# Design Algorithms for Fast Network Restoration via Diversity Coding

Serhat Nazim Avci and Ender Ayanoglu
Center for Pervasive Communications and Computing
Department of Electrical Engineering and Computer Science
University of California, Irvine
Irvine, CA 92697-2625

***Abstract*—The technique of diversity coding was introduced to provide fast restoration against link failures in communication networks. We have recently devised algorithms to incorporate diversity coding into networks with arbitrary topology, so that restoration is achieved very fast, in an almost hitless manner, while spare capacity requirements are competitive with the alternative state-of-the-art restoration techniques. In addition to two techniques we have recently developed, in this paper we introduce two new techniques. These two techniques provide optimal pre-provisioning of aggregate traffic and active provisioning of dynamic traffic, respectively. In both cases, diversity coding results in smaller restoration time, simpler synchronization, and much reduced signaling complexity than the existing techniques in the literature. For pre-provisioning, a Mixed Integer Programming (MIP) formulation is developed. For dynamic provisioning, an algorithm employing Integer Linear Programming (ILP) searches for the best coding group to add the new connection demand in an optimum manner.**

## I. Introduction

In communication networks, especially in wide area networks carrying telephone and Internet traffic, failures occur commonly [1]. Among all failures, single link failures are the most common, consisting of 70% of all the failures [2]. There are a number of techniques proposed for the purpose of recovery, or restoration, after such failures such as special rings, mesh restoration, or the $p$-cycle technique [1], [3], [4]. Two criteria to measure the effectiveness of such techniques are the restoration time and the required spare capacity, measured typically in terms of the product of the transmission rate and the distance. These two measures are typically complementary and the network designer has a set of different alternatives to choose from. For the telephone network, the industry goal is to achieve a restoration time less than 50 ms, according to considerations of human perception. For mission critical Internet traffic, reduction of the restoration time to much smaller values is desirable, because of the additional complexities introduced by different layers of Internet protocols.

The so-called 1:1 and 1+1 Automatic Protection Switching (APS) techniques employ spare capacity, inactive and active respectively, to protect against failure for a given connection [1]. The spare is chosen to be link disjoint with the protected connection. Although these could be construed to be early versions of restoration techniques, they are not practical due to the requirment of an enormous extra capacity if used to protect all the connections in a network.

This work is partially supported by NSF under Grant No. 0917176.

In the 1990s, when high-speed optical transmission standards for long-haul networks were being developed, restoration against link failures was taken into consideration and incorporated into the standard. This standard is known as Synchronous Optical Network (SONET). Because it was designed mainly for the telephone network, the restoration time target was taken as 50 ms. By employing redundant fibers in a ring structure, SONET switches to extra capacity in the case of a failure. Different versions of deployment with different configurations exist but in all of them, the result is more than 100% extra capacity, measured in terms of the product of the extra fibers and the distance, or fiber miles.

A number of techniques provide extra capacity in a shared fashion without imposing an overlay topology in mesh networks. Because of the shared nature of extra capacity, these approaches can reduce spare capacity, but trading off the restoration time.Some mesh-based techniques try to employ "hot standby" [5] or "pre cross-connected" [6] configurations. These approaches reduce the reconfiguration time in switches without sacrificing capacity, but the delay and complexity due to signaling still remains.

A technique that combines the speed advantage of SONET rings and the spare capacity of mesh restoration is known as the $p$-cycle protection [7]. Depending on network topology, it can provide advantages against mesh restoration.

All of the restoration techniques discussed so far, with the exception of 1+1 APS, employ feedback signaling and rerouting, which are the causes of large restoration times. As discussed earlier, 1+1 APS has an enormous capacity requirement. On the other hand, it is possible to employ channel coding on extra capacity, thereby providing a sharing of extra capacity among potential failures and at the same time, removing the need for feedback signaling, as in 1+1 APS. As a result, both the restoration time and spare capacity reduction goals can be achieved, albeit within certain limits. This idea was introduced in [8], [9] and is called *diversity coding*. In a simple scenario, it can be considered as recovering any of the failed data in $N$ link-disjoint primary paths by means of the coded data in a link-disjoint parity link. The erasure data on the single parity link is formed by applying exclusive or (XOR) operation over the data signals in $N$ primary paths. In [3], [10], diversity coding was applied to arbitrary network topologies, using a heuristic algorithm, by the authors of this paper. The results indicate that diversity coding is much faster than a typical mesh restoration technique known as source

rerouting, and the $p$-cycle technique. When the destination node for the primary paths is common, diversity coding can provide near-hitless restoration.

