Graph Transformation Approaches for Diverse Routing in Shared Risk Resource Group (SRRG) failures

Pallab Datta and Arun K. Somani
Dependable Computing and Networking Laboratory
Department of Electrical & Computer Engineering
Iowa State University, Ames, IA, 50011, USA
E-mail: {pallab, arun}@iastate.edu

Abstract

Failure resilience is a desired feature of the Internet. Most traditional restoration architectures assume single-failure assumption, which is not adequate in present day WDM optical networks.

Multiple link failure models, in the form of Shared-Risk Link Groups (SRLG's) and Shared Risk Node Groups (SRNG's) are becoming critical in survivable optical network design. We classify both of these form of failures under a common scenario of shared-risk resource groups (SRRG) failures. We develop graph transformation techniques for tolerating multiple failures arising out of shared resource group (SRRG) failures.

Diverse Routing in such multi-failure scenario essentially necessitates finding out two paths between a source and a destination that are SRRG disjoint. The generalized diverse routing problem has been proved to be **NP-Complete**. The proposed transformation techniques however provides a polynomial time solution for certain restrictive failure sets. We study how restorability can be achieved for dependent or shared risk link failures and multiple node failures and prove the validity of our approach for different network scenarios. Our proposed technique is capable of improving the diverse route computation by around 20-30% as compared to approaches proposed in literature.

I. Introduction

WDM optical networks have evolved as the primary transport medium in modern day networks. Customers expect to see uninterrupted service, even in the event of failures such as power outages, equipment failures, natural disasters and cable cuts. Many optical-layer protection schemes for WDM networks have been proposed in the literature [1][2]. Protection schemes can be classified either as *link protection* or *path protection* based on the initialization locations of the re-routing process. Link protection schemes route a connection around a failed link. Path protection attempts to provide a backup path from the source to the destination that maybe independent of the working path. Path-based protection has been established to be the more capacity-efficient approach for mesh based networks as compared to link based rerouting schemes [1][3] and is also used in this work.

For end-to-end path based restoration, for each demand the network provides two diverse paths: the service path and the restoration path. When the service path fails, the traffic gets re-routed to the restoration path. There are two commonly used protection schemes: shared path protection and dedicated path protection. In case of shared path protection, spare capacity is shared among different protection paths, while in dedicated path protection, the spare capacity is dedicated to individual protection paths. Shared path protection, although more difficult to implement, is more capacity efficient than dedicated path protection [1].

The diverse routing problem is to find two paths between a pair of nodes in the optical layer such that no single failure in the physical layer may cause both paths to fail. The problem of finding two diversely routed paths in optical networks is much more difficult than the traditional edge/node disjoint

Pallab is presently working as a Technical Staff Member at Los Alamos National Lab. He can be contacted at pallab@lanl.gov

The research reported in this paper is funded in part by the National Science Foundation under grant ANI-9973102 and by Defense Advanced Research Projects Agency and National Security Agency under grant N6001-00-1-8949.

path problem in graph theory [11][21] because two fibers belonging to different links maybe routed using common conduits.

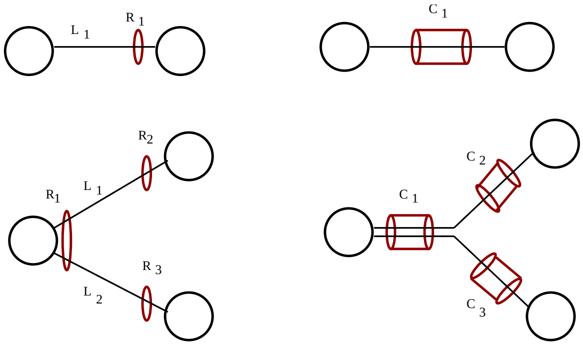


Fig. 1. Shared-Risk Link Groups and their corresponding physical routes.

Instances where separate fiber optic links share a common failure structure is often referred to as SRLG (Shared-Risk Link Group) [4][7]. Two examples of such shared-risk link groups are shown in Fig. 1, which illustrates two diverse fiber links that may be placed in the same conduit at the physical layer and are subject to a single point of failure. In this paper, we consider the diverse routing problem within a generalized framework where SRLG's are used to represent a set of optical links that are affected by a single failure in the physical layer. Finding a pair of diverse paths in the optical layer translates to finding a pair of SRLG-diverse paths.

The diverse routing problem has been conjectured to be NP-Complete in [9][10][11][21]. Recent studies have proven the NP-completeness of the generalized SRLG diverse routing problem [9][10][15][16]. The least coupled SRLG path problem, the minimum cost SRLG diverse routing problem and the routing problem under both wavelength capacity and path length constraints have also been shown to be NP-complete [9][17].

In [10][21], different heuristic approaches have been studied for diverse routing in presence of SRLG's. One of the common problems that arises in restoration path computation is the existence of a *trap topology* [10][18]. With a trap topology, if a service path is independently routed over a trap topology, then there may not exist a diverse restoration path, even though two diverse paths exist in the network.

In this paper, we address the problem of diverse routing in SRLG situations as well as multiple failures arising out of nodes sharing a common risk of failure. We classify both these sub-problems under a generalized scenario of shared risk resource group (SRRG) routing. We develop a polynomial time graph transformation heuristic for solving a sub-set of the generalized version of the diverse routing problem in networks with shared risk resource groups. We analyze the complexity of these routing methodologies and also validate the correctness of these algorithms.

The remainder of the paper is organized as follows: Section II will give a brief description of shared risk resource groups (SRRG's). Section III describes the graph transformation techniques for diverse SRLG routing. Section IV performs the complexity analysis of the SRLG disjoint routing methodology. Section V discusses about diverse routing in SRNG scenarios. Section VI presents the computational complexity for SRNG disjoint routing. We conclude the paper in Section VII.

II. Shared Risk Resource Groups (SRRG's)

An important class of SRRG's comprises of multiple links sharing a common component such as a duct, whose failure causes failure of all links in that group. Any physical failure of one of these ducts can invoke a logical failure of multiple links as illustrated in Figure 2(a). A single link can also be part of more than

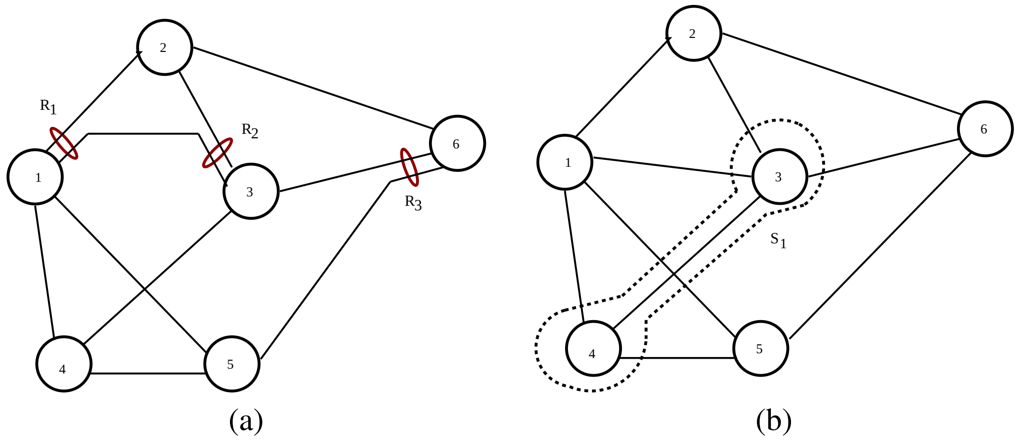


Fig. 2. Network With (a) Shared-Risk Link Groups (SRLG's) and (b) Shared Risk Node Groups (SRNG's).

one SRLG. As shown in Figure 2(a) the link connecting nodes 1 and 3 is part of two SRLG's R_1 and R_2. In our research, we concentrate on *co-incident SRLG's* [4], which are groups incident on a common node.

Another instance of a shared-risk resource failure (SRRG), is the failure of two nodes that are connected by a link and we model it as shared risk node groups (SRNG's) (Figure2(b)). In practice such a failure of one or more nodes may be due to simultaneous attack on two adjacent routers by a malicious user, which leads to simultaneous failure of two nodes in the network.

A. Trap Avoidance in Diverse SRLG Routing

In a diverse SRLG routing, there might be instances where an algorithm fail to find a pair of SRLG disjoint routes for a given source and destination due to topologically induced restrictions also known as *unavoidable traps*[18].*Unavoidable traps* are constraints imposed by the underlying topology, and cannot be worked around by any algorithm, i.e. there does not exist any SRLG disjoint paths between a source and destination node pair. For example if a network is not 2-edge connected then there is no algorithm that can guarantee the presence of two SRLG disjoint routes in the topology. *Avoidable traps*, however are traps that are not imposed due to the underlying topology, but are due to shortcomings of the routing algorithm.

These traps can be easily avoided by choice of an intelligent routing algorithm. Some heuristics have been proposed in [8][18] for elimination of avoidable traps in a topology. Several approaches for avoiding traps in a topology has been also studied in [12][14]. The concept of trap topology has been explained in [12] in the context of finding k-shortest paths. The work in [14] discusses the effects of dual failures arising from Shared Risk Link Groups(SRLG's). We study the concept of traps in an attempt to find SRRG (both SRLG and SRNG) disjoint routes.

