

Spare Capacity Allocation for Non-Linear Link Cost and Failure-Dependent Path Restoration

Yu Liu David Tipper

Department of Information Science and Telecommunications

University of Pittsburgh, Pittsburgh, PA 15260, USA

{yuliu, tipper}@tele.pitt.edu

Abstract— The design of survivable mesh based communication networks has received considerable attention in recent years. One task is to route backup paths and allocate spare capacity in the network to guarantee seamless communications services survivable to a set of failure scenarios. This is a complex multi-constraint optimization problem, called spare capacity allocation (SCA). This paper presents a matrix based model for the NP-complete SCA problem on directed networks and develops an approximation algorithm, termed successive survivable routing (SSR) to include non-linear capacity cost and failure-dependent path restoration. The SCA problem using a modular link cost function is studied using SSR. Numerical results of total spare capacity and total cost are compared showing that SSR finds solutions with lower total costs when a non-linear cost function is considered. Moreover, the SCA problem using failure-dependent path restoration, which finds multiple backup paths for each working path, is discussed. A numerical comparison among different path restoration schemes using SSR is given showing the tradeoffs.

Keywords— spare capacity allocation, protection and restoration algorithms, network survivability, survivable routing

I. INTRODUCTION

NETWORK survivability techniques have been proposed to guarantee seamless communication services in the face of network failures. Traditionally there are two components to making a network survivable, namely: 1) network design and capacity allocation, and 2) traffic management and restoration. Network design and capacity allocation techniques try to mitigate system level failures such as loss of a network link, by placing sufficient diversity and capacity in the network topology. For example, designing the topology and determining the capacity of links in a backbone network so that the network can carry the projected demand even if any one link is lost due to a failure. Traffic management and restoration procedures seek to direct the network load such that a failure has minimum impact when it occurs and that connections affected by the failure are restored while maintaining network stability. An example is the use of dynamic fault recovery routing algorithms to make use of the spare capacity remaining after a failure. These two components are complementary and cooperate to achieve seamless service upon failures.

The *spare capacity allocation* (SCA) problem is to decide how much spare capacity should be reserved on network links and where to route backup paths to protect given working traffic paths when the network topology is at least 2-connected. A literature review of different spare capacity allocation algorithms and restoration schemes has been given in [1].

Currently, survivable techniques assume the network design phase is centrally implemented while the restoration phase is distributed. This survivability framework has been reexamined recently, as increasing requests for bandwidth-on-demand services have fostered an open market to trade backbone network capacity. How to improve the cost performance ratio of the spare resources by using this flexible bandwidth market is becoming an important issue. Moreover, the current backbone network architecture is moving towards a two-layered architecture with packets transported directly over an optical network. In order to provide rapid restoration in a distributed manner such an architecture strongly favors restoration schemes where protection paths are pre-calculated and spare bandwidth is reserved and shared among several backup paths which do not have any common risk [2], [3], [4], [5].

Thus, recently distributed restoration schemes which reserve pre-planned spare capacity have been introduced to provide fast restoration as well as reduce the cost of spare resources. For example, the resource aggregation fault tolerance (RAFT) scheme for RSVP-based IntServ services was developed by Dovrolis and Ramanathan [5] and Sharing with Partial Information (SPI) for IP/MPLS networks was proposed by Kodialam and Lakshman [6], [7], [8]. These schemes focus on protecting working traffic by reserving spare bandwidth in backup paths. Though network redundancy is reduced to some degree by sharing spare capacity in these schemes, our recent results [1] have shown that total spare capacity can be reduced to near minimum by using a distributed algorithm, called *successive survivable routing* (SSR). SSR routes backup paths sequentially and maintains a small amount of state information with $O(L^2)$ complexity, where L is the number of links in the network and the protection is targeted at all single link failures. In this paper we present a matrix based formulation of SSR and extensions of the SSR method to incorporate non-linear link cost and failure-dependent path restoration.

The rest of the paper is organized as follows. In Section II, a novel matrix-based arc-flow integer programming model is formulated to minimize the total cost of spare capacity on directed networks. Section III presents the successive survivable routing algorithm to approximate the optimal solution to the matrix based SCA problem. Section IV uses a modular cost function to show the benefit of using SSR for networks with non-linear link cost functions. Section V extends the original restoration scheme, *path restoration with link-disjointed backup route or failure-*

independent path restoration (FID), to *failure-dependent path restoration* (FD). In addition, the *stub release* function [9] is captured. This function releases the unused working capacity when a failure happens, further reducing the total spare capacity needed. Numerical results comparing the total spare capacities required as determined by the SSR scheme using the above three path restoration schemes, FID, FD, and FD with stub release (FDStubR), are provided. Section VI concludes the paper.

II. A MATRIX-BASED ARC-FLOW MODEL

In this section, the spare capacity allocation (SCA) problem is formulated to protect any given failure scenarios considering failure-independent (FID) path restoration. If any single link failures are the scenarios, this approach is called path restoration with link-disjoint routes, since a backup path is always link disjoint from its working path. We assume all traffic flows require a 100% restoration level for any single failure. This level of restoration requires that all affected flows be detoured to their backup paths upon any given failure. Provisioning enough spare capacity is the prerequisite condition to such restoration. Given a network topology, traffic flows and their working paths, the objective of SCA is to minimize the total cost of spare capacity. The decision variables include where to route backup paths and how much spare capacity should be reserved. Backup paths can share capacity on their common links if their corresponding working paths are not subject to the same failure. For example, consider two flows in Network 1 of Fig. 5 with the link-disjoint working paths 1-2 and 1-3. If their backup paths are 1-4-2 and 1-4-3, these two backup paths can share their spare capacity reservations on link 1-4 to resile any single link failure.

