	B-1,2	2	2	2	2
R1-A1	W-2	2	1	2	4
	B-3,4	2	2	1	1.5
A2-A4	W-1,3	2	2	2	2
	B-2	1	1	2	3
A1-A4	W-2	2	1	2	4
	B-4	1	1	1	2

The highest value of RSK occurs by removing W-2, and its value is 4. After removing path 2 from the working multipath, we repeat the whole calculation and remove the next highest RSK value until all conflicts are resolved. The result of E-SiT is shown in Fig. 2.

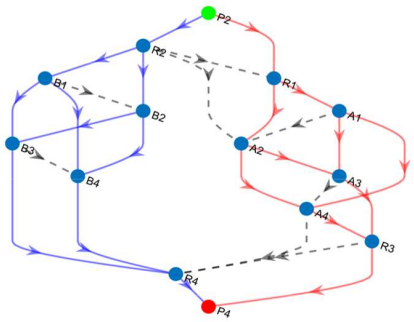

Fig. 2. Two edge-disjoint multipaths: working in blue and backup in red. The dashed links are either not used by the original single multipath or deemed unusable due to the conflict resolution method in E-SiT.

Fig. 2 shows the removal of paths that share links R2-R1, R2-A2, A4-R4, and R3-R4. Accordingly, there are four paths left for each multipath. On the other hand, the original SiT produced 5 and 2 paths for working and backup multipaths. E-SiT provides better balance and increases the total number of these paths as well.

V. EXPERIMENTAL RESULTS

This section shows the validation and effectiveness of our proposed approach against networkwide cost and reliability performance metrics. For validation, we conduct our experiment using the network in Fig. 3, which contains 34 nodes and 82 edges. This topology represents a User Plane Function (UPF) architecture, a key component in a 3GPP 5G core infrastructure. In Fig. 3, there are 10 sites (0-9), including 8 anycast groups (0-7). Sites 8 and 9 are allocated for source and destination nodes. Each anycast group is a complete graph of 4 nodes, i.e., K4. All of the links between any two sites use the same TE metric, i.e., 50. The TE metric is set to 10. We adopt the TE metric as an optimization objective rather than the Interior Gateway Protocol (IGP) metric. We use MATLAB R2019b for all simulations and run them on x64-based Intel Core i5-8250U CPU @ 1.60GHz with 8 GB of RAM, running Windows 10.

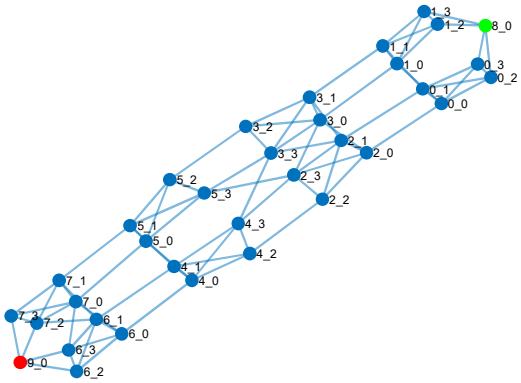

Fig. 3. Network topology including 10 sites and 8 anycast groups.

1) Validation

In this experiment, 20 units of data traffic were transmitted from node 8_0 to node 9_0. The solution of our MILP for the network in Fig. 3 is the whole graph in Fig. 4. Even splitting of traffic is clearly shown among ECMPs. As expected, the MILP balanced the traffic load in the network, and there is symmetry in the output multipath. To split this multipath, we use our E-SiT algorithm. The final result is also shown in Fig. 4, in which the working multipath is in blue color while the backup counterpart is in red color.

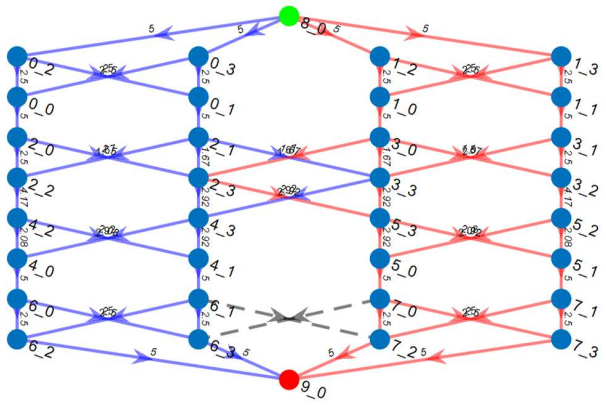

Fig. 4. The MILP-based multipath is split into two edge-disjoint multipaths: working in blue and backup in red. The label on arcs refers to the load passing over this arc, and dashed links carry no traffic.

The number of individual paths in the original MILP multipath is 160. The first links' assignment divided the paths evenly among the working and backup multipath. The final result shows 40 paths for each.