The graph transformation techniques suggested in our work attempts to find shortest cycles in a transformed graph, instead of using Dijkstra's shortest paths, and map these cycles on the actual topology. Since the shortest cycle algorithm [21] is used in our graph transformation algorithm, and the shortest cycle guarantees finding the cycle that is of minimal edge length (if the topology is 2-edge connected and such disjoint routes exists), our approach can always successfully find SRLG disjoint routes in a given topology.

One of the traditional approaches for computing diverse SRLG routes, is to compute the shortest path using the Dijkstra's algorithm from a node *S* to node *D*. For all the edges in the primary or service path,

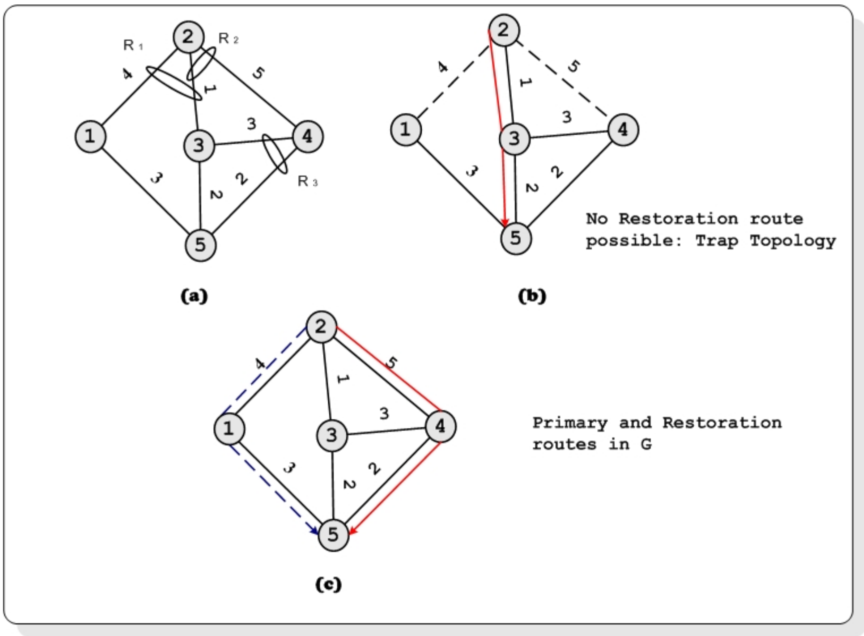


Fig. 3. Trap Avoidance in Diverse SRLG Routing

remove the edges that share a common failure with these edges. The Dijkstra's algorithm is again run on the residual network to compute the second shortest path from *S* to *D*. When routing over a trap topology, the pre-selected service path may not have a diverse restoration path, even though diverse paths exist in the network. For example, Figure 3(a) shows a network topology with the shared risk link groups and the edge weights. Computing the SRLG disjoint paths using the traditional Dijkstra's algorithm would lead to selection of the path $2 \rightarrow 3 \rightarrow 5$ as the primary path, thus making selection of a restoration route infeasible. However the graph as shown in Figure 3(c) does have shortest cycles which can be computed using an elegant routing methodology. As can be seen, the cycles includes two shared risk diverse paths P_1: $2 \rightarrow 1 \rightarrow 5$ and P_2: $2 \rightarrow 4 \rightarrow 5$.

B. *Graph Transformation for Diverse SRRG Routing*

In the following sections we present a graph transformation technique to search for two shared-risk group disjoint paths to tolerate shared-risk resource group failures. We assume that there is only at most one SRRG failure at any given time.

C. *Notations*

The following notations are used in the following sections for describing the graph transformation algorithm.

- $G = (V, E)$: Directed graph G, where V is the set of vertices and E is the set of edges. $|V| = N$ and $|E| = L$.
- $T_{N \times N}$: Traffic matrix between any two s-d pair.
- α_e: Weight of an edge e in the directed graph G. For bi-directional links, $\alpha_{uv} = \alpha_{vu}$.
- χ: Total number of Shared-Risk Link groups (SRLG's) in the network.
- R_i: The i^{th} Shared Risk Link Group. $i = 1 \cdots \chi$.
- ψ: Total number of Shared-Risk Node groups (SRNG's) in the network.
- S_i: The i^{th} Shared Risk Node Group. $i = 1 \cdots \psi$

- d_i: A dummy vertex representing the i^{th} shared risk link group or a shared risk node group.
- *N (v)*: The neighborhood of the vertex *v*.

Input: Graph $G = (V, E)$ and SRLG Groups $R_i's$.
Output: The transformed graph $G' = (V', E')$.
The following Data Structures are maintained:

Arrange all SRLG's as $R_i = \{(u_i, v_{i1}), (u_i, v_{i2}), \cdots (u_i, v_{ik_i})\}$. where $|R_i| = k_i$, i=$1, \cdots \chi$
Graph Transformation Algorithm: Obtain the transformed graph G'= (V',E') by following the edge and node transformation rules.
$V' := V \cup \{d_i\}$, $i = 1, \cdots \chi$.
Edges in E_1 are kept as it is in the original graph.
Connect edges in E_4 with initial weight of zero.
For each edge in E_2 and E_3 an alternate edge is created that includes the node on which the SRLG is incident.
$\forall e\{u_i, v_{ij}\} \in E_2$ create an edge (d_i, u_{ij}) with $\alpha_{(d_i,u_{ij})}$ = $\alpha_{(u_i,v_{ij})}$ belonging to E_5.
$\forall e\{u_i, v_{ij}\} \in E_3$ the edge will be of the type $\{u_i, u_j\}$ for some $i,j = 1, \cdots \chi$.
Create an edge (d_i, d_j) with weights $\alpha_{(d_i,d_j)}$ = $\alpha_{(u_i,u_j)}$ and $\alpha_{(d_j,u_j)}$ = *H*. (These edges are represented as E_6).
Here $H \geq max(\alpha_{u_i,u_j})$, where $u_i \in N(u_j)$.
(Make a note that u_i, u_j can be used in any order.)

Fig. 4. Graph Transformation Algorithm

III. SRLG Diverse Routing

We make the following assumptions. Under these assumptions the proposed transformation technique yields a polynomial time solution.

- There can be any number of shared-risk link groups in a network.
- Each shared risk link group is smaller in size than the degree of the node on which it is incident.
- A shared risk link group is not a proper subset of any other shared risk link group.
- An edge can be shared between atmost two shared risk link groups.

Each shared-risk link group R_i can be represented as
$R_i = \{(u_i, v_{i1}), (u_i, v_{i2}), \cdots (u_i, v_{ik_i})\}$.
where u_i is the vertex on which an SRLG is incident. $|R_i| = k_i$, i=$1, \cdots \chi$

A *dummy vertex* d_i is used to represent each shared risk link group R_i in the transformed graph *G' = (V', E')*. *G'* is derived from *G* using the vertex transformation:
$V' := V \cup \{d_i\}$, $i = 1, \cdots \chi$.

Hence in the transformed graph *G'*, the total number of vertices are given by $|V'| = N + \chi$, where *N* is the cardinality of the set *V*. This is shown in Figure 5.

Let $E = E_1 \cup E_2 \cup E_3$. The modified edges in the transformed graph *G'* are given by:
$E' = E_1 \cup E_4 \cup E_5 \cup E_6$
where $E_1 = \{e \in E \mid e \notin R_i \; \forall i\}$
$E_2 = \{e \in E \mid e \in R_i, \; e \notin R_j, \; i \neq j\}$
$E_3 = \{e \in E \mid e \in R_i, R_j \; and \; i \neq j\}$
$E_4 = \{(u_i, d_i) \mid R_i \; is \; an \; SRLG\}$

E_4 is set of edges of type (u_i, d_i) where u_i is the node on which there is an incident SRLG.

E_5 is a set of edges of type (d_j, v_j) replacing sets of edges in E_2.

E_6 is set of edges of the type (d_i, d_j) created to represent edges that are common to more than one SRLG.

Hence $|E_5| = |E_2|$ and $|E_6| = |E_3|$ and E_4 are the new set of extra edges in *G'*. The transformation algorithm is given in Figure 4. In this graph E_1 = {1-4, 1-5, 2-6, 3-4, 4-5}, E_2 = {1-2, 3-2, 6-3, 6-5} and E_3 = {1-3}.

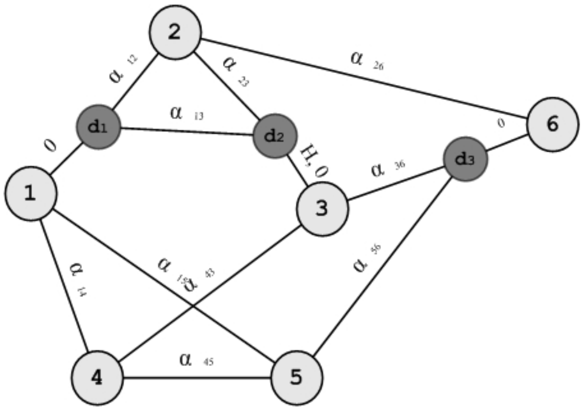


Fig. 5. Graph Transformation using Dummy Nodes for Diverse SRLG Routing.