In [11], we introduced a technique to convert a mesh restoration technique with Shared Path Protection [12] into one employing diversity coding. This technique is called Coded Path Protection (CPP) while we abbreviate the Shared Path Protection technique as SPP. CPP provides, in addition to coding inside the network, decoding inside the network as has been sought for within the context of network coding. In terms of the analogy given in [13], CPP provides, in addition to the "poison," also the "antidote" wherever appropriate. This approach reduces the restoration time considerably but may slighlty increase spare capacity requirements as compared to SPP.

In this paper, we introduce two techniques that employ diversity coding. The first is an optimum design algorithm employing Mixed Integer Programming (MIP) for a given network topology and a corresponding traffic matrix. We call this approach pre-provisioning. Although based on MIP, this algorithm is faster than the heuristic algorithms of [3], [10]. We consider primary paths with the same destination node and design via MIP, a tree structure that serves as the protection path of the link-disjoint primary paths.

The second technique addresses dynamic provisioning where traffic changes by a bandwidth-on-demand paradigm. For this problem, we have developed an Integer Linear Programming (ILP) approach. This algorithm has a very low complexity while being adaptive to changes in traffic demand.

The first technique is described in Section II under the title "Diversity Coding Tree," and the second one is described in Section III under the title "Dynamic Provisioning."

## II. Diversity Coding Tree

Previously, we introduced a heuristic algorithm to map the basic diversity coding structures to networks with arbitrary topology [3], [10]. It is desirable to replace this heuristic with one that is based on an optimum formulation. Such a formulation can be achieved by using a form of Linear Programming (LP). In this section, we present an optimal Mixed Integer Programming (MIP) approach. This approach maps diversity coding to networks with arbitrary topology and static traffic scenarios. One of the most significant advantage of this approach, which we call the *Diversity Coding Tree* algorithm, is the reduction of complexity.

One drawback of the previous heuristic technique was the high variance between the restoration times of the coding group combinations. Depending on the topology used, some coding groups need multiple times of restoration time of others. Therefore, we adopted a version of diversity coding in which connections are coded if they have the same destination node. This leads decoding operations to be performed at the common destination node and eliminates the time consuming decoding and complex synchronization operations at the intermediate nodes.

The optimal diversity coding tree algorithm was inspired by a $p$-cycle approach that uses a cycle exclusion technique [14]. It builds trees for generating protection paths for connections that end on the same destination node. The branches of these trees can merge until they reach the root of the tree, or the destination node. An example is provided in Fig. 1. In this figure, nodes $S1$, $S2$, and $S3$ transmit their data, given as $a$, $b$, and $c$, respectively, to the common destination node $D$. These paths are shown with solid black lines and arrows. The protection is disjoint to all these paths and it encodes the data to be protected on the tree structure, shown in this case with dashed black lines and arrows. Such a structure has two advantages. First it reduces the complexity of the LP substantially by decomposing the problem into smaller subproblems without loss of optimality. Every subproblem inputs only the connections with a specific destination node. Therefore, connections are divided into groups based on their destination nodes and each group is run independently. Second, it eliminates the feedback links used in the general diversity coding approach. As a result, the complexity of the LP formulation, the restoration time, signaling, and synchronization complexity is reduced. The solution is applicable to any static traffic requirements.

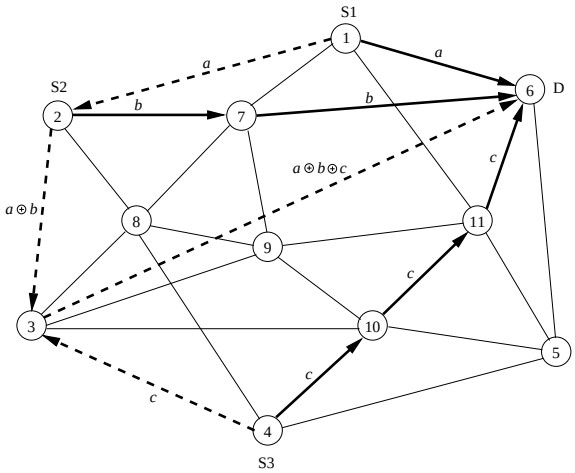


Fig. 1. An example of the diversity coding tree structure. It employs COST239 network topology.

The formulation is provided below. The input parameters are provided as

- $G(V, E)$ : Network graph,
- $N$ : Enumerated list of all connections,
- $a_e$ : Cost associated with link $e$,
- $T$ : Maximum number of diversity coding trees allowed, typically one third of the number of connections in each subproblem,
- $\Gamma_i(v)$ : The set of incoming links of each node $v$,
- $\Gamma_o(v)$ : The set of outgoing links of node $v$,
- $\alpha$ : A constant employed in the algorithm, chosen very small,
- $\beta$ : A constant employed in the algorithm, chosen very large.

Next we provide the variables. With one exception, they are binary and take the value of 0 or 1.