We have provided a matrix representation of the SCA problem for undirected networks in [1]. Here, a general matrix representation for directed networks is given. A network is represented by a directed graph of N nodes and L links with R flows. For simplicity the physical link capacity is assumed unlimited. Note that the approach can be generalized to incorporate capacitated links by adding constraints or using non-linear link cost functions as detailed in [10].

A flow r , $1 \leq r \leq R$ is specified by its origin/destination node pair $(o(r), d(r))$ and traffic demand m_r . Working and backup paths of flow r are represented by two $1 \times L$ binary row vectors $\mathbf{p}_r = \{p_{rl}\}$ and $\mathbf{q}_r = \{q_{rl}\}$ respectively. The l -th element in one of the vectors equals to one if and only if (iff) the corresponding path uses link l . The path link incidence matrices for working and backup paths are the collections of these vectors, forming two $R \times L$ matrices $\mathbf{P} = \{p_{rl}\}$ and $\mathbf{Q} = \{q_{rl}\}$ respectively. Let $\mathbf{M} = \text{Diag}(\{m_r\}_{R \times 1})$ denote the diagonal matrix representing the demands of each flow. Note that if the protection level of flows is under 100%, the elements in \mathbf{M} can be adjusted to reserve partial spare capacities on backup paths.

We characterize K failure scenarios in a binary matrix $\mathbf{F} = \{\mathbf{f}_k\}_{K \times 1} = \{f_{kl}\}_{K \times L}$. The row vector \mathbf{f}_k in \mathbf{F} is for failure scenario k and its element f_{kl} equals one iff

TABLE I
NOTATION

| | |
|--|---|
| N, L, R, K | Numbers of nodes, links, flows and failure scenarios |
| n, l, r, k | Indices of nodes, links, flows and failures; $1 \leq n \leq N, 1 \leq l \leq L, 1 \leq r \leq R, 1 \leq k \leq K$ |
| $\mathbf{P} = \{\mathbf{p}_r\} = \{p_{rl}\}$ | Working path link incidence matrix |
| $\mathbf{Q} = \{\mathbf{q}_r\} = \{q_{rl}\}$ | Backup path link incidence matrix |
| $\mathbf{M} = \text{Diag}(\{m_r\})$ | Diagonal matrix of capacity demand m_r of flow r |
| $\mathbf{G} = \{g_{lk}\}_{L \times L}$ | Spare provision matrix |
| $\mathbf{G}^r = \{g_{lk}^r\}_{L \times L}$ | Contribution of flow r to \mathbf{G} |
| $\mathbf{s} = \{s_l\}_{L \times 1}$ | Vector of link spare capacity assignments |
| $\phi = \{\phi_l\}_{L \times 1}$ | Spare capacity cost function |
| W, S | Total working capacity, total spare capacity |
| $\eta = S/W$ | Network redundancy |
| $o(r), d(r)$ | Origin/destination nodes of flow r |
| $\mathbf{v}_r = \{v_{rl}\}$ | Link metrics for flow r |
| $\mathbf{B} = \{b_{nl}\}_{N \times L}$ | Node link incidence matrix |
| $\mathbf{D} = \{d_{rn}\}_{R \times N}$ | Flow node incidence matrix |
| $\mathbf{F} = \{f_{kl}\}_{K \times L}$ | Binary failure link incidence matrix, $f_{kl} = 1$ iff link l fails in failure k |
| $\mathbf{U} = \{u_{rk}\}_{R \times K}$ | Binary flow failure incidence matrix, $u_{rk} = 1$ iff failure k will affect flow r 's working path |
| $\mathbf{T} = \{t_{rl}\}_{R \times L}$ | Binary flow tabu-link matrix, $t_{rl} = 1$ iff link l should not be used on flow r 's backup path |

link l fails in scenario k . In this way, we can represent scenarios with multiple simultaneously failed links. We also define a flow failure incidence matrix $\mathbf{U} = \{\mathbf{u}_r\}_{R \times 1} = \{u_{rk}\}_{R \times K}$, where $u_{rk} = 1$ iff flow r will be affected by failure k , and $u_{rk} = 0$ otherwise. A flow tabu-link matrix $\mathbf{T} = \{\mathbf{t}_r\}_{R \times 1} = \{t_{rl}\}_{R \times L}$ has $t_{rl} = 1$ iff the backup path of flow r should not use link l , and $t_{rl} = 0$ otherwise. We can find \mathbf{U} and \mathbf{T} given \mathbf{P} and \mathbf{F} as shown in (1) and (2) respectively. Note that, a binary matrix multiplication operation " \odot " is used in these two equations which modifies general addition $1 + 1 = 2$ to Boolean addition $1 + 1 = 1$.

$$\mathbf{U} = \mathbf{P} \odot \mathbf{F}^T \quad (1)$$

$$\mathbf{T} = \mathbf{U} \odot \mathbf{F} \quad (2)$$

The network topology is represented by the node-link incidence matrix $\mathbf{B} = (b_{nl})_{N \times L}$ where $b_{nl} = 1$ or -1 if and only if node n is the origin or destination of link l . We define $\mathbf{D} = (d_{rn})_{R \times N}$ as the flow node incidence matrix where $d_{rn} = 1$ or -1 iff $o(r) = n$ or $d(r) = n$.

We let $\mathbf{G} = \{g_{lk}\}_{L \times K}$ denote the *spare provision matrix* whose elements g_{lk} are the minimum spare capacity required on link l when failure k happens. Given the backup paths \mathbf{Q} , demand matrix \mathbf{M} , and flow failure incidence \mathbf{U} the spare provision matrix can be determined as in (5). The

minimum spare capacity required on each link is denoted by the column vector $\mathbf{s} = \{s_l\}_{L \times 1}$ which is found as in (4). The function \max in (4) asserts that any element in \mathbf{s} is equal to the maximum number in the corresponding row of \mathbf{G} . In this way, the minimum spare capacity on a link is always greater than or equal to the maximum spare capacity required by any single link failure.