For example for the graph shown in Figure 2(a), there are three SRLG's R_1, R_2 and R_3. The shared-risk link groups incident on node 1, 3 and 6 are R_1 : {(1,2) (1,3)}, R_2 : {(3,1) (3,2)} and R_3 : {(6,3) (6,5)}. It is to be noted that the edge (1,3) is identical to the edge (3,1), since each link is assumed to be bi-directional. The explanations of each step of the transformation are also given in Figure 4.

It is to be noted that, in instances where a link is common to two shared risk link groups, failure of any link in one group doesn't propagate to the other group. For example in Figure 2(a), failure of link $1 \longrightarrow 2$ doesn't imply failure of link $2 \longrightarrow 3$.

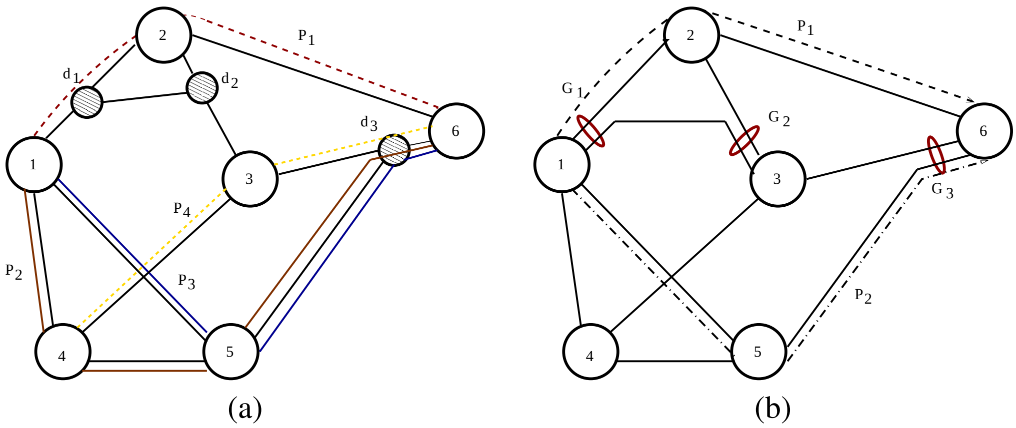


Fig. 6. (a) SRLG Disjoint Routes on transformed Graph G', (b)SRLG Disjoint Routes on the Original Graph G.

In Figure 5 the dummy nodes d_1, d_2 and d_3 are introduced to represent the three shared-risk link groups R_1, R_2 and R_3. As can be seen in Figure 5, there is a link between dummy nodes d_1 and d_2 to account for the edges in E_3. The transformed graph is shown in Figure 5. After the transformation, the edge sets are E_4 = {1-d_1, 3-d_2, 6-d_3}, E_5 = {d_1-2, d_2-2, d_3-3, d_3-5} and E_6 = {d_1-d_2}.

Input: Graph $G' = (V', E')$, and traffic matrix $T_{N\times N}$.
Output: The shortest cycle for each $t_{sd} \in T_{N\times N}$ in the transformed graph $G' = (V', E')$.
Cycle Computation Algorithm: Obtain the cycles in the transformed graph G'= (V',E') using the following algorithm:

$\forall$ ($t_{sd} \in T_{N\times N}$)
{
Obtain the Edge-Disjoint Shortest Cycle on the transformed graph *G'*.
The paths of the shortest cycle have one or more segments of the following two types:
A. $P_{G'}$: $(- \cdot\cdot x - y - \cdot\cdot\cdot -)$ or
B. $P'_{G'}$: $(- \cdot\cdot x - d_i - y - \cdot\cdot\cdot -)$ or
C. $P''_{G'}$: $(- \cdot\cdot x - d_i - d_j - \cdot\cdot\cdot - d_m - y - \cdot\cdot -)$, where $1 \le m \le \chi$.
}

Fig. 7. Cycle computation in transformed graph G'

Computing Cycles: In order to find out two shared-risk group disjoint routes between any s-d pairs, we apply the edge-disjoint shortest-cycle algorithm [21] on the transformed graph *G'* to find the two group-disjoint routes for a given s-d pair. The algorithm for the cycle computation in the transformed graph is given in Fig 7. Note that a path can have two kind of segments A or B as shown in the algorithm in Figure 7. The two group-disjoint paths between nodes 1 $\rightarrow$ 6 in the transformed graph *G'* can be possibly the two paths of either of the cycles shown in Table I.

TABLE I
DIFFERENT POSSIBLE CYCLES IN THE TRANSFORMED GRAPH *G'*

Cycle No.	Primary Lightpath	Backup Lightpath
C_1	1-d_1-2-6	1-4-3-d_3-6
C_2	1-d_1-2-6	1-4-5-d_3-6
C_3	1-d_1-2-6	1-5-d_3-6
C_4	1-d_1-d_2-2-6	1-4-5-d_3-6
C_5	1-d_1-d_2-2-6	1-4-3-d_3-6
C_6	1-d_1-d_2-2-6	1-5-d_3-6

Some of these cycles are shown in Figure 6(a). It will be shown that the two paths comprising the shortest-cycle on the transformed graph *G'* always provides us SRLG disjoint routes in the original graph *G*.

Reverse Mapping of Paths: Once we obtain the shortest-cycle on the transformed graph *G'*, we need to map the two routes on the original graph *G*. The details of the reverse mapping of the edges are presented in Figure 8. The paths can be comprised of different types of segments as shown in **Case A**, **Case B** and **Case C** in Figure 8. In **Case A**, the path is transformed back as in the original path. In **Case B**, the path segment is transformed back by dropping the dummy vertex d_i. In **Case C**, the reverse transformation follows by selecting vertices to represent pairwise dummy vertices.

As an example, if we choose the cycle C_3 in Figure 6(a) to be the shortest-cycle (assuming equal weights of all links in the original graph *G*) between nodes 1$\rightarrow$6, then the two routes, obtained from the cycle are P_1: 1-2-6 and P_2: 1-5-6 as shown in Figure 6(b).

Input: Cycles in the transformed Graph $G' = (V', E')$. The paths of the cycle are in the form $P_{G'}$, $P'_{G'}$ and $P''_{G'}$.
Output: Mapping the paths of the cycle in *G'* to the paths in the original graph *G*.
Reverse Transformation of cycles: Transformation of the paths to the original graph G.
Case A. Return the original path of the form:
P_G: $(s - \cdots - x - y - \cdots - d)$.
Case B. Replace segment of type $- \cdots x - d_i - y \cdots -$ to obtain the path:
P'_G: $(s - \cdots - x - y - \cdots - d)$
Case C. Replace segment of type $- \cdots x - d_i - d_j - \cdot - d_m - y - \cdots -$ by the path:
P''_G: $(s - \cdots - x - a - b - \cdots - c - y - \cdots - d)$
where $xa \in E(G)$, $xa \in R_i \cap R_j$.
Similarly the other dummy vertices are taken in pairs, and transformed to obtain the path P''_G.

Fig. 8. SRLG Disjoint Routing in original graph

A. Validity of Proposed Algorithms

In this subsection we present certain theorem's which validate the correctness of our proposed transformations for finding out SRLG disjoint routes.

Theorem III.1: If there exists a cycle in the graph *G*, then the transformation guarantees that the cycle will be found.

Proof: A link in a graph can either belong to one SRLG, or it can be in one of the scenarios as depicted in Figure 9. Recall that a link can belong to atmost two SRLG's. If each link in the graph, is in a single SRLG, then the transformation guarantees that the two disjoint paths between the source and destination will use any one of the two links $u - d_i - v$ or $u - d_i - w$, and hence are SRLG-disjoint. If any link belongs to more than one SRLG, than as depicted by Figure 9(a), the transformation would yield a graph *G'* as given in Figure 10. The weights on the links indicate the weights obtained after the graph transformation. Two possible cases exist:

Case I: If the paths of the cycle includes the common edge 1-2 (equivalently edge $d_1 - d_2$ in *G'*), as shown in Figure 10, then it cannot include any of the other two edges 1-3 or 2-3.

Case II: If the path happens to pass through one of the non-common edges 1-3 as shown in Figure 11, then the cycle cannot use the common edge 1-2. It is to be noted that for an s-d pair it will not use the second non-common edge 2-3, because of the higher weight on $d_j v$ or $d_i u$ if another path exists to complete the cycle. This ensures group disjointness between the paths comprising the cycle. Note that each d_i has an odd degree of $k_i + 1$ and hence can only allow only two edges to be used in a cycle and hence both the common and the not common edge cannot be used at the same time.