- $x_e(i)$ : Equals 1 iff the primary path of connection $i$ passes through link $e$,
- $n(i, t)$ : Equals 1 iff connection $i$ is protected by the diversity coding tree $t$,
- $c_e(t)$ : Equals 1 iff the diversity coding tree $t$ passes through link $e$,

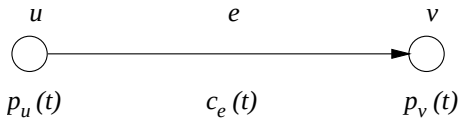


Fig. 2. A typical link in the diversity tree $t$.

- $p_v(t)$ : A continuous variable between 0 and 1, resulting in an MIP formulation. It keeps the "voltage" value of node $v$ in tree $t$. It is possible to set this variable as an integer larger than 0 but that makes the simulation slower.

The following inequality builds the primary paths of each connection

$$\sum_{e\in\Gamma_i(v)} x_e(i) - \sum_{e\in\Gamma_o(v)} x_e(i) = \begin{cases} -1 & \text{if } v = s_i, \\ 1 & \text{if } v = d_i, \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where $s_i$ and $d_i$ are the source and destination nodes of connection $i$, respectively. The following inequalities build a valid diversity coding tree. If the node $v$ is a destination node, the tree must terminate at the destination node using one of the incoming links. The outgoing links of the destination node must be empty of the tree branches. To ensure these properties, the required inequalities are

$$\sum_{e\in\Gamma_i(v)} c_e(t) \geq \frac{\sum_{i=1}^{N} n(i,t)}{\beta} \quad \forall e \in E, t, \quad (2)$$

$$\sum_{e\in\Gamma_o(v)} c_e(t) \leq 0 \quad \forall e \in E, t. \quad (3)$$

If it is a source node then there must be one outgoing link that belongs to the diversity coding tree. It is stated by

$$\sum_{e\in\Gamma_o(v)} c_e(t) \geq \frac{n(i,t)}{\beta} \quad \forall e \in E, t. \quad (4)$$

At the intermediate nodes, if at least one branch of diversity coding tree gets into the node then the tree must use on the outgoing links to travel to the destination (root) node. The inequalities needed under this rule are

$$\sum_{e\in\Gamma_o(v)} c_e(t) \geq \frac{\sum_{e\in\Gamma_i(v)} c_e(t)}{\beta} \quad \forall e \in E, t. \quad (5)$$

In order to prevent getting cyclic (or loop) structures inside the diversity coding trees, we choose to assign a "voltage" value to each node in the tree, as in [14]. We would like to emphasize that this "voltage" value is only used in the sense of a resemblance to the familiar Kirchoff's voltage law. It is an assigned variable to prevent loops, and has nothing to do with actual voltages. In this formulation, the voltage value at the head node should be higher than the voltage value at the tail node of the links which are part of the diversity coding tree. Fig. 2 shows a typical link in the network. The voltage value of node $v$ must be higher than the voltage value of node $u$. This voltage relationship prevents the cyclic structure to be a part of the diversity coding trees such as in Fig. 3.

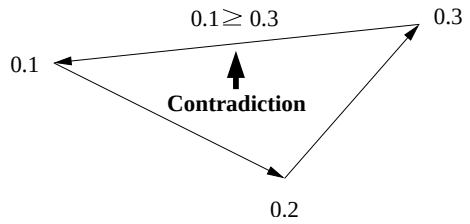


Fig. 3. Voltage value contradiction in a loop structure.

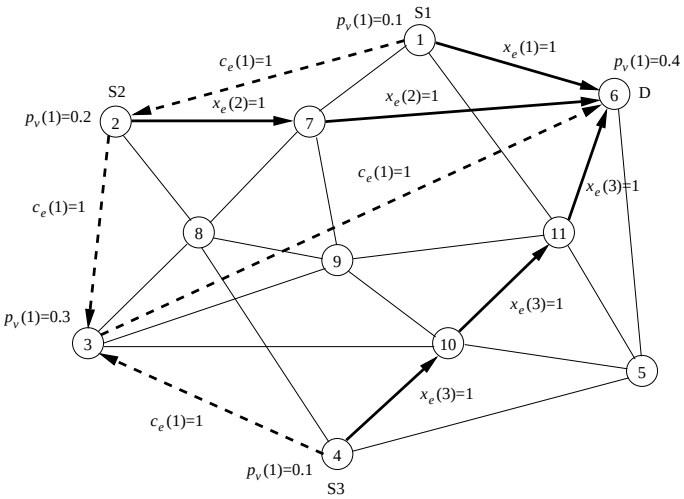


Fig. 4. A diversity coding example with the MIP variables set. The values $n(i,t) = 1$ for $i = 1, 2, 3$.