Let $\phi_l(s_l)$ denote the link cost function of spare capacity on link l . $\phi(\mathbf{s}) = \{\phi_l(s_l)\}_{L \times 1}$ is a column vector of link costs. The total cost of spare capacity is $e^T \phi(\mathbf{s})$ where e is a column vector of all ones. Given the notation and definitions (summarized in Table I), the spare capacity allocation problem can be formulated as follows.

$$\min_{\mathbf{Q}, \mathbf{s}} \quad e^T \phi(\mathbf{s}) \quad (3)$$

$$\text{s.t.} \quad \mathbf{s} = \max \mathbf{G} \quad (4)$$

$$\mathbf{G} = \mathbf{Q}^T \mathbf{M} \mathbf{U} \quad (5)$$

$$\mathbf{T} + \mathbf{Q} \leq 1 \quad (6)$$

$$\mathbf{Q} \mathbf{B}^T = \mathbf{D} \quad (7)$$

$$\mathbf{Q} : \text{binary} \quad (8)$$

The objective function in (3) is to minimize the total cost of spare capacity by selection of the backup paths and spare capacity allocation. Note this formulation allows the use of realistic non-linear link cost functions. Constraints (4) and (5) give the method to calculate \mathbf{s} and \mathbf{Q} . Constraint (6) guarantees that backup paths will not use any link which might fail simultaneously with their working paths. For any single link failure, it assures backup paths are link-disjoint from their working paths. Flow conservation constraint (7) guarantees that backup paths given in \mathbf{Q} are feasible paths of flows in a *directed* network. This type of constraint is also called the mass balance constraint [11]. Only source and destination nodes of a flow have non-zero traffic accumulation, while its intermediate nodes only allow traffic transpassing.

The optimization model above (3)-(8) is an *arc-flow* multi-commodity flow model and is a *NP-complete* combinatorial problem [10]. Compared with the *path-flow* formulations in [9], [13], it needs additional constraints to find feasible backup paths, but pre-calculated backup path sets are not necessary. These changes make it suitable for an approximation algorithm based on iterative routing as developed in the next section.

As an example of the SCA problem structure and solution consider the five node network shown in Fig. 1. The network load is assumed a full mesh of unit capacity demands between each node pair, we set $\mathbf{M} = \mathbf{I}$ where \mathbf{I} is the identity matrix of size R . Shortest path routing is used for the working paths \mathbf{P} which are shown in Fig. 2. We consider the case of any single link failure which results in $\mathbf{F} = \mathbf{I}$. Then, $\mathbf{G} = \mathbf{Q}^T \mathbf{P}$. The matrices \mathbf{P} , \mathbf{G} , and the optimum values for \mathbf{Q} , and \mathbf{s} are given in Fig. 2. The total spare capacity reservation is given by $S = e^T \mathbf{s} = 11$.

One can see from the discussion above that the spare provision matrix \mathbf{G} plays a critical role in the SCA problem. Another way to compute \mathbf{G} is through aggregating

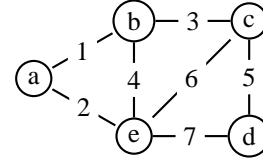


Fig. 1. Example five-node network, links are indexed numerically and nodes are indexed alphabetically

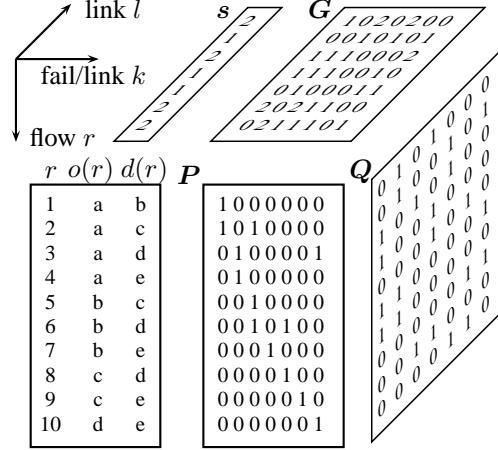


Fig. 2. SCA structure for failure-independent path restoration on the five-node network in Fig. 1

per-flow based information of working and backup paths. First, the contribution of a single traffic flow r to \mathbf{G} is given by $\mathbf{G}^r = \{g_{lk}^r\}_{L \times K}$ in (9), where \mathbf{u}_r and \mathbf{q}_r are the r th row vectors in \mathbf{U} and \mathbf{Q} . The spare provision matrix \mathbf{G} , can be found as in (10).

$$\mathbf{G}^r = m_r(\mathbf{q}_r^T \mathbf{u}_r), \quad \forall r, 1 \leq r \leq R \quad (9)$$

$$\mathbf{G} = \sum_{r=1}^R \mathbf{G}^r \quad (10)$$

Using the above matrix method, per-flow based information in \mathbf{P} , \mathbf{Q} is replaced by \mathbf{G} as the required information for finding backup paths of new arrival flows. The space complexity is reduced from $O(RL)$ to $O(LK)$ and it is independent of the number of flows R . This improves the scalability of the spare capacity sharing operation and makes it possible to be implemented in an online distributed fashion and discussed below.

III. SUCCESSIVE SURVIVABLE ROUTING

The successive survivable routing (SSR) algorithm has been introduced in [1]. In SSR each traffic flow first routes its working path, then routes its backup paths from its source node based on specially designed link metrics. The backup paths can be updated according to changing network status. This will further reduce the total cost of spare capacity. Here, we briefly provide new formulas based on the above model and give the implementation of SSR at the source node of flow r in Fig. 3.