2) Effectiveness

We compare our proposed framework to the inverse-capacity OSPF (InvCap-OSPF) routing solution. InvCap-OSPF sets link weight inversely proportional to its capacity. For example, in Fig. 5, InvCap-OSPF will assign weights of value 1 to all links except for links (D, G), (E, F), and (F, G) which will be assigned a weight of ½ as their links' capacity is 2.

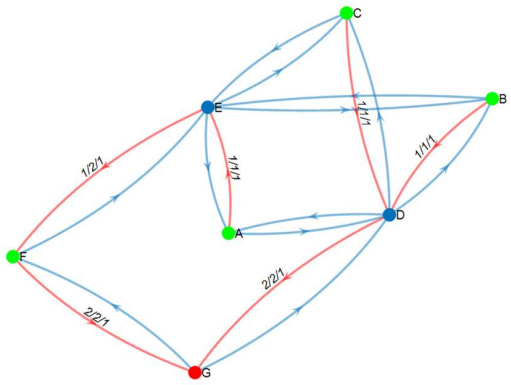

Fig. 5. Our MILP result (in red color) for demands of 1 unit between 4 source nodes (in green) and one target node (in red). Links labeled as triplet traffic-load units/capacity/weight.

- *Networkwide Cost Function*

In this experiment, we employ the link cost function as proposed in [10] to compare our MILP solution and InvCap-OSPF. The link cost increases as a function of utilization, with exponential growth as utilization exceeds 100%. The networkwide cost of a routing scheme is the sum of all links' costs. To conduct our experiment, we use the network and weight settings in Fig. 5. Fig. 5 shows traffic load demand of value 1 between source nodes (A, B, C, and F) towards destination node G. To calculate the networkwide cost, traffic loads take the following 12 values [0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.2]. Fig. 6 shows the result of our proposed framework versus the InvCap-OSPF. Each value in this graph represents the networkwide cost that occurs at the corresponding load value.

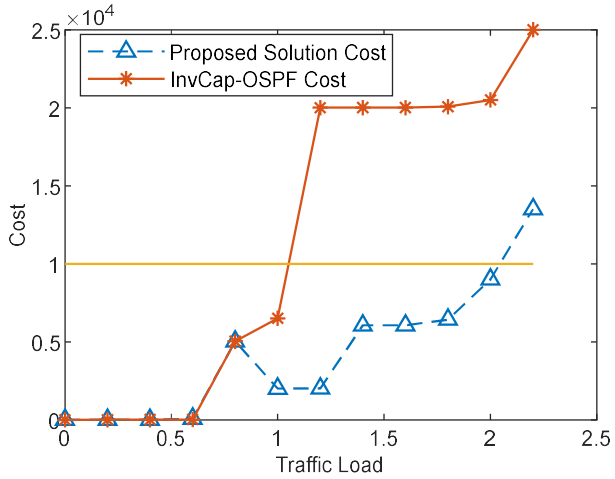

Fig. 6. Networkwide cost for our MILP solution versus InvCap-OSPF.

Fig. 6 shows that InvCap-OSPF makes the network overloaded in an early stage of iteration over traffic load values. On the other hand, our proposed MILP enhances the cost networkwide; It handles ~80% more traffic load than the InvCap-OSPF before reaching the horizontal line at cost 1. This is because our proposed solution pushes traffic flows to balance the network and minimize the maximum utilization of links. Hence, avoiding high penalties for utilizations beyond 100%.

- *Network Reliability*

This experiment shows the reliability effectiveness of E-SiT compared to SiT. We test the traffic load interruption under a failure process when traffic loads are transmitted between (s,t)-pair terminals using the multipath generated by SiT and E-SiT. A successful packet delivery between any (s,t)-pair nodes through either working or backup multipath is valued by one (i.e., high reliability); otherwise, it is zero. For this purpose, we use a network similar to the one in Fig. 3. The network contains 30 anycast groups linked together with a connectivity density parameter, densPara=0.5. A random selection of C pairs of these groups is connected where C is the largest number such that $C < \text{NumPairs} \times \text{densPara}$. In our case, C=217, n=120, and m=2104. Two random nodes from each C anycast pair are selected and connected so that each of the two nodes is linked to the two nodes in the other anycast pair (i.e., in a complete bipartite fashion). We randomly assign a failure probability value to each edge. We iterate this assignment process 30 times. In each iteration, 20 edges with the highest failure probability will be removed, one at a time. For each edge removal, reliability will be calculated. The average of 30 values for every edge removal will be calculated. In this experiment, we choose two (s,t)-pair nodes (i.e., k=2), one for each multipath. The marked points in Fig. 7 represent these average values.