The inequality below expresses what is described in the previous paragraph mathematically.

$$p_v(t) - p_u(t) \geq \alpha \cdot c_e(t) - (1 - c_e(t)) \quad \forall e = u \to v, \forall t. \quad (6)$$

In addition, the program has a set of inequalities to satisfy the link disjointness property between primary paths and the diversity coding tree that supports them. This is ensured by the following constraint

$$x_e(i) + c_e(t) \leq 2 - n(i,t) \quad \forall e \in E, i, t. \quad (7)$$

The primary paths must also be link disjoint with each other if they are protected via the same tree. This is formulated as

$$x_e(i) + x_e(j) \leq 3 - n(i,t) - n(j,t) \quad \forall e \in E, i, j, t. \quad (8)$$

The final constraint ensures that a connection can be protected by only one diversity coding tree

$$\sum_{i=1}^{T} n(i,t) = 1 \quad \forall i. \quad (9)$$

The objective function is

$$\min \left( \sum_{e\in E} \sum_{i=1}^{n} a_e x_e(i) + \sum_{e\in E} \sum_{t=1}^{T} a_e c_e(t) \right). \quad (10)$$

The previous example network is shown in Fig. 4. There is a traffic pattern that the program finds the tree previously

TABLE I
$SCP$ WITH DIVERSITY CODING TREE

| COST 239 Network, 11 nodes, 26 spans | | | |
|---|---|---|---|
| Dest. Node | Total Capacity | Shortest Capacity | $SCP$ |
| Node 1 | 1906 | 1053 | 81% |
| Node 2 | 2170 | 1241 | 74% |
| Node 3 | 2235 | 1321 | 69% |
| Node 4 | 2502 | 1410 | 77% |
| Node 5 | 2026 | 1153 | 75% |
| Node 6 | 3408 | 1887 | 80% |
| Node 7 | 1746 | 892 | 95% |
| Node 8 | 1745 | 838 | 108% |
| Node 9 | 1246 | 662 | 88% |
| Node 10 | 2191 | 1031 | 125% |
| Node 11 | 1501 | 798 | 88% |
| Overall | 22676 | 12286 | 84% |

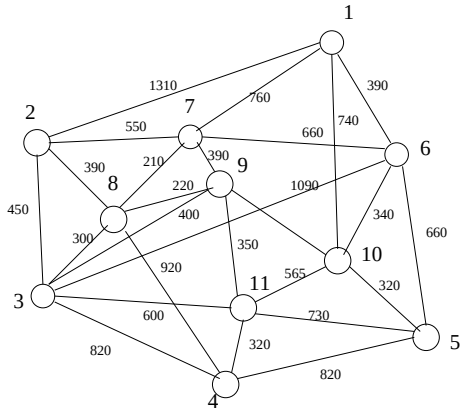


Fig. 5. European COST 239 network.

shown so that the values $c_e(1) = 1$ on the edges $e$ of this tree. The values $n(i,t)$ identify the connections to be protected. The values $p_v(1)$ are set by the recursive expression (6).

### A. Pre-Provisioning Results

We have some initial results using this approach. We have used the pan-European COST239 [15] network depicted at Fig. 5. We calculated the Spare Capacity Percentage ($SCP$) as

$$SCP = \frac{Total\ \ Capacity - Shortest\ \ Working\ \ Capacity}{Shortest\ \ Working\ \ Capacity}.$$

The traffic scenario is taken from [14] and made symmetric for our simulations. The results are provided in Table I. Shortest capacity at Table I refers to total capacity when there is no protection and connections are routed over shortest paths. Each row of the Table I refers to a different subproblem where the destination node is fixated from node 1 to node 11 for each subproblem. The last row is the summation of the results throughout the whole network. We know from the LP software that the best results have not been achieved for some nodes due to computer memory limitations. Therefore, there is still some space for improvement. Nevertheless, the result for the overall network in Table I of 84% is 8% better than the result achieved via the heuristic algorithm given at [3]. For a fair comparison, common destination diversity coding is also employed in the heuristic algorithm, which resulted in 92% $SCP$. We also note that the nodes at the edges of the network result in lower $SCP$ as compared to the nodes inside the network. This shows that our intuition based on the results of heuristic algorithm is correct. However, we now have a significantly better algorithm. In addition, this result was calculated in a much shorter time. In the diversity coding tree algorithm, the problem is decomposed into smaller subproblems and each subproblem can be run in parallel which increases the speed significantly.

## III. DYNAMIC PROVISIONING

In the previous section, we discussed the design of diversity coding restoration in pre-provisioning of static traffic. In this problem, the traffic is known *a priori.* On the other hand, in a dynamic environment, connections are dynamically set up under a bandwidth-on-demand paradigm. The dynamic provisioning of these connections against single link failures is an important problem. In this section, we will discuss the design of diversity coding restoration for this problem.