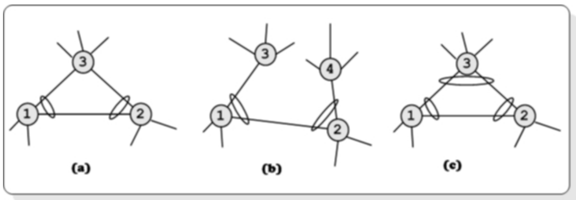


Fig. 9. Different shared link scenarios

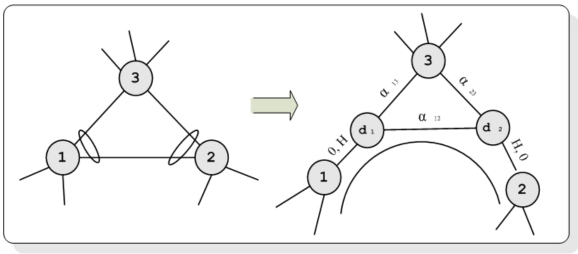


Fig. 10. Path scenarios in cycle computation in *G'*

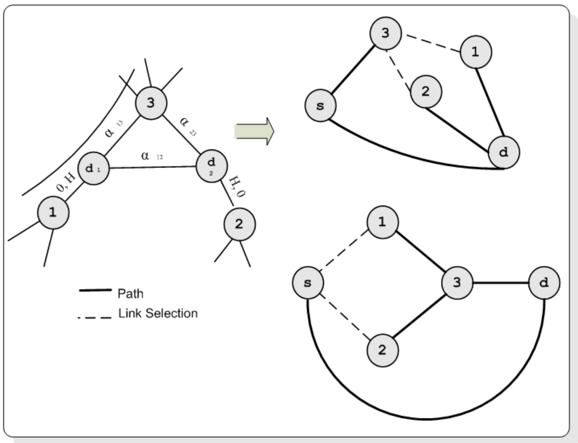


Fig. 11. Path in a cycle, including a non-common edge

In case of a network shown in Figure 9(b), the paths can be either in the format shown in Figure 12(a) & (b). Again, two possible cases may exist:

Case I: If the path of the cycle is as shown in Figure 12(a), then the cycle cannot consist of the edges 1-3 and 2-4 which is desired for ensuring group-disjoint paths.

Case II: If the path of the cycle is as shown in Figure 12(b), then the cycle can comprise of the edges 1-3 and 2-4, but cannot have the common edge 1-2, which again ensures that the paths are group disjoint and a cycle is completed.

Now, let us consider a network with SRLG's as shown in Figure 13. The transformed graph can be in any of the two cases as shown in Figure 13 (a) & (b).

Case I: If the edges $1 - d_1 - d_2 - 2$ are selected as one of the paths of the cycle, then the edges 1-3 and 2-3 cannot be part of the cycle, thus ensuring group disjointness. If the path is as shown in Figure 13(a), then the edges 1-2 and 3-2 cannot be part of the cycle.

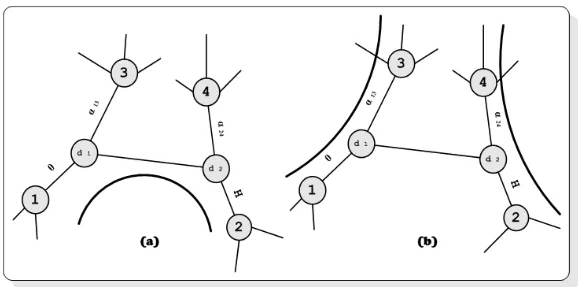


Fig. 12. Path possibilities in a cycle for network instance Figure 9(b)

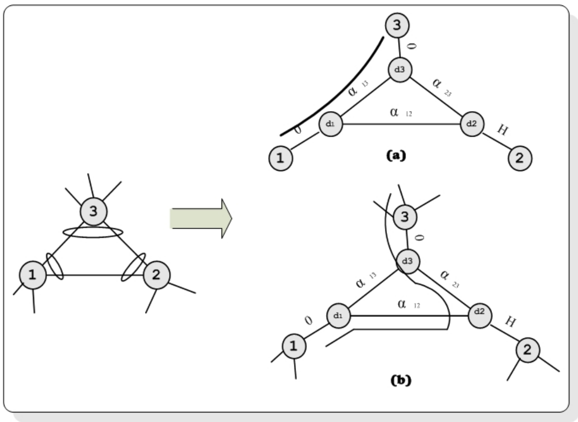


Fig. 13. SRLG scenario depicting a cycle of sharing links

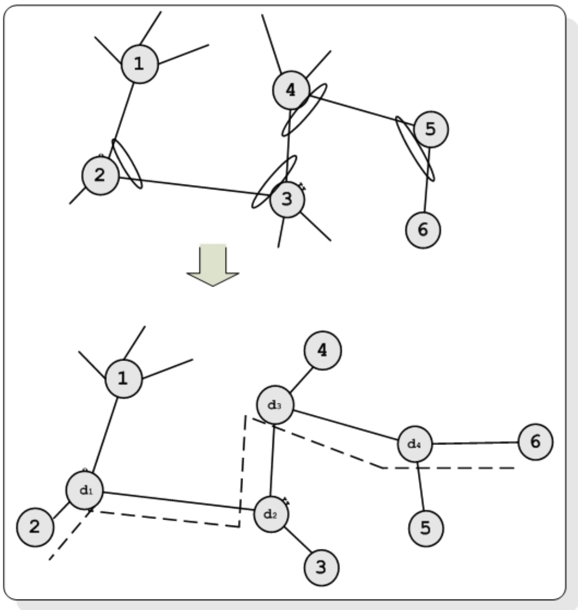


Fig. 14. Cascading SRLG's

Case II: If the path is as shown in Figure 13(b), then the edge 1-3 cannot belong to the cycle. Hence a cycle can be always computed, if one such exists in the topology.

Furthermore, we can have a network topology where the SRLG's are lined up in a cascaded manner as shown in Figure 14. The transformed network is shown in the same figure. As can be seen in the worst-case, the path would traverse through all the dummy nodes. The following two cases exist:

Case I: If the edges $2 - d_1 - 1$ is selected as one of the paths of the cycle, then the edge 2-3 cannot be part of the same cycle, thus ensuring group disjointness.

Case II: If the path is of the type $u - d_i - d_j - \cdot - \cdot - d_m$, where $1 \leq m \leq \chi$, then the path in the original graph can be obtained by reverse transforming the dummy vertices pairwise to edges in *G*, as shown in Figure 8. Again this guarantees that a cycle can be found. This completes the proof of Theorem III.1. ■

Theorem III.2: If there exists an edge-disjoint shortest cycle in the transformed graph *G'*, then using the above transformation, the paths P_1 and P_2 comprising the cycle can be always mapped to two SRLG disjoint routes in the original graph *G*.

Proof: If P_1 and P_2 comprises the two paths of the edge-disjoint shortest cycle in the transformed graph *G'*, then each dummy vertex d_i can appear atmost once in either of the paths P_1 or P_2.

Thus the paths P_1 and P_2 consists of segments which maybe either in the form of *P*: $(- \cdots - u - d_i - v - \cdots -)$ or *P'*: $(- \cdots - u - d_i - d_j - \cdots - d_m - v - \cdots -)$, where $1 \leq m \leq \chi$. In case 1 (also shown in Figure 11), there is an exact mapping of the edges $u - d_i - v$ in the transformed graph *G'* to the edge *uv* in the original graph *G*. This can be obtained by just dropping the dummy vertex from the path obtained in the transformed graph *G'* to obtain the path in the original graph. In case 2 (as shown in Figure 14), the edges $d_i - d_j - \cdots - d_m$ in the transformed graph, is mapped back to the edges in the original graph by taking the dummy vertices pairwise, and transforming it to the edges *uv*, such that *uv*$\in$ *E(G)*, *uv* $\in R_i \cap R_j$ and ud_i, $d_jv \in$ *E'(G')*. Since the primary and backup lightpaths do not have any overlapping d_i's, thus the routes mapped back on the original graph are always SRLG disjoint. This completes the proof of Theorem III.2. ■

B. Complexity Analysis of SRLG Routing

We evaluate the overall complexity of the SRLG disjoint routing algorithm using our technique. The computational complexity can be broken up into three parts, one the complexity involved in the transformation of the graph, second the complexity of finding out edge-disjoint shortest cycles in the transformed graph, and finally the complexity of mapping the paths obtained on the transformed graph to the paths in the original graph.

The complexity involved in determining whether an edge belongs to more than one shared-risk link group, is computationally of the order of O(*L*). One way to solve this is to scan the shared risk groups in order and a data structure be maintained against each link to determine the number of groups it belongs to. Depending on the outcome of this decision, the edges are transformed following the algorithm presented above. Thus the overall complexity of the graph transformation is O(*L*).

The transformation of the graph from *G* $\rightarrow$ *G'* adds χ number of nodes. Two paths that are shared-risk group disjoint are computed using the shortest-cycle algorithm on the transformed graph as described in [21]. The computational complexity of the shortest-cycle algorithm is given by O(L^2). We have two distinct cases, one in which the SRLG's are such that there is no edge which belongs to more than one group and another scenario where an edge can possibly belong to two shared risk groups. In both the cases, the additional number of edges introduced in the transformed graph *G'* is χ. Hence the complexity of finding two SRLG disjoint paths would be O(L^2).

Once the edge-disjoint shortest cycle is computed in the transformed graph, the paths comprising the cycle, is mapped back on the original graph *G*. This has a computational complexity of O(*L*) since a path can have a maximum of (*L* + χ) edges in a transformed graph *G'* and determining the next hop vertex would involve a search among all neighboring edges. Combining the three above complexities, the overall complexity of the diverse SRLG routing is given by the dominant term, which is the complexity involved in finding the edge-disjoint shortest cycle on the transformed graph. Thus the overall complexity is given by O(L^2).