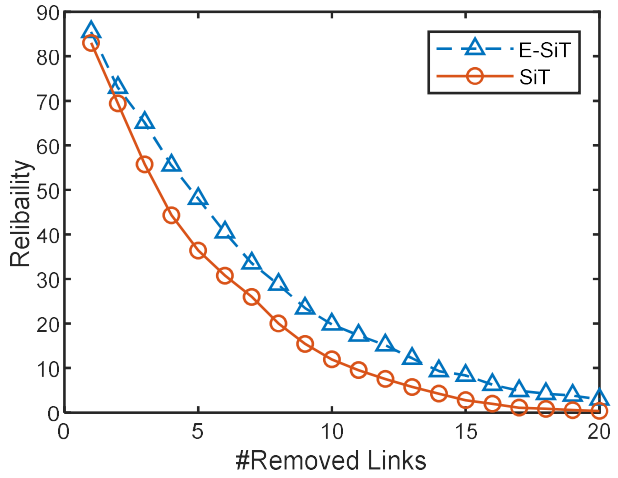

Fig. 7. Network reliability under successive link failure: SiT vs. E-SiT.

As shown in Fig. 7, E-SiT is further effective in network reliability. At the beginning of the failure process, the reliability values are comparable with a little advantage for E-SiT over SiT. This is because the network did not lose many individual paths at this stage. As more links fail, E-SiT mitigates the impact of link failure and keeps network reliability about 47% higher than the network performance under SiT application. At the last part of the failure process, SiT-based reliability collapses much faster than the E-SiT counterpart. This is because E-SiT has more individual paths that can handle more failures and are still able to deliver traffic loads.

VI. CONCLUSIONS

In this paper, we provide a framework for engineering the traffic flow in segment routing-based networks. Our framework is twofold: In the first phase, the MILP problem is formulated to calculate the optimal routing of multi-source single-destination traffic flows to minimize the maximum utilization of network links. The second phase, edge-disjoint multipaths are created out of the MILP multipath generated in the first phase. To establish the two edge-disjoint working and backup multipaths, we design an algorithm called SiT. SiT finds the two edge-disjoint multipath successfully. However, we further enhance SiT to E-SiT to capture the essence of path confliction over shared links regarding the loss and gain of conflicting individual paths that construct these two multipaths.

Our experiments show the validation and effectiveness of our framework. We compare our proposed solution to InvCap-OSPF routing solution. We employ two performance metrics for this comparison: Networkwide cost and reliability. Our solution shows more than an 80% increase in handling traffic loads than the other solution, with 47% higher reliability. Evaluating the quality of working and backup mutlipaths in terms of edge-cut sets, centrality measures, and operational paths is an interesting topic to investigate in the future.

ACKNOWLEDGMENTS

This work is supported by Mitacs and Juniper Networks.

REFERENCES

[1] Y. Egawa, A. Kaneko, and M. Matsumoto, "A mixed version of Menger's theorem," *Combinatorica*, vol. 11, no. 1, pp. 71–74, Mar. 1991, doi: 10.1007/BF01375475.

[2] J. Edmonds and R. M. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," *J. ACM*, vol. 19, no. 2, pp. 248–264, Apr. 1972, doi: 10.1145/321694.321699.

[3] J. Y. Yen, "Finding the K Shortest Loopless Paths in a Network ," *Manage. Sci.*, vol. 17, no. 11, pp. 712–716, Jul. 1971, doi: 10.1287/mnsc.17.11.712.

[4] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959, doi: 10.1007/BF01386390.

[5] B. Martín, Á. Sánchez, C. Beltran-Royo, and A. Duarte, "Solving the edge-disjoint paths problem using a two-stage method," *Int. Trans. Oper. Res.*, vol. 27, no. 1, pp. 435–457, 2020, doi: 10.1111/itor.12544.

[6] V. Pereira, M. Rocha, and P. Sousa, "Traffic Engineering with Three-Segments Routing," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 3, pp. 1896–1909, Sep. 2020, doi: 10.1109/TNSM.2020.2993207.

[7] X. Li and K. L. Yeung, "Traffic Engineering in Segment Routing Networks Using MILP," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 3, pp. 1941–1953, Sep. 2020, doi: 10.1109/TNSM.2020.3001615.

[8] C. K. Dominicini *et al.*, "KeySFC: Traffic steering using strict source routing for dynamic and efficient network orchestration," *Comput. Networks*, vol. 167, p. 106975, Feb. 2020, doi: 10.1016/j.comnet.2019.106975.

[9] M. Jadin, F. Aubry, P. Schaus, and O. Bonaventure, "CG4SR: Near Optimal Traffic Engineering for Segment Routing with Column Generation," in *Proceedings - IEEE INFOCOM*, Apr. 2019, vol. 2019-April, pp. 1333–1341, doi: 10.1109/INFOCOM.2019.8737424.

[10] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proceedings - IEEE INFOCOM*, 2000, vol. 2, pp. 519–528, doi: 10.1109/infcom.2000.832225.