It can be observed that the complexity of the design problem is substantially simplified if the provisioning of each connection is carried out one-by-one instead of optimizing the whole set of connections at once. We realistically assume that due to preserving the Quality-of-Service (QoS) requirements of the existing connections, one does not tear down the existing solution as a new demand arises. The result is a design algorithm with low complexity. The set of assumptions are as follows.

1. The existing connections cannot be rearranged due to QoS requirements.
2. At the beginning, the demand matrix is an empty set.
3. Centralized information about the state of the network is updated and conveyed to the nodes every time there is a change.
4. Every node is able to run the algorithm and calculate the routes.
5. Connections can be set up and terminate after some duration.

As in the previous section, the structure in this application is diversity coding with a single destination. In other words, only the connections with the same destination node are coded together. This has the advantages of lower complexity, lower restoration time, lower signaling, and higher coding flexibility. The fact that all of the decoding operations are carried out at a single node simplifies the coding-decoding structure. Both the primary and protection paths are coded together which adds up to the coding flexibility of diversity coding.

When the algorithm receives a new connection demand, it detects its destination node. Then it finds the diversity coding groups which have the same destination node as the new demand because it will attempt to add the new demand to one of them. In order to satisfy decodability and save capacity, one of the paths of the new connection must be link disjoint to the links used in the coding group whereas the other path must be combined and coded with one of the paths of the coding

group. As an example, assume that we have an existing coding structure with four connections. The received matrix is

$$\begin{bmatrix} a_{11}x + a_{12}y + a_{13}z + a_{14}v \\ a_{21}x + a_{22}y + a_{23}z + a_{24}v \\ a_{31}x + a_{32}y + a_{33}z + a_{34}v \\ a_{41}x + a_{42}y + a_{43}z + a_{44}v \\ a_{51}x + a_{52}y + a_{53}z + a_{54}v \end{bmatrix} \tag{11}$$

where $x$, $y$, $z$, and $v$ are the coded signals and $a_{ij}$ are the binary coding parameters. The matrix $\mathbf{A} = [a_{ij}]_{5\times 4}$ is still full rank if we delete one of its rows. When a new connection will be added to the group, one of its paths will be combined with one of the paths in the coding group. The secondary path of this connection will be link disjoint with the links in the coding group. As a result, the received matrix is transformed to the following format

$$\begin{bmatrix} a_{11}x + a_{12}y + a_{13}z + a_{14}v + 0l \\ a_{21}x + a_{22}y + a_{23}z + a_{24}v + 0l \\ a_{31}x + a_{32}y + a_{33}z + a_{34}v + 0l \\ a_{41}x + a_{42}y + a_{43}z + a_{44}v + 0l \\ a_{51}x + a_{52}y + a_{53}z + a_{54}v + l \\ 0x + 0y + 0z + 0v + l \end{bmatrix}. \tag{12}$$

The link disjoint path of the new connection is represented as the sixth row of the new matrix, as $l$ is the signal of the new connection. The other copy of signal $l$ is coded with other signals in the coding group as denoted in the fifth row. For decodability, the new signal should be coded with at most one path in the coding group. The proof of decodability of the new coding structure is straightforward. For the no link failure case, we can derive $x$, $y$, $z$, and $v$ by solving first four rows and we can derive $l$ using the last row. If we delete one of the first four rows, then the received matrix becomes

$$\begin{bmatrix} 0x + 0y + 0z + 0v + 0l \\ a_{21}x + a_{22}y + a_{23}z + a_{24}v + 0l \\ a_{31}x + a_{32}y + a_{33}z + a_{34}v + 0l \\ a_{41}x + a_{42}y + a_{43}z + a_{44}v + 0l \\ a_{51}x + a_{52}y + a_{53}z + a_{54}v + l \\ 0x + 0y + 0z + 0v + l \end{bmatrix}. \tag{13}$$

We derive $l$ from the last row and subtract it from the fifth row. Then the matrix generated from rows 2 to 5 that multiplies the vector $(x, y, z, v)^T$ has full rank. Therefore, all of $x$, $y$, $z$, and $l$ can be extracted. If we delete the fifth row, no extra operation is required to extract all of the signals. If the last row is deleted as

$$\begin{bmatrix} a_{11}x + a_{12}y + a_{13}z + a_{14}v + 0l \\ a_{21}x + a_{22}y + a_{23}z + a_{24}v + 0l \\ a_{31}x + a_{32}y + a_{33}z + a_{34}v + 0l \\ a_{41}x + a_{42}y + a_{43}z + a_{44}v + 0l \\ a_{51}x + a_{52}y + a_{53}z + a_{54}v + l \\ 0x + 0y + 0z + 0v + 0l \end{bmatrix}, \tag{14}$$

then, the first four rows can be used to extract signals $x$, $y$, $z$, and $v$. These signals can then be used to find the value of $l$ from the fifth row.