In the flow chart, Step 1 initiates SSR on flow r with its working path \mathbf{p}_r , destination node $d(r)$ and the failure

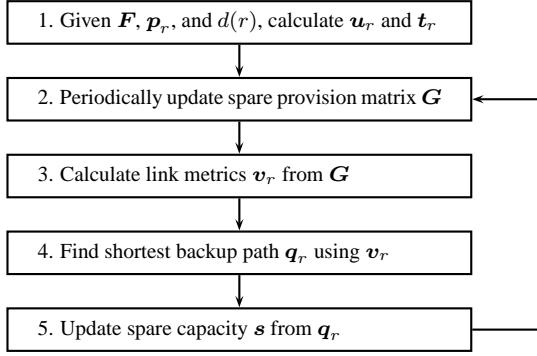


Fig. 3. Flow chart of the SSR algorithm at the source node of flow r

matrix F . Then, u_r and t_r , which are part of U and T in (1) and (2), are calculated.

Step 2 periodically collects current network state information, including G . Such state information is critical to find a backup path for flow r which can minimize the total additional spare capacity over the network. The update period of G should be long enough to guarantee the stability of the algorithm.

Keeping G up-to-date is important for the efficiency of a distributed protocol. There are two methods for collecting G over the network. One is link-based and the other is node-based. In the link-based method, the l -th row vector of G is stored in its corresponding link l (on its source node physically) and represents the required spare capacities for different failure scenarios on the link. The maximum element in this row vector is used to reserve the spare capacity on link l . The method to calculate this row vector is by collecting all the working paths whose backup paths use link l . Once all links have their row vectors up-to-date, a node collects all the row vectors from its outward links and form a link state packet with space complexity of $O(NK)$. The packet is then broadcasted to other nodes. In this way, a spare provision matrix G is synchronized over the network.

In the node-based method, the source node calculates the contribution matrix G^r for all flow r originated from it. Then this node aggregates all such G^r locally, and disseminates such information through link state packets. Compared to the above link-based method, though the node-based method increases the size of link state packets from $O(NK)$ to $O(LK)$, it does not require to spread working paths along their backup paths. Hence it requires less signaling support. Both methods can update the spare provision matrix G at the size of $L \times K$. The per-flow based path information is not required to be stored for backup path routing and spare capacity reservation. This improves the scalability and suitable for a distributed implementation of SSR.

Although keeping G synchronized takes time, it is not a critical drawback for the pre-planning of spare capacity in SSR. First, if SSR is used as a centralized algorithm, then state information synchronization is not required. Secondly, in a distributed implementation, the time scales of backup path provisioning and cost reduction of spare ca-

capacity are different. Backup paths are used for protection instead of carrying traffic. It is necessary for backup paths to be provided quickly, but the global spare capacity is only required to be reduced in a relatively longer time period. Each flow can find a backup path first, then update it later to reduce the total cost of spare capacity. Note that the different timing requirements further alleviates the scalability problem of the above state information synchronization process.

In Step 3, a shortest path algorithm is used to find a backup path q_r . The vector of link metrics v_r are found as follows:

(a) Given G , q_r and G^r for current flow r , let $G^- = G - G^r$ and $s^- = \max G^-$ be the spare provision matrix and the link spare capacity vector after backup path q_r is removed.

(b) Let q_r^+ denote an alternative backup path for flow r , and $G^{r+}(q_r^+) = m_r q_r^{+T} u_r$. Then, this new path q_r^+ produces a new spare capacity reservation vector $s^+(q_r^+) = \max(G^- + G^{r+}(q_r^+))$.

(c) Let $q_r^+ = e - t_r$, which assumes the backup path is using all possible links. Then, we can find a vector of *link metrics* for flow r as

$$\begin{aligned} v_r &= \{v_{rl}\}_{L \times 1} \\ &= \phi(s^+(e - t_r)) - \phi(s^-), \end{aligned} \quad (11)$$

where t_r is the binary flow tabu-link vector of flow r . An element v_{rl} is the incremental cost of spare capacity on link l if this link is on the backup path.

Step 4 improves the backup path by using a shortest path algorithm with link metrics v_r . The objective of this routing is to minimize the total incremental cost of spare capacity introduced by the new backup path. Since this routing algorithm not only finds a backup path, but also minimize the total cost of provisioning survivable service, we call this scheme *survivable routing* (SR).

In Step 5, if the backup path is changed in Step 4, the spare capacity reservations along the path will be updated accordingly. Since the backup path and its spare capacity are not used unless a failure happens, it is possible to modify current backup paths as well as the reserved spare capacity to reduce the total cost of spare capacity according to the changing traffic requirements and network states. An example of this approach is the *make-before-break* concept, proposed in RSVP extensions for LSP tunnels in MPLS networks [14].

After this step, the algorithm returns to Step 2 to start the next backup path update. This iterative process keeps improving the total cost of spare capacity. Thus the algorithm is called *successive survivable routing* (SSR). The algorithm terminates when there is no backup path update and it reaches a local optimum. Because the above iteration keeps reducing the objective function, SSR converges quickly. This fast convergence has been shown in extensive numerical results [1], [10].

IV. APPLICATIONS WITH MODULAR LINK COST

SSR has been compared with several other algorithms in [1] and shown to find the best trade-off between near optimality and fast computation speed. In this section, the applicability of the SSR algorithm to the SCA problem with non-linear link cost functions is shown.

Recently work on SCA with non-linear link cost has appeared using genetic algorithms [15] or branch and bound optimization with a limited search space [16]. It was shown that using non-linear objective functions directly in the optimization process finds lower costs than using a non-linear cost calculation after optimization with linear objective functions. Here we examine the performance of SSR when solving SCA with non-linear link cost.