IV. SRNG (Shared Risk Node Group) Diverse Routing

In this section we consider shared-risk node groups. We develop a transformation technique for finding out two routes in scenarios where more than one node shares a common risk of failure. We identify such scenarios where more than one node shares a common risk of failure as shared risk node groups (SRNG's). We assume that each node can be part of at most one SRNG and we also assume that the size of each SRNG is limited to two adjacent nodes sharing an edge between them.

Each shared-risk node group S_i can be represented as

$S_i = \{(u_i, v_i) \mid u_i \in N(v_i)\}$.
where $|S_i| = 2$, $i = 1, \cdots \psi$

A *dummy vertex* d_i is used to represent each shared-risk node group in the transformed graph *G' = (V', E')*. *G'* is derived from *G* by following the vertex transformation:

$V' := V \setminus \{u_i, v_i\} \cup \{d_i\}$, $i = 1, \cdots \psi$.

Hence in the transformed graph *G'*, the total number of vertices are given by the cardinality of the set *V'*, $|V'| = N - \psi$, where *N* is the cardinality of the set *V*. The edges in the transformed graph *G'* are represented as:

$E' = E \setminus \{E_1 \cup E_2 \cup E_3\} \cup E_4$
where $E_1 = \{u_i v_i\}$, $i = 1, \cdots \psi$. $|E_1| = \psi$.
$E_2 = \{u_i x\}$, $u_i \in S_i$, $x \in N(u_i)$, $x \notin S_i$, $i = 1, \cdots \psi$
$E_3 = \{v_i x\}$, $v_i \in S_i$, $x \in N(v_i)$, $x \notin S_i$, $i = 1, \cdots \psi$
E_4 are new edges from d_i to $N(u_i) \cup N(v_i)$, $i = 1, \cdots \psi$.

The graph transformation algorithm is presented in Figure 15. All the edges belonging to the sets E_1, E_2 and E_3 are deleted and new edges are introduced between the neighborhood of u_i and v_i and d_i.

Input: Graph G = (V,E) and SRNG Groups S_i's.
Output: The transformed graph $G' = (V', E')$.
Graph transformation algorithm:
$S_i = \{(u_i, v_i) \mid u_i \in N(v_i)\}$.
where $|S_i| = 2$, $i = 1, \cdots \psi$

Obtain the transformed graph *G'=(V',E')* by the following transformations:
$V' := V \setminus \{u_i, v_i\} \cup \{d_i\}$, $i = 1, \cdots \psi$.
Delete all E_1, E_2 and E_3 edges.
Introduce new edges E_4. The new edges for all the shared node risk groups $i = 1, \cdots \psi$ assume the weights:

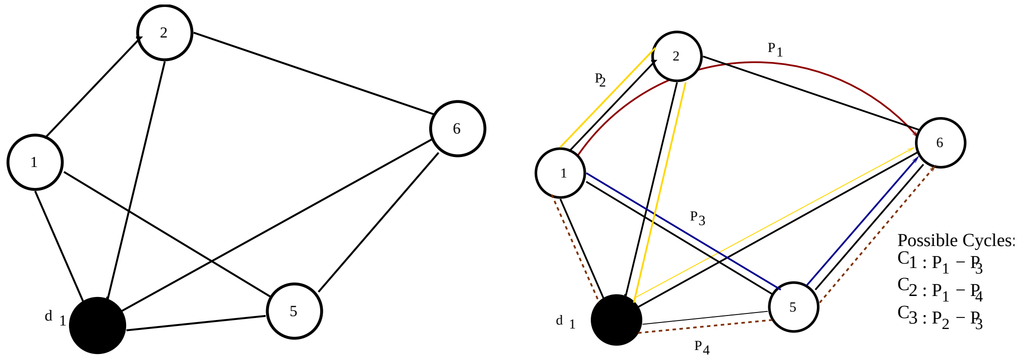
$$\alpha_{d_i x} = \begin{cases} \alpha_{u_i x} & if\ x \in N(u_i) \setminus N(v_i) \\ \alpha_{v_i x} & if\ x \in N(v_i) \setminus N(u_i) \\ min(\alpha_{u_i x},\ \alpha_{v_i x}) & if\ x \in N(v_i), N(u_i) \end{cases}$$

Fig. 15. Graph transformation algorithm

We demonstrate the use of this algorithm in Figure 16(a). The this figure, the links 2→3, 1→3, 1→4, 6→3 and 5→4 are replaced by modified edges with adjusted edge weights as shown in the graph transformation algorithm.

Computing Cycles: Two routes for a given s-d pair in the original graph *G*, that are node group disjoint are obtained by finding the node-disjoint shortest-cycle [21] in *G'* as shown in Figure 16(b) and Figure 17. Infact finding out a node disjoint shortest cycle on the transformed graph ensures that the two paths comprising the cycle, can guarantee fault-tolerance against failure of any single node on the path and also from node groups present in the topology. For example the two node disjoint paths between nodes $1 \rightarrow 6$ can be the two paths comprising either of the following cycles in *G'*: C'_1: {1-2-6, 1-5-6} or C'_2: {1-2-6, 1-d_1-6} or C'_3: {1-2-6, 1-d_1-5-6} and C'_4: {1-2-d_1-6, 1-5-6} as shown in Figure 16(b).

Reverse Mapping of Paths After the node-disjoint shortest cycles are found, the paths corresponding to the cycles are mapped back to obtain the two node-group disjoint routes on the original graph *G*. The fact that the routes in *G'* are node-disjoint guarantees node group disjoint routes in the original graph *G*.



(a) (b)

Fig. 16. (a) Transformed Graph Indicating Shared Risk Node Groups,(b) SRNG Disjoint Routes in the transformed Graph G'.

The details of the reverse mapping of the edges are presented in Figure 18. The paths can be comprised of different types of segments as shown in **Case A**, **Case B** and **Case C** in Figure 18. In **Case A**, the path is returned identical to the input path. In **Case B**, the path segment is transformed back by selecting a vertex *w* which minimizes α_{xw} + α_{wy}. In **Case C**, as illustrated in Figure 19. The reverse transformation of edges follows by selecting vertices that minimizes the total cost of the segment between *x* and *y*.

Input: Graph G' = (V',E') and traffic matrix $T_{N\times N}$.
Output: The shortest cycle for each $t_{sd} \in T_{N\times N}$ in the transformed graph $G' = (V', E')$.
Cycle computation algorithm: Obtain the cycles in the transformed graph G' using the following algorithm

$\forall$ ($t_{sd}\in T_{N\times N}$)
{
Obtain the Node-Disjoint Shortest Cycle on the transformed graph G' for any traffic $t_{sd} \in T_{N\times N}$ using the node-disjoint shortest cycle algorithm presented in [21]. The paths of the shortest cycle have one or more segments of the following types:
A. $P_{G'}$: $(- \cdots - x - y - \cdots -)$
B. $P'_{G'}$: ($-\cdots$-x-d_i-y-$\cdots-$) or
C. $P''_{G'}$: ($-\cdots$ x-d_i-d_j-$\cdots$-d_m-y-$\cdots -$), where 1 $\leq$ m $\leq$ ψ.
}

Fig. 17. Cycle computation algorithm

If the cycle C'_3 is chosen, then the paths on the original graph would be 1-2-6 and 1-4-5-6 (and not 1-3-6 because, there is no link between 3$\rightarrow$5). However if the cycle C'_4 is chosen, then the two routes are 1-2-3-6 and 1-5-6. This is shown in Figure 20.

Theorem IV.1: In a graph with SRNG's, if there exists a node-disjoint shortest cycle in the transformed graph *G'*, then using the above reverse mapping, the paths P_1 and P_2 composing the cycle always maps to two SRNG disjoint routes in the original graph *G*.

Proof: Each *dummy node* representing a shared risk node group can appear atmost once in any one of the two routes P_1 or P_2 of a cycle, and these routes are node-disjoint. The transformation of these routes on the original graph *G* hence guarantees that they are SRNG disjoint. ■

Input: Cycles in the transformed graph *G' = (V',E')*
Output: Mapping the paths of the cycle in *G'* to the paths in the original graph *G*.
Reverse transformation of cycles: Transformation of cycles computed in *G'* to original paths in *G*.

Case A. P_G: The path is returned identical to the input path and is of the format: $(- \cdots - x - y - \cdots -)$.
Case B. P'_G: At vertices *x* & *y* select vertex *w* s.t. $w \in S_i$, *N(x)*,*N(y)*. For more than one choice of *w*, select *w* s.t. it minimizes α_{xw} + α_{wy}. The segment transformed back in the original graph, hence becomes $(- \cdots - x - w - y - \cdots -)$.

If no unique *w* exists, determine if $u_i \in$ *N(x)*, $u_i \notin$ *N(y)*, $u_i \in S_i$. The segment transformed back in the original graph, hence becomes $(- \cdots - x - u_i - v_i - y - \cdots -)$. Else the segment transforms to $(- \cdots - x - v_i - u_i - y - \cdots -)$.
Case C. Let *e'*, *e"* denote the critical edges connecting the vertices in the node-groups S_i and S_j respectively. Let *e* denote the bridge link between S_i and S_j.
P''_G: The reverse mapping of the segment *x*-d_i-d_j- leads to a selection of vertices *w*, *w'* such that $w \in$ *N(x)*, S_i, and *w'* $\in$ *N(w)*, S_j, and corresponding edges which satisfy that one of α_{xw} + α_e or α_{xw} + $\alpha_{e'}$ + α_e or α_{xw} + α_e + $\alpha_{e''}$ or α_{xw} + $\alpha_{e'}$ + α_e + $\alpha_{e''}$ is *minimized*, depending on choice of vertices. This is done for all successive pair of $d_i - d_j$ vertices.