There is one critical point in order to satisfy the decodability of the coding structure. None of the paths in the coding structure must diverge after any intermediate node. Otherwise, a new connection can be coded in multiple rows in the received matrix, impairing the full-rank property under some failure scenarios.

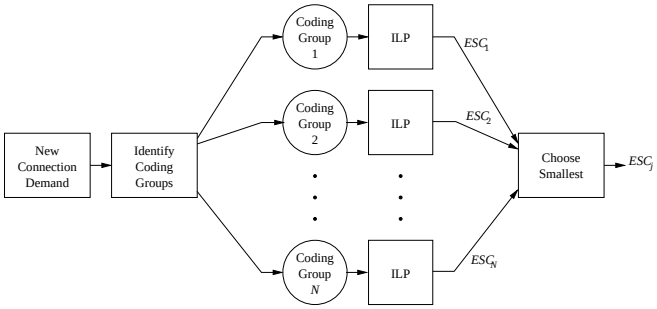


Fig. 7. Extra spare capacity is calculated for each coding scenario and the minimum is chosen.

We have developed an integer linear programming (ILP) based algorithm to protect new demands for connection in a network. It is optimal under the assumptions previously stated. This algorithm attempts to provide protection for the new demand with the lowest extra spare capacity required. The important observation to make here is the fact that some capacity can be saved by adding the new connection to one of the existing coding groups with the same destination node. The algorithm investigates each coding group and finds out how much extra spare capacity is required after adding the new connection in each coding group. Since this extra capacity changes with respect to each coding group, the total capacity to route and protect the new connection is calculated after every addition. The algorithm then chooses the coding group which incurs the lowest extra spare capacity to route and protect the new connection. This operation is depicted in Fig. 7. In this figure, $ESC$ means extra spare capacity required to add the new connection to each coding group. As will be described in detail in the sequel, we use an ILP algorithm to find link disjoint primary and protection paths for every coding scenario. The cost vector of the links is adjusted for every coding scenario, depending on the topology of the coding group. The coding group which requires the least extra spare capacity is chosen to add the new connection. One of the coding groups in Fig. 7 is an empty group which lets the new connection demand to start a new coding group if that solution requires the lowest $ESC$. The coding group is updated with the new connection and the algorithm is ready to incorporate a new connection demand.

The parameters of the ILP formulation to find a pair of link disjoint primary and secondary paths are as follows.

- $G(V, E)$ : Network graph,
- $N$ : Enumerated list of all connections,
- $a_e$ : Cost associated with link $e$,
- $\Gamma_i(v)$ : The set of incoming links of each node $v$,
- $\Gamma_o(v)$ : The set of outgoing links of node $v$.

The binary ILP variables which take the value of 0 or 1 are

- $x_e$ : Equals 1 iff the primary path of the connection passes through link $e$,
- $y_e$ : Equals 1 iff the secondary path of the connection passes through link $e$.