SSR allows non-linear link cost function ϕ in the process without further modification. Since modularity of the link cost function is very common in practice, a modularity function is derived from several modular points from a non-linear envelope function $f(x) = ax^b$ at $x = 1, 3, 12, 48, 192$ for standard SONET carriers OC- x . Their costs are 3.7, 7.72, 19.55, 49.5, 125.32 respectively. The parameters, $a = 3.7$ and $b = \frac{2}{3} \approx 0.67$, are estimated from SONET service prices in [17]. The modular link cost function $\phi(\cdot)$ is given in (12). Both functions, $f(\cdot)$ and $\phi(\cdot)$, are shown in Fig. 4.

$$\phi(x) = \begin{cases} 0, & x = 0; \\ 3.7, & 0 < x \leq 1; \\ 7.72, & 1 < x \leq 3; \\ 19.55, & 3 < x \leq 12; \\ 49.5, & 12 < x \leq 48; \\ 125.32, & 48 < x \leq 192; \\ 125.32 \times \lceil x/192 \rceil, & x > 192 \end{cases} \quad (12)$$

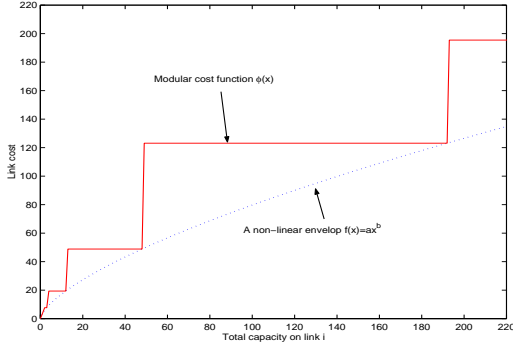


Fig. 4. Modular cost function $\phi(\cdot)$

The eight networks drawn from the literature shown in Fig. 5–12 were used for numerical experiments. They are summarized in Table II, which lists the number of network nodes N , links L , average nodal degree \bar{d} and number of demand pairs D . Note that the number of links L are for undirected networks. On directed networks, this number is doubled to represent two links for both directions. Without loss of generality, we assume symmetrical traffic flows between any node pairs and they follow the same path in both

directions. In all cases the network load was a full mesh of demand flows with the demand uniformly distributed between 1 to 10.

Working paths are given shortest paths with their total capacity reservation W shown in the first row of Table III. We use 64 random number seeds for generating flow sequences for backup path updates. For these 64 results, their maximum and minimum total spare capacities S and costs $COST$ are given in Table III. In Table III one can see that using different objectives in SSR produce different solutions. When the total spare capacity is the objective, all networks have lower total spare capacities but higher total costs which are calculated by using modular cost after solutions are found. When the total cost based on modular cost function is the objective function, all networks have lower total costs but higher total spare capacity. These results show that SSR achieves good redundancies when the optimization uses non-linear link cost function.

TABLE II
NETWORK INFORMATION

| Network | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|-----------|-----|------|------|------|-----|------|------|------|
| N | 10 | 12 | 13 | 17 | 18 | 23 | 26 | 50 |
| L | 22 | 25 | 23 | 31 | 27 | 33 | 30 | 82 |
| \bar{d} | 4.4 | 4.17 | 3.54 | 3.65 | 3 | 2.87 | 2.31 | 3.28 |
| R | 90 | 132 | 156 | 272 | 306 | 506 | 650 | 2450 |

V. FAILURE-DEPENDENT PATH RESTORATION

Failure-dependent path restoration has been discussed in several papers [9], [18], [7]. Iraschko, MacGregor and Grover [9] provided an integer programming path-flow formulation of this scheme and introduced the stub release function. A standard branch and bound (BB) algorithm was used to find solutions for networks whose candidate backups paths set was constrained by a hop count limit. The BB algorithm used is slow and does not scale to larger networks. Xiong and Mason [18] also gave a path-flow formulation for path restoration including both failure independent and failure dependent schemes. A heuristic algorithm was given based on routing algorithm and provided to minimize the cost of backup routes. A similar scheme is Kodilam's Sharing Partial Information (SPI) formulation in [6] which has also been used to route local restorable bandwidth guaranteed tunnels recently [7]. The link metrics used above for backup path routing/selection were based on residual link capacities or simply hop count. We have shown in [1] that these heuristic metrics are not enough to achieve near optimal spare capacity allocation in failure-independent path restoration.

In this section, we modify the model in Section II to incorporate failure-dependent path restoration. In this case, for each failure scenario k , there is a backup path link adjacent matrix $\mathbf{Q}^k = \{\mathbf{q}_r^k\}_{1 \times L} = \{\mathbf{q}_{r,l}^k\}_{R \times L}$. The backup paths must be feasible for each failure case and we replace flow conservation constraint (7) with (13) below.

$$\mathbf{q}_r^k \mathbf{B}^T = \mathbf{d}_r, \forall k, r, u_{rk} = 1, 1 \leq k \leq K, 1 \leq r \leq R \quad (13)$$

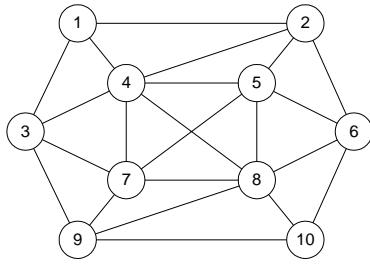


Fig. 5. Network 1 (10 nodes, 22 links)

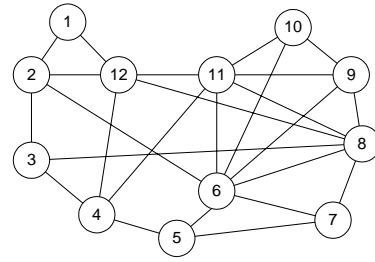


Fig. 6. Network 2 (12 nodes, 25 links)

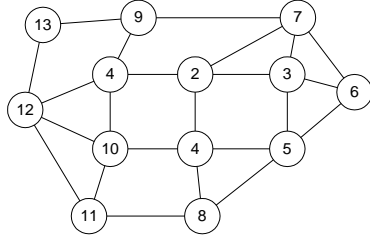


Fig. 7. Network 3 (13 nodes, 23 links)