Fig. 18. Reverse mapping of paths algorithm

A. *Complexity Analysis of SRNG Routing*

This section evaluates the overall complexity of SRNG disjoint routing. The computational complexity can be broken up into three parts, the complexity involved in the graph transformation, second the complexity of finding out node-disjoint cycles in the transformed graph and finally mapping the paths obtained on the transformed graph to paths in the original graph.

The total number of shared risk node groups in the network is given by ψ and each group has two nodes. Thus the complexity involved in determining whether an edge has an end-vertex which belongs to an SRNG and transforming it involves an exhaustive search, which is computationally of the order of O($L \cdot \psi$). Depending on the outcome of this decision, the edges are transformed following the algorithms explained above. Thus the complexity of the graph transformation is O($L \cdot \psi$).

The transformation of the original graph *G* into the final graph *G'* reduces ψ number of nodes in the original graph. Two paths that are shared-risk group disjoint are computed using the node-disjoint shortest-cycle algorithm on the transformed graph [21]. The computational complexity of the shortest-cycle algorithm is given by O(L^2). Following the graph transformation all the edges between the nodes of each node group gets deleted, hence leading to a deletion of total ψ number of edges. Let us denote the set of vertices which has an edge to both the nodes of an SRNG as the vertex set V_s. Let the cardinality of this set be denoted by $|V_s|$. Transition from $G \rightarrow G'$ leads to a further deletion of $|V_s|$ edges. Hence the complexity of finding two SRNG disjoint paths is O(L^2).

Once the node-disjoint shortest cycles are computed on the transformed graph, the paths comprising the cycle, are mapped back to the original graph *G*. In cases where nodes belong to only one shared risk node group, we scan through each edge in the path and at the vertices *x* & *y*, we scan through its entire

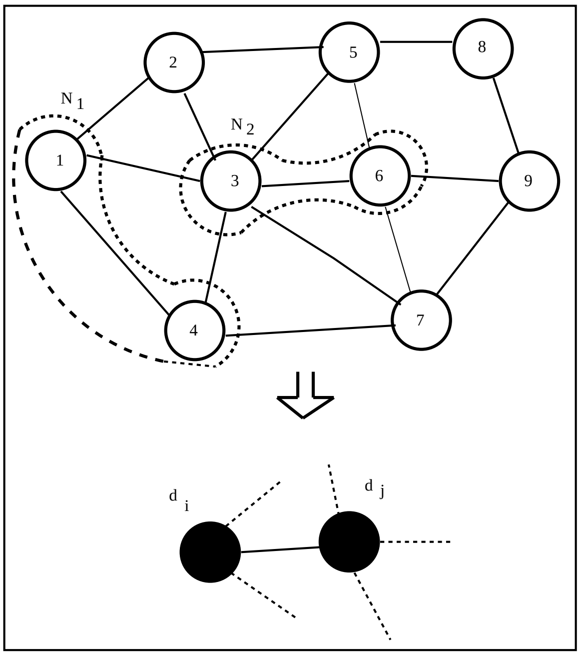


Fig. 19. Example showing links between two dummy nodes and the reverse transformation.

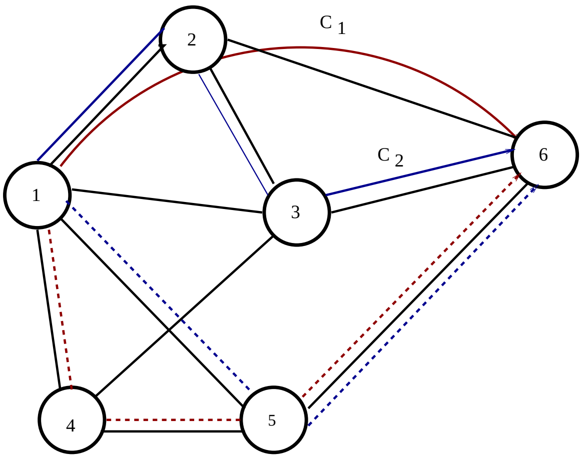


Fig. 20. Shared Risk Node Group Disjoint Routes on the Original Graph G.

neighborhood and select the appropriate vertices as described in Section 6. Hence the complexity involved in this procedure is O(L), since the maximum hop length of a path can be restricted to *E'* edges and we need to scan through all the neighbors and compare the path lengths.

Combining the three above complexities, the overall complexity of the diverse SRNG routing is given by the dominant term, which is the complexity involved in finding the node-disjoint shortest cycle on the transformed graph. Thus the overall complexity is given by O(L^2).

V. Minimizing Capacity in Dynamic Routing

When a connection is routed on an optical network, one of the primary objectives is to not only route it over diverse paths, but also minimize the overall spare capacity utilized in routing the incoming request. One of the traditional approaches involves sharing of backup bandwidth between multiple connections, that do not share risk of failure at the same instance of time.

The network is modeled as having E links, F fibers per link, and W wavelengths per fiber. Hence the network can be represented by a 3-tuple (e, f, w), where $1 \leq e \leq E$, $1 \leq f \leq F$, and $1 \leq w \leq W$. The multiplexing of backup bandwidth on a particular fiber-wavelength combination is done by maintaining a *Backup Request List (BRL)* on each fiber-wavelength at every link. The Backup request List is a link-list of all the requests whose backup paths are multiplexed on a particular fiber-wavelength at a link.

The primary path of the requests are routed using the *first-fit* fiber & wavelength assignment. To ensure multiplexing of backup bandwidth, a Backup Request List (BRL) is maintained on each wavelength on each fiber at every link. Hence the BRL on a link is denoted by $\{e_i, f_j, (w_1, w_2, \cdot\cdot\cdot\cdot, w_W)\ \}$, where e_i denotes the link or edge id, f_j denotes the fiber id and each of w_i denotes the BRL on that fiber and wavelength. When an incoming request R_i arrives, the SRRG disjoint paths are computed using the graph transformation technique.

A check is made on its backup path on each fiber and wavelength to see if there is any conflict with the existing requests in the BRL. After the check a vector *Backup Availability Matrix* $B_{l_1}^{f_1} : \{m_1, m_2 \cdot \cdot \cdot m_W\}$ is computed on each fiber and wavelength on the links of the backup path of the request. The Backup Availability Matrix is filled up with 0 if there are no free backup possible at that fiber-wavelength combination or if that wavelength on that fiber is used up by a primary connection. It is filled up with a 1 if the incoming request has no conflict with any of the requests on the BRL. Hence the information stored on the Backup Availability Matrix helps us in determining whether the incoming request can share backup bandwidth on any fiber-wavelength combination.

A vector called *Routing-Info Matrix* is computed by combining the Backup Availability Matrix vector on all links along the path of the backup request. The *Routing-Info Matrix* is computed by the operation $RI_{f_i} = B_{l_1}^{f_i} \otimes B_{l_2}^{f_i} \cdot\cdot \otimes B_{l_k}^{f_1}$, $1 \leq i \leq F$ and k is the number of hops in the path. The operator $\otimes$ simply denotes a binary **OR** operation. Once the *Routing-Info Matrix* is computed, the requests are assigned channels (wavelengths) by scanning through the *Routing-Info Matrix* and using the *first-fit* wavelength assignment algorithm. The above channel establishment procedure ensures minimum fiber usage and multiplexing of backup bandwidth while routing an incoming connection in the network [19][20]. This methodology can be applied in general for any heterogeneous wavelength routed network.

VI. Performance Results

Three network topologies, a 14 node, 23 link NSFNET, 11 node, 22 link NJLATA and 15 node, and 24 link modified COST239 network as shown in Fig. 21 were used to assess the performance of our algorithm for tolerating SRRG failures. Each of these topologies consists of links with 1 fiber per link and 20 wavelengths per fiber. Each link also consists of 2 unidirectional links that are assumed to be part of the same shared risk link group, meaning that if the link in one direction fails, the link in the opposite direction also fails because they would presumably physically routed together. No nodes offer any wavelength switching capabilities, thus the *wavelength continuity constraint* is obeyed.

The *MICRON*- Methodology for information Collection and Routing in Optical Networks [19] framework for information collection and routing is used as the methodology for channel allocation for any incoming request.

The arrival of the requests at a node follow a Poisson process with rate λ, and are equally likely to be destined to any other node. The holding time of the requests follow an exponential distribution with unit mean. The capacity requirements of each request is a unit wavelength. In Fig. 23(a), we depict the blocking probability for all the three topologies using the proposed graph transformation technique and the shortest-path diverse routing technique presented in [5][21].