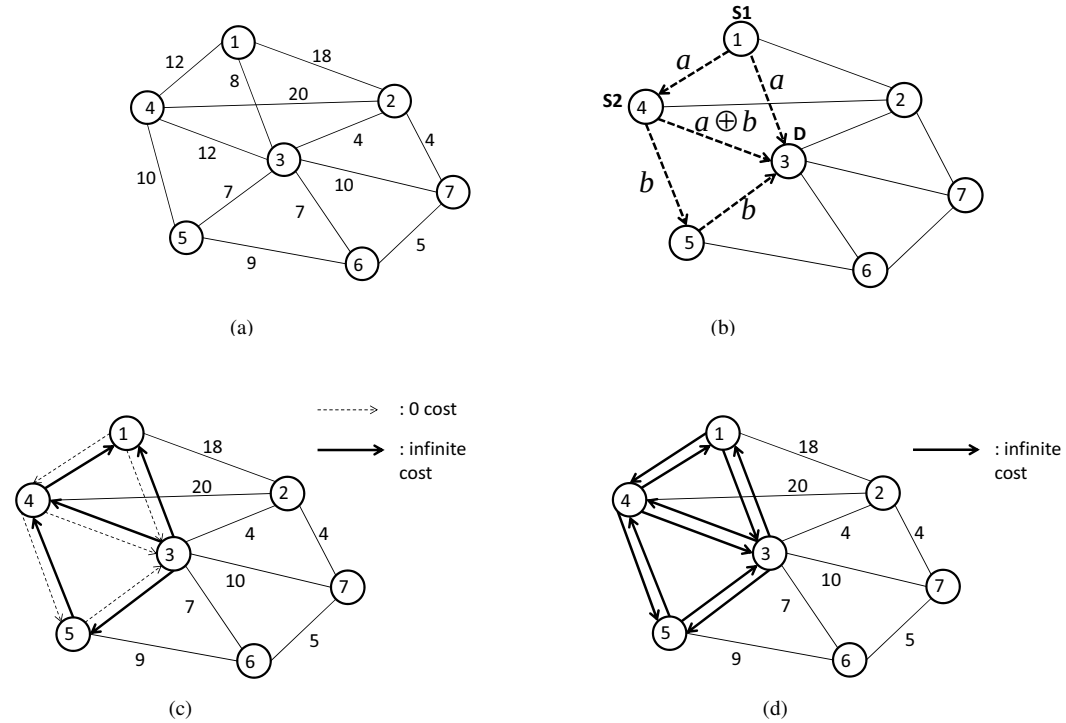


Fig. 6. (a) The topology of the network associated with the regular link costs, (b) An existing coding group, (c) The link costs as the primary path of a new connection is routed, (d) The link costs as the secondary path of a new connection is routed.

The objective function is

$$\min \sum_{e \in E} (x_e + y_e) \cdot a_e. \tag{15}$$

The origination, flow, and termination of the primary path ($x_e$) and the secondary path ($y_e$) are determined by

$$\sum_{e \in \Gamma_i(v)} x_e - \sum_{e \in \Gamma_o(v)} x_e = \begin{cases} \text{-1} & \text{if } v = s, \\ 1 & \text{if } v = d, \\ 0 & \text{otherwise.} \end{cases} \quad \forall v. \tag{16}$$

$$\sum_{e \in \Gamma_i(v)} y_e - \sum_{e \in \Gamma_o(v)} y_e = \begin{cases} \text{-1} & \text{if } v = s, \\ 1 & \text{if } v = d, \\ 0 & \text{otherwise.} \end{cases} \quad \forall v. \tag{17}$$

The link disjointness between the primary and secondary paths is satisfied by

$$x_e + y_e \leq 1 \quad \forall e. \tag{18}$$

The adjusted coding vector is not only useful in calculating the extra spare capacity required but also it also helps to route the primary and secondary paths of the new connection. Each connection has two paths, one of them is supposed to be coded with the established coding group and the other must be link-disjoint to the coding group. Coding a new flow with one of the links at the coding group incurs zero cost, which means the cost of that link is set to zero for that path. On the other hand, that specific link has infinite cost for the other path which is designed to be link-disjoint to the coding group. For decodability purposes, once the primary path is coded with one of the paths in the coding group, it will stay on that path till the destination node is reached. Otherwise, the primary path may span multiple paths in the same coding group ,which can impair the full rank property under some failure scenarios. It has no effect on the total cost since every link in coding group has zero cost for the primary path.

### A. Cost Adjustment Example

In Fig. 6(a), there are two unidirectional links between each node and the numbers next to them are their costs. As an example, we have a coding group that is shown in Fig. 6(b). There are two source nodes $S1$ and $S2$ and one destination node $D$. The signals transmitted from $S1$ and $S2$ are $a$ and $b$ respectively. The third path also carries the coded version of these signals to the destination node. The dashed lines show links that are incorporated in the coding group. Assume that we have a new connection request from any arbitrary node (except node 3) to node 3. We want to calculate the required spare capacity to add this connection to the coding group. When we run the ILP formulation, there will be two different topologies in terms of the costs of the links. The topology for the primary path differs from the topology for the secondary path depending on the existing coding group. This fact is visualized in Fig. 6(c) and Fig. 6(d) for the primary and secondary paths, respectively. In Fig. 6(c), the cost of links shown by dashed lines have zero cost for the primary path. On the other hand, the cost of the solid black lines is infinite for the same path. In Fig. 6(d), the solid black lines have infinite cost when the secondary path is routed. The links which are not associated with the coding group have the same regular cost for both of the paths as shown in Fig. 6(c) and 6(d).

### B. Limited Capacity Case

All of the discussion about the dynamic provisioning algorithm so far was based on the assumption that there is enough capacity on each link to place the primary and secondary paths. Otherwise, some of the new connection demands may be blocked due to insufficient capacity over the trail of the candidate paths. When the link capacities are limited, the cost vector of the links is adjusted within the consideration of both free capacity of each link and the topologies of the suitable coding groups. In addition to the adjustments of the example at Section III-A, the cost of a link is set to infinite for both of the paths if there is no available capacity on it. At the end, a new demand is blocked if it can not be routed and protected with a finite cost.

### C. Connection Teardown

We need to update the algorithm when a connection leaves the demand matrix after a duration. The teardown operation consists of three steps.