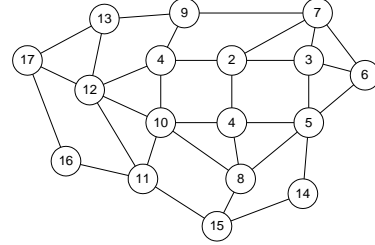


Fig. 8. Network 4 (17 nodes, 31 links)

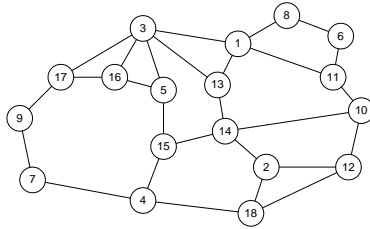


Fig. 9. Network 5 (18 nodes, 27 links)

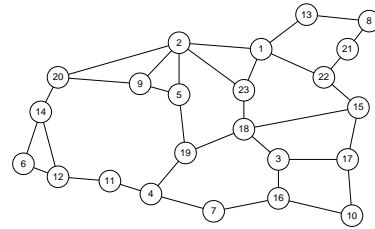


Fig. 10. Network 6 (23 nodes, 33 links)

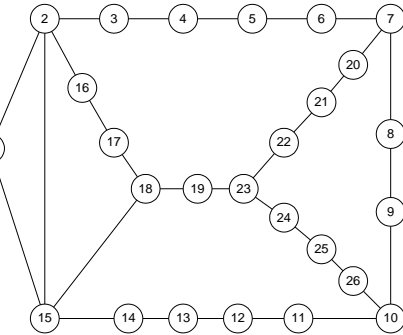


Fig. 11. Network 7 (26 nodes, 30 links)

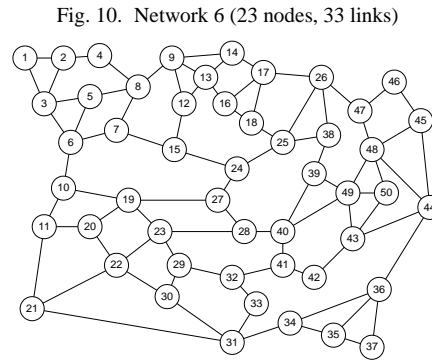


Fig. 12. Network 8 (50 nodes, 82 links)

TABLE III
COMPARISON OF SSR USING LINEAR AND MODULAR OBJECTIVES

| Network | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|-------------------------|--------|--------|--------|--------|--------|--------|---------|---------|
| W | 812 | 1206 | 1762 | 3542 | 4420 | 8924 | 14990 | 61188 |
| Minimize total capacity | | | | | | | | |
| S_{min} | 360 | 610 | 748 | 1478 | 2678 | 5886 | 10130 | 31172 |
| S_{max} | 426 | 696 | 848 | 1580 | 2760 | 6020 | 10158 | 31606 |
| $COST_{min}$ | 844.6 | 1355.0 | 1699.0 | 2671.6 | 4129.6 | 7157.0 | 10375.1 | 28665.3 |
| $COST_{max}$ | 1159.5 | 1798.0 | 2065.8 | 3224.6 | 4796.0 | 7611.9 | 10777.4 | 30676.6 |
| Minimize total cost | | | | | | | | |
| S_{min} | 418 | 714 | 888 | 1676 | 3020 | 6548 | 11188 | 36324 |
| S_{max} | 526 | 886 | 1174 | 2078 | 3368 | 7296 | 11674 | 39310 |
| $COST_{min}$ | 800.1 | 1199.0 | 1459.4 | 2515.6 | 3977.9 | 6853.7 | 10223.5 | 27365.7 |
| $COST_{max}$ | 1001.0 | 1477.7 | 1857.9 | 2925.1 | 4644.4 | 7915.1 | 10526.7 | 30373.3 |

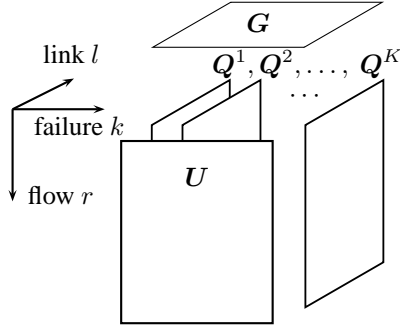


Fig. 13. Cubical structure for failure-dependent path restoration

The k -th column vector $\mathbf{G}_k = \{g_{lk}\}_{L \times 1}$ in \mathbf{G} is determined by \mathbf{U} 's k -th column vector $\mathbf{U}_k = \{u_{rk}\}_{R \times 1}$, \mathbf{M} , and \mathbf{Q}^k in (14) instead of (5). We use capital variable \mathbf{U}_k here for a column vector to distinguish it from the row vector u_r .

$$\mathbf{G}_k = \mathbf{Q}^{kT} \mathbf{M} \mathbf{U}_k, \quad 1 \leq k \leq K \quad (14)$$

This operation can be represented in a cubical structure shown in Fig. 13. When the number of traffic flows increases, the height of the cube increases. But the spare provision matrix \mathbf{G} remains the same size at $L \times K$. Hence, it remains scalable. Moreover, each traffic flow has its path information stored at a horizontal layer in the binary data cube. The flow contribution \mathbf{G}^r can be calculated in (15) in replacement of (9). \mathbf{G}_k^r is the k -th column vector in \mathbf{G}^r .

$$\mathbf{G}_k^r = m_r \mathbf{q}_r^{kT} u_{rk}, \quad 1 \leq k \leq K, 1 \leq r \leq R \quad (15)$$

The failure disjoint constraint (6) is changed to (16), where the tabu link vector \mathbf{t}_r^k is given in (17) in replacement of (2). These constraints ensure that the potential backup path will not use links affected by failure k which are marked as ones in \mathbf{f}_k , one of the row vectors in failure matrix \mathbf{F} .

$$\mathbf{t}_r^k + \mathbf{q}_r^k \leq 1, \quad \forall k, r, u_{rk} = 1, 1 \leq k \leq K, 1 \leq r \leq R \quad (16)$$

$$\mathbf{t}_r^k = \mathbf{f}_k, \quad \forall u_{rk} = 1 \quad (17)$$

The resulting arc-flow SCA model for failure-dependent path restoration has the objective (3) and constraints (4),(8), (13),(14), and (16). Note that constraint (14) can be replaced by (15) and (10).