The results indicate that the graph transformation techniques achieves better blocking performance as

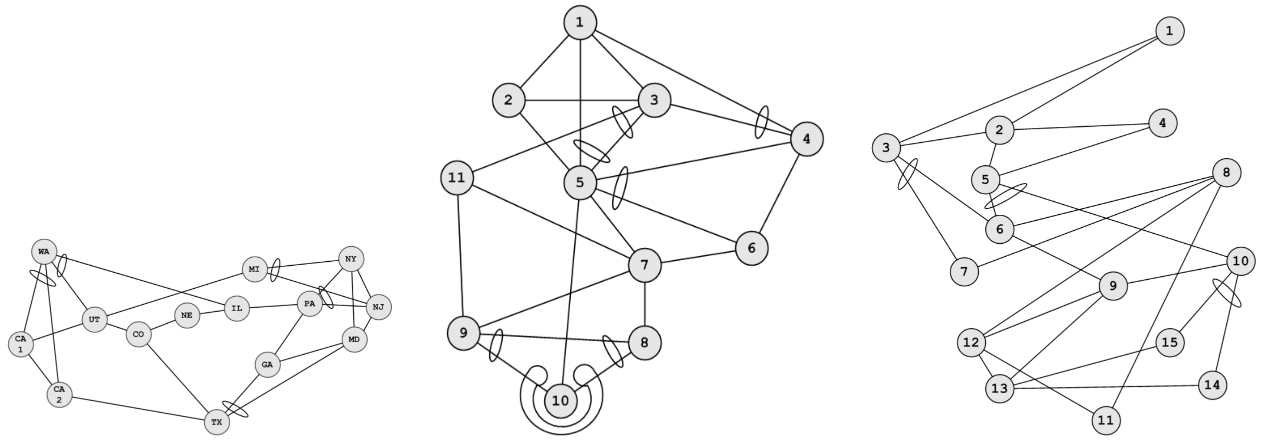


Fig. 21. (a) 14 node, 23 link NSFNET; (b) 11 node, 22 link NJLATA; (c) 15 node, 24 link modified COST239 network

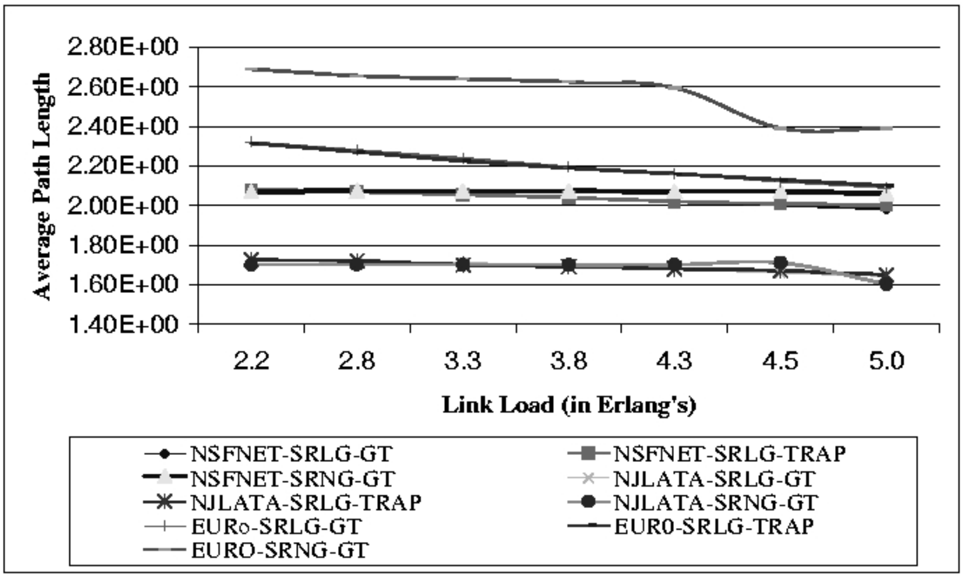


Fig. 22. Average Path length vs. Link Load (in Erlang's)

compared to the techniques studied in [21]. The trend is similar in all topologies, however the maximum benefits are achieved in NJLATA. This is probably because it has a high average nodal degree of 4.0 as compared to 3.28 in NSFNET and 3.2 in case of EURO-net. High nodal degree helps in the computation of diverse routes for an incoming connection.

Fig. 23(b) demonstrates the blocking performance for all the three studied topologies using our algorithm and the techniques studied in [5][21] in the presence of shared-risk node groups (SRNG's). The blocking values computed are significantly higher for shared-risk node groups as compared to those for shared-risk link groups. The graph transformation techniques are able to provide diverse routes for around 20-30% more incoming requests, as compared to the approaches proposed in [21] which are unable to provide any diverse routes for the studied shared-risk node groups.

Fig. 22 demonstrates the average path length in the presence of shared-risk link groups and shared-risk node groups using both the graph transformation technique as well as the shortest-node pair disjoint routing technique. We observe that with increasing link load, and increasing blocking probability, the

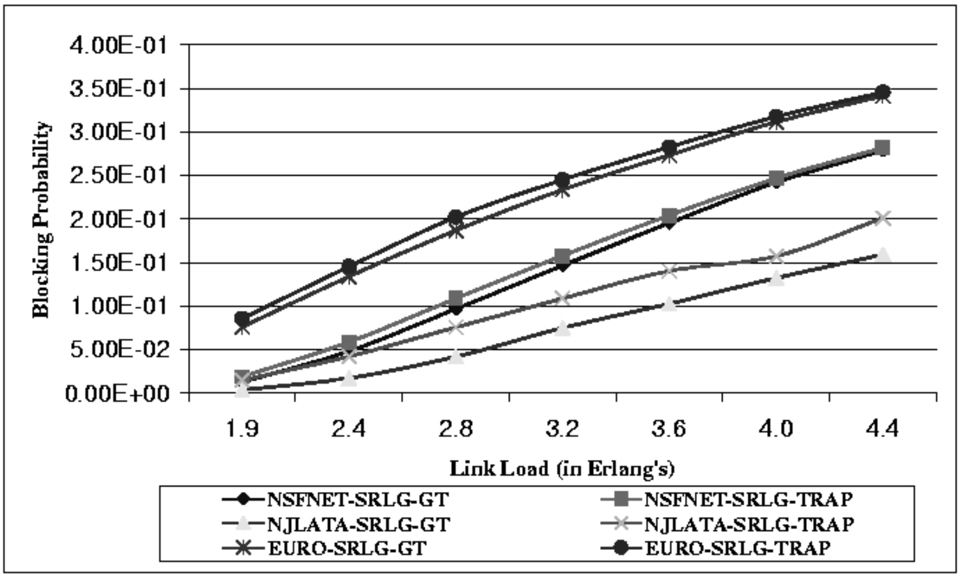


Fig. 23. Blocking Probability vs. Link Load (in Erlang's) for SRLG's

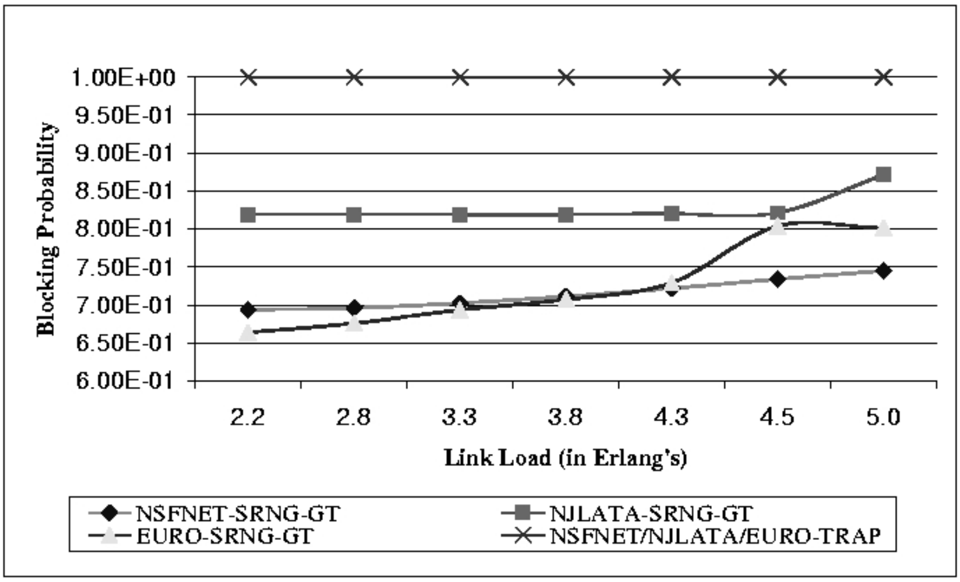


Fig. 24. Blocking Probability vs. Link Load (in Erlang's) for SRNG's

average path length decreases for all topologies.

The effective network utilization in the presence of shared-risk link groups (SRLG's) and shared-risk node groups (SRNG's) are represented in Fig. 25. As can be observed in these figures, the effective utilization of the network increases with increasing load. This is because, as the network load increases, there are more number of connections that are serviced by the network. As can be observed, the effective utilization varies between 18% -32% for both the graph transformation technique as well as the shortest-path heuristic in case of shared-risk link groups. It varies between 5% -14% for shared-risk node groups across different topologies using the graph transformation technique. The effective utilization is zero for shared node risk groups, using the shortest-path heuristic.