- First, the connection is dropped from its coding group and the topology of that coding group is updated. It is done by subtracting the links which purely carry the signal of interest from that coding group topology. The links that carry the coded version of this signal are kept in the coding group topology. Referring to the example at Fig. 6(b), if the connection $S2-D$ leaves the network then the links 4-5 and 5-3 are subtracted from the coding group topology. However, the link 4-3 stays in the coding group since it continues to carry the signal $a$.
- In the second step, the received matrix of the coding group is updated by excluding the signal associated with the leaving connection.
- In the last step, the capacity of the links that are subtracted from the respective coding group are increased by 1. It should be noted that the last step is required only if the capacity of the links are limited.

## IV. Conclusion

In this paper, we introduced two algorithms that offer near-hitless recovery against link failures for static and dynamic traffic. In both cases, we used the basic technique of *diversity coding*. However, unlike the original work introducing this technique, we used diversity coding in networks with arbitrary topology.

The first algorithm is an optimal MIP formulation to protect a given set of connections by jointly optimizing the primary paths and the coded restoration paths. We call the graph that defines the optimum protection path the diversity coding tree. This tree is used to protect primary paths whose destination is the same node. This choice is made so as to minimize the restoration time. In addition to the restoration time, the choice of a common destination significantly decreases the number of variables and inequalities in the MIP formulation. As a result, this formulation runs faster, and results in better restoration times, than our previous formulation based on a heuristic algorithm.

The second algorithm is for traffic demand that arrives dynamically. This algorithm employs an ILP-based formulation for diversity coding for dynamic connections in a dynamic fashion. It identifies a number of "coding groups" which have a common destination node, and picks the one with the smallest cost, or lowest spare capacity. If no coding group is available, then the connection initiates a new group by itself.

## References


[1] W. D. Grover, *Mesh-Based Survivable Networks: Options and Strategies for Optical, MPLS, SONET, and ATM Networking*. Prentice-Hall PTR, 2004.

[2] M. Menth, M. Duelli, and J. Milbrandt, "Resilience analysis of packet-switched communication networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 6, p. 1, December 2009.

[3] S. Avci and E. Ayanoglu, "Recovery from link failures in networks with arbitrary topology via diversity coding," in *Proc. IEEE GLOBECOM*, December 2011, pp. 1–6.

[4] J.-P. Vasseur, M. Pickavet, and P. Demeester, *Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. Elsevier, 2004.

[5] G. Li, A. Chiu, and J. Strand, "Resilience design in all-optical ultralong-haul networks," *J. Opt. Netw.*, vol. 5, no. 8, pp. 625–636, July 2006.

[6] T. Chow, F. Chudak, and A. Ffrench, "Fast optical layer mesh protection using pre-cross-connected trails," *IEEE/ACM Trans. Netw.*, vol. 12, no. 3, pp. 539–548, June 2004.

[7] W. Grover and D. Stamatelakis, "Cycle-oriented distributed preconfiguration: ring-like speed with mesh-like capacity for self-planning network restoration," in *Proc. ICC '98*, vol. 1, 1998, pp. 537–543.

[8] E. Ayanoglu, C.-L. I, R. D. Gitlin, and J. E. Mazo, "Diversity coding: Using error control for self-healing in communication networks," in *Proc. IEEE INFOCOM '90*, vol. 1, June 1990, pp. 95–104.

[9] ——, "Diversity coding for transparent self-healing and fault-tolerant communication networks," *IEEE Trans. Commun.*, vol. 41, pp. 1677–1686, November 1993.

[10] S. Avci, X. Hu, and E. Ayanoglu, "Hitless recovery from link failures in networks with arbitrary topology," in *Proc. of the Information Theory and Applications Workshop*, February 2011, pp. 1–6.

[11] S. Avci and E. Ayanoglu, "Coded path protection: Efficient conversion of sharing to coding," in *Proc. IEEE ICC (to appear)*, June 2012.

[12] S. Ramamurthy, L. Sahasrabuddhe, and B. Mukherjee, "Survivable WDM mesh networks," *J. Lightwave Technol.*, vol. 21, no. 4, pp. 870–883, April 2003.

[13] D. Traskov, N. Ratnakar, D. Lun, R. Koetter, and M. Médard, "Network coding for multiple unicasts: An approach based on linear optimization," in *Proc. IEEE ISIT*, July 2006, pp. 1758–1762.

[14] B. Wu, K. L. Yeung, and P.-H. Ho, "ILP formulations for *p*-cycle design without candidate cycle enumeration," *IEEE/ACM Trans. Netw.*, vol. 18, no. 1, pp. 284–295, February 2010.

[15] P. B. *et al.*, "Ultra high capacity optical transmission networks: Final report of action COST 239," Faculty Elect. Eng. Computing, Univ. Zagreb, Zagreb, Croatia, Tech. Rep., 1999.