In the SSR flow chart of Fig. 3, all \mathbf{q}_r and \mathbf{v}_r are modified to \mathbf{q}_r^k and \mathbf{v}_r^k accordingly. This SSR is used to find a backup path only when failure k disrupts flow r 's working path. This condition happens when $u_{rk} = 1$. The vector of link metrics \mathbf{v}_r^k is based on \mathbf{t}_r^k above and given in (18).

$$\mathbf{v}_r^k = \{v_{rl}^k\}_{L \times 1} = \phi(\mathbf{s}^+(e - \mathbf{t}_r^k)) - \phi(\mathbf{s}^-), \quad 1 \leq r \leq R, 1 \leq k \leq K, u_{rk} = 1 \quad (18)$$

Stub release

In the failure-dependent path restoration scheme, stub release allows the restoration process to release unused working path capacity when traffic flow is rerouted. Although it

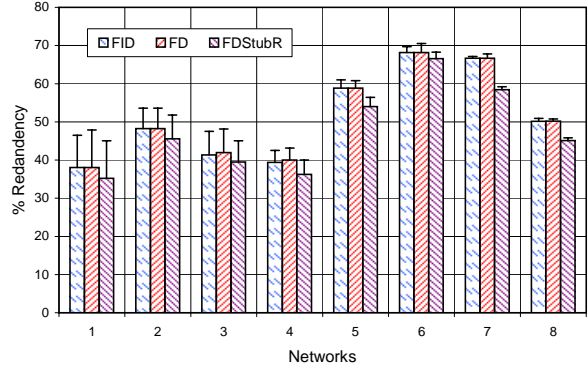


Fig. 14. Comparison of redundancy η over networks using different path restoration schemes.

has more signaling overhead, more reserved capacity will be freed and available for other backup paths after a failure.

In order to model the stub release, we define a new backup path matrix $\bar{\mathbf{Q}}^k$, which will replace the backup path matrix \mathbf{Q}^k used to find \mathbf{G} in (14). The matrix $\bar{\mathbf{Q}}^k$ is based on \mathbf{Q}^k obtained from the SSR algorithm for the failure-dependent path restoration case. The elements of $\bar{\mathbf{Q}}^k$ are given in (19). In this equation, the stub release function is represented by adding “-1” in appropriate positions in the backup paths matrix \mathbf{Q}^k . A working capacity of flow r on link l will be released only when its working path uses this link ($p_{rl} = 1$); current failure k disconnects its working path ($u_{rk} = 1$); and this failure does not affect this link ($f_{kl} \neq 1$). Since the working capacity released by the stub release function only depends on failures, there is no change to be made in the SSR algorithm given earlier for the FD scheme.

$$\bar{q}_{rl}^k = \begin{cases} q_{rl}^k - 1, & p_{rl} = 1, u_{rk} = 1, f_{kl} \neq 1 \\ q_{rl}^k, & \text{otherwise} \end{cases} \quad (19)$$

Comparison of path restoration schemes

The three path restoration schemes, i.e. failure-independent (FID), failure-dependent (FD), and failure-dependent with stub release function (FDStubR), were compared using SSR for the eight networks of Table II. The flows are a full mesh in the network with unit traffic load. Working paths are given as shortest paths with their total capacity reservation W shown in the first row. We still use 64 random number seeds for generating flow sequences for backup path updates and the SSR results have a range which is given as the maximum and minimum numbers. The ranges of total spare capacities (S_{min}, S_{max}) and the summation of CPU time (T_{sum}) to compute all these 64 cases are listed in Table IV. The corresponding redundancy range (η_{min}, η_{max}) for each scheme is also listed in Table IV and shown in Fig. 14 where the error bars mark the ranges.

From these results, FDStubR gives the lowest total spare capacities. Because the stub release function frees unused capacity of a disrupted working path for other backup paths. Note that the improvement in redundancy of FDStubR when compared with the much simpler FD is relatively small. Also, one can see that the solutions found by

TABLE IV
COMPARISON OF DIFFERENT RESTORATION SCHEMES USING SSR

| Network | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|--|-------|-------|-------|-------|-------|-------|-------|-------|
| W | 142 | 224 | 324 | 640 | 826 | 1670 | 2732 | 11104 |
| Failure-independent path restoration (FID) | | | | | | | | |
| S_{min} | 54 | 108 | 134 | 252 | 486 | 1138 | 1822 | 5568 |
| S_{max} | 66 | 120 | 154 | 272 | 504 | 1164 | 1834 | 5654 |
| T_{sum} | 0.5 | 1 | 1.2 | 3.2 | 3 | 6.5 | 7.3 | 191.8 |
| η_{min} | 38.03 | 48.21 | 41.36 | 39.38 | 58.84 | 68.14 | 66.69 | 50.14 |
| η_{max} | 46.48 | 53.57 | 47.53 | 42.50 | 61.02 | 69.70 | 67.13 | 50.92 |
| Failure-dependent path restoration (FD) | | | | | | | | |
| S_{min} | 54 | 108 | 136 | 256 | 486 | 1138 | 1822 | 5572 |
| S_{max} | 68 | 120 | 156 | 276 | 502 | 1178 | 1852 | 5638 |
| T_{sum} | 3.68 | 4.73 | 5.12 | 10.86 | 10.21 | 25.11 | 29.8 | 1111 |
| η_{min} | 38.03 | 48.21 | 41.98 | 40.00 | 58.84 | 68.14 | 66.69 | 50.18 |
| η_{max} | 47.89 | 53.57 | 48.15 | 43.13 | 60.77 | 70.54 | 67.79 | 50.77 |
| Failure-dependent path restoration with stub release (FDStubR) | | | | | | | | |
| S_{min} | 50 | 102 | 128 | 232 | 446 | 1112 | 1596 | 5006 |
| S_{max} | 64 | 116 | 146 | 256 | 466 | 1140 | 1618 | 5088 |
| T_{sum} | 3.67 | 5.15 | 5.71 | 10.67 | 10.85 | 26.2 | 33.02 | 1090 |
| η_{min} | 35.21 | 45.54 | 39.51 | 36.25 | 54.00 | 66.59 | 58.42 | 45.08 |
| η_{max} | 45.07 | 51.79 | 45.06 | 40.00 | 56.42 | 68.26 | 59.22 | 45.82 |