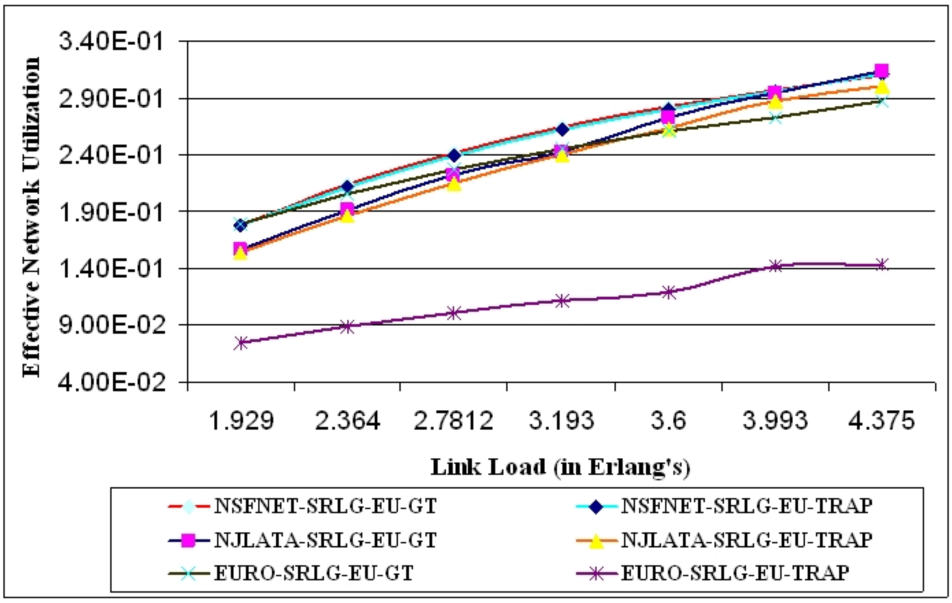


Fig. 25. Effective utilization vs. Link load for Shared-Risk Link Groups (SRLG's)

VII. Conclusion

In this paper we proposed graph transformation techniques to solve the diverse routing problem in networks with shared risk resource groups. We proposed a methodology for tolerating dependent or shared risk link failures and coordinated node failures in a network, by creating different graph transformations, routing on the transformed graph and transforming the routes on the modified graph to the original graph. One of the elegant features of the proposed strategy is that it can identify a cycle if one such exists in the topology in the presence of SRLG or SRNG groups.

The proposed transformation algorithm needs addition of a small number of edges and vertices to the original graph, and computation of link-disjoint or node-disjoint shortest cycles in the transformed graph. It provides a polynomial time solution for shared resource groups incident on a common node or between two adjacent nodes, which is an interesting result since the generalized diverse routing problem in presence of risk-groups has been proved to be NP-Complete.

We validate the correctness of our approach, and show that the transformation technique always guarantees to yield shared risk group disjoint routes, if such a route exists in the graph. Our proposed technique is capable of improving the diverse route computation by around 20-30% as compared to traditional approaches proposed in [5]. This approach for diverse routing under multiple failure scenarios is elegant and can be applied to large networks with huge traffic demands. As part of our future work, we plan to extend this graph transformation technique for accommodating groups of link failures that are not incident on a common node.

References

[1] S. Ramamurthy and B. Mukherjee, "Survivable WDM Mesh Networks, Part 1: Protection," *IEEE INFOCOM 2*, pp.744 -751, 1999.

[2] O. Gerstel and R. Ramaswami, "Optical layer survivability-an implementation perspective," *IEEE Journal on Selected Areas in Communications*, vol. 18, no.10, pp. 1885–1889, Oct 2000.

[3] R. Doverspike and B. Wilson, "Comparison of capacity efficiency of DCS network restoration routing techniques," *Journal of Network and System Management*, vol. 2, no.2, pp. 95 -123, 1994.

[4] J. Doucette and W. D. Grover, "Capacity design studies of span-restorable mesh transport networks with shared-risk link group (SRLG) effects," *SPIE Optical Networking and Communications Conference (Opticomm 2002)*, Boston, MA, July-Aug 2002.

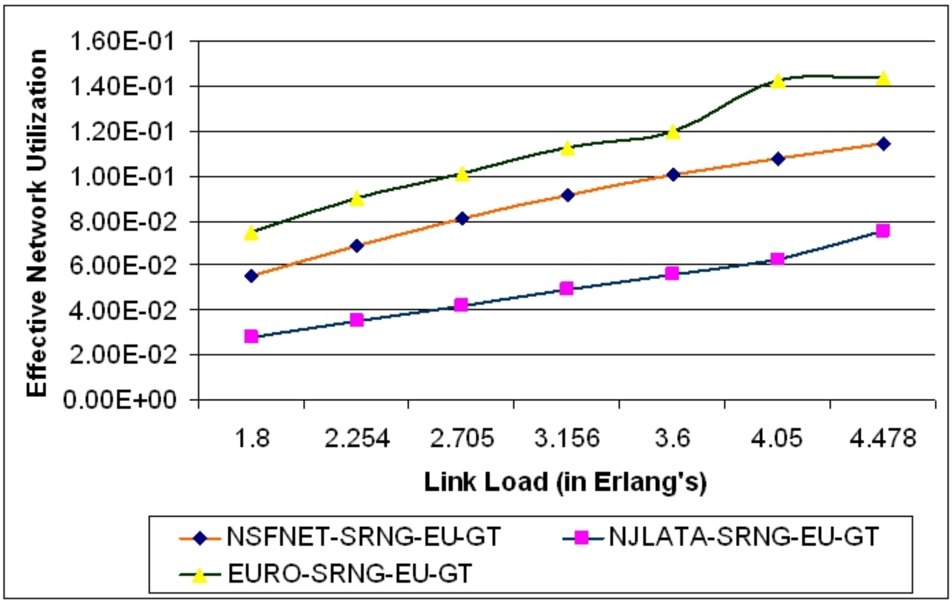


Fig. 26. Effective utilization vs. Link load for Shared-Risk Node Groups (SRNG's)

[5] G .Z. Li, B. Doverspike and C. Kalmanek, "Fiber span failure protection in mesh optical networks", *SPIE Optical Networking and Communications Conference (Opticomm)*, volume 4599, pages 130-142, 2001.
[6] Reinhard Diestel, "Graph Theory," *Springer-Verlag, New York Graduate Texts in Mathematics*, second edition, February 2000.
[7] D. Papadimitriou et.al, "Interference of shared risk link groups," *Internet Draft, OIF Contribution OIF 2001-066*, July 2001.
[8] E. Bouillet, J. -F. Labourdette, G. Ellinas, R. Ramamurthy, S. Chaudhuri, "Stochastic approaches to compute shared mesh restored lightpaths in optical network architectures," *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE Proceedings*, volume 2 , pages 801 - 807, June 2002.
[9] J. Q. Hu, "Diverse routing in mesh optical networks," *IEEE Transactions of Networking*, To appear.
[10] G. Li, B. Doverspike and C. Kalmanek, "Fiber span failure protection in mesh optical networks," *SPIE Optical Networking and Communications Conference (Opticomm 2001)*, 2001.
[11] K. Lee and K. Siu,"An algorithmic framework for protection switching WDM networks," *Proceeding of NFOEC'01*, pp.402-410, Baltimore, MD, July 2001.
[12] D.A. Dunn, W.D. Grover, M.H. MacGregor, "A comparison of k-shortest paths and maximum flow methods for network facility restoration," *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 1, pp. 88-99, Jan 1994.
[13] K. Kar, M.S. Kodialam, T.V. Lakshman, "Routing restorable bandwidth guaranteed connections using maximum 2-route flows," *IEEE/ACM Transactions of Networking*, 11(5), pp.772-781, 2003.
[14] W.D. Gover, "Mesh-based Survivable Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking," *Prentice Hall PTR*, Upper Saddle River, New Jersey, 2003.
[15] E. Modiano and A. Narula-Tam, "Survivable lightpath routing: a new approach to the design of WDM-based networks," *IEEE Journal of Selected Areas in Communication*, May 2002.
[16] G. Ellinas *et al.*, "Routing and restoration architectures in mesh optical networks," *Optical Networks Magazine*, Jan 2003.
[17] M. Garey and D. Johnson, "Computers and Intractability: A Guide to the theory of NP-completeness," , 1979.
[18] D. Xu, Y. Xiong and C. Qiao, "Protection with multi-segments (PROMISE) in Networks with Shared Risk Link Groups (SRLG's)," *IEEE/ACM Transactions on Networking*, vol.11, no.2, pp.248-258, April-2003.
[19] R. Srinivasan, "MICRON: A framework for connection establishment in optical networks," *in Proceedings of OPTICOMM*, pp. 139-150, Dallas, October 2003.
[20] P. Datta, M. Sridharan and A. K. Somani, "A Simulated Annealing Approach for Topology Planning and Evolution of Mesh-Restorable Optical Networks," *7th IFIP Conference on Optical Networks Design and Modeling*, vol.1, pp.23 -40, Budapest, Hungary 2003.
[21] R. Bhandari, "Survivable Networks: Algorithms for Diverse Routing," *Kluwer Academic Publishers*, 1999.
[22] J. Suurballe and R. Tarjan, "A quick method for finding shortest pair of disjoint paths," *Networks*, vol.14, pp.325-336, 1984.