SSR in FD sometimes are worse than those of FID. Since SSR is an approximation algorithm which uses local optimum to approximate global optimal solutions, the larger search space in FD scheme might introduce more local optimal point to terminate the optimization process earlier than that in FID scheme. This justifies the above contradiction. However, since FID solutions are always a subset of FD solutions, we can use an FID solutions as the initial points to start SSR for FD scheme. This approach might find better solutions in FD. These results show that SSR can be easily extended to failure dependent path restoration schemes to further reduce network redundancy.

VI. SUMMARY

This paper presents a novel matrix-based approach for the spare capacity allocation problem. Following the matrix formulation an approximate solution technique termed successive survivable routing (SSR) is developed. The basic SSR algorithm is extended to capture arbitrary failure scenarios on directed networks; use non-linear link cost in the objective function and formulate failure-dependent path restoration and with stub release. Combined with the advantages of fast computation speed and near optimal solutions, SSR is a good and flexible algorithm to allocate spare capacity and backup paths on mesh networks.

ACKNOWLEDGMENT

Thanks to the anonymous reviewers for their valuable comments on ways to improve this paper.

REFERENCES

- [1] Y. Liu, D. Tipper, and P. Siripongwutikorn, "Approximating optimal spare capacity allocation by successive survivable routing," in *Proceeding of IEEE INFOCOM*, Anchorage, AL, April 24–28 2001, pp. 699–708.
- [2] J. Strand, A. L. Chiu, and R. Tkach, "Issues for routing in the optical layer," *IEEE Communications Magazine*, vol. 39, no. 2, pp. 81–87, Feb. 2001.
- [3] R. Doverspike and J. Yates, "Challenges for MPLS in optical network restoration," *IEEE Communications Magazine*, vol. 39, no. 2, pp. 89–96, Feb. 2001.
- [4] S. Chaudhuri, G. Hjalmtysson, and J. Yates, *Control of Lightpaths in an Optical Network*, Internet Engineering Task Force, Apr. 2000, draft-chaudhuri-ip-olxc-control-00.txt.
- [5] C. Dovrolis and P. Ramanathan, "Resource aggregation for fault tolerance in integrated service networks," *ACM Computer Communication Review*, vol. 28, no. 2, pp. 39–53, 1998.
- [6] M. Kodialam and T.V. Lakshman, "Dynamic routing of bandwidth guaranteed tunnels with restoration," in *Proceeding of IEEE INFOCOM*, Mar. 2000.
- [7] M. Kodialam and T.V. Lakshman, "Dynamic routing of locally restorable bandwidth guaranteed tunnels using aggregated link usage information," in *Proceeding of IEEE INFOCOM*, Apr. 2001.
- [8] M. Kodialam and T.V. Lakshman, "Integrated dynamic IP and wavelength routing in IP over WDM networks," in *Proceeding of IEEE INFOCOM*, Apr. 2001.
- [9] R. Iraschko, M. MacGregor, and W. Grover, "Optimal capacity placement for path restoration in STM or ATM mesh survivable networks," *IEEE/ACM Transactions on Networking*, vol. 6, no. 3, pp. 325–336, June 1998.
- [10] Y. Liu, *Spare capacity allocation method, analysis and algorithms*, Ph.D. dissertation, School of Information Sciences, University of Pittsburgh, 2001.
- [11] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms and Applications*, Prentice Hall, New York, 1993.
- [12] L. R. Foulds, *Graph Theory Applications*, Universitext. Springer-Verlag, 1992.
- [13] D. Medhi and D. Tipper, "Some approaches to solving a multi-hour broadband network capacity design problem with single-path routing," *Telecommunication Systems*, vol. 13, no. 2, pp. 269–291, 2000.
- [14] D. O. Awduche, L. Berger, D. Gan, T. Li, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP tunnels," Internet draft, Internet Engineering Task Force, Sept. 1999, draft-ietf-mpls-rsvp-lsp-tunnel-06.txt.
- [15] A. Al-Rumaih, D. Tipper, Y. Liu, and B. A. Norman, "Spare capacity planning for survivable mesh networks," in *Proceedings IFIP-TC6 Networking 2000*, Paris, France, May 2000 2000. Lecture Notes in Computer Science (LNCS) 1815, pp. 957–968, Springer-Verlag.
- [16] J. Doucette and W.D. Grover, "Influence of modularity and economy-of-scale effects on design of mesh-restorable DWDM networks," *IEEE Journal on Selected Areas of Communications*, vol. 18, no. 10, pp. 1912 – 1923, Oct. 2000.
- [17] E. McDysan and D. L. Spahn, *Hands-on ATM*, McGraw-Hill, 1998.
- [18] Y. Xiong and L. G. Mason, "Restoration strategies and spare capacity requirements in self-healing ATM networks," *IEEE/ACM Transactions on Networking*, vol. 7, no. 1, pp. 98–110, Feb. 1999.