Approximating Optimal Spare Capacity Allocation by Successive Survivable Routing

Yu Liu, *Member, IEEE*, David Tipper, *Senior Member, IEEE*, and Peerapon Siripongwutikorn, *Member, IEEE*

***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 the spare capacity allocation (SCA) problem. This paper unravels the SCA problem structure using a matrix-based model, and develops a fast and efficient approximation algorithm, termed successive survivable routing (SSR). First, per-flow spare capacity sharing is captured by a spare provision matrix (SPM) method. The SPM matrix has a dimension the number of failure scenarios by the number of links. It is used by each demand to route the backup path and share spare capacity with other backup paths. Next, based on a special link metric calculated from SPM, SSR iteratively routes/updates backup paths in order to minimize the cost of total spare capacity. A backup path can be further updated as long as it is not carrying any traffic. Furthermore, the SPM method and SSR algorithm are generalized from protecting all single link failures to any arbitrary link failures such as those generated by Shared Risk Link Groups or all single node failures. Numerical results comparing several SCA algorithms show that SSR has the best trade-off between solution optimality and computation speed.**

***Index Terms*—MPLS traffic engineering, multi-commodity flow, network planning and optimization, network survivability, protection and restoration, spare capacity allocation, survivable routing.**

I. Introduction

NETWORK survivability techniques have been proposed to guarantee seamless communication services in the face of network failures. Most of this work concentrates on various backbone transport networks, such as SONET/SDH, ATM, and WDM [1]. However, circuit-switched backbone networks are being replaced or overlapped with packet-switched networks which provide better manageability of bandwidth granularity and connection type using MPLS or GMPLS. This architecture migration has been significantly accelerated by the growth of the Internet. The increasing Internet traffic volume and its flexible QoS have made traditional service requirements of cost-effectiveness and survivability much more complex, especially in protocol scalability and prompt bandwidth provisioning for fluctuating traffic. Service survivability, as one of the critical requirements for backbone traffic, has become a focus for fast service provisioning. Therefore, it is of increasing importance and necessity for network survivability to catch up with this trend.

Traditionally, network survivability includes two components, survivable network design and restoration schemes. Survivable network design pre-plans the topology or virtual layout as well as the spare capacity reservation on network links for potential failures. The restoration scheme is in general distributed and provides fault detection, signalling and routing mechanisms to restore failed connections promptly. These two components are complementary to each other and cooperate to achieve seamless services upon failures. On a given two-connected mesh network, the *spare capacity allocation* (SCA) problem is to decide how much spare capacity should be reserved on links and where to route backup paths to protect given working paths from a set of failure scenarios. It is usually treated as a centralized problem.

The above network survivability framework with centralized design and distributed restoration has been challenged recently. Restoration schemes which allow distributed spare capacity reservation have been introduced for RSVP-based IntServ [2], and IP/MPLS [3], [4] recently. These schemes reserve shared spare capacity for all backup paths according to the current network status. Furthermore, they are fast enough to be used in a distributed protocol to reduce the setup response time of survivable service requests compared to slower centralized shared protection path design algorithms. In these schemes, while the network redundancy has been reduced to some degree by sharing spare capacity, the results in this paper show that the total spare capacity can be further reduced to near optimality by using the *successive survivable routing* (SSR) algorithm proposed here.

SSR routes backup paths sequentially by using shortest path algorithm on a set of link costs. These costs are calculated from a spare provision matrix (SPM) with $O(LK)$ complexity, where L is the number of network links and K is the number of failures to be protected. The SPM matrix keeps the minimum information which captures the essential structure of spare capacity sharing in the SCA problem. SSR is suitable to protect not only all single link failures, but also any arbitrary link failures, such as the Shared Risk Link Group (SRLG) concept [5], [6]. Numerical results on a set of sample networks comparing SSR with

Manuscript received December 28, 2002; revised February 1, 2004; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor C. Qiao. This work was supported by the National Science Foundation under Grant ANIR 9980516 and by the Defense Advanced Research Projects Agency under Grant F30602-97-1-0257. This paper extends two conference papers presented at the IEEE INFOCOM, Anchorage, AK, 2001, and the IEEE Global Communications Conference, San Antonio, TX, 2001.

Y. Liu is with OPNET Technologies, Cary, NC 27511 USA (e-mail: yliu@opnet.com).

D. Tipper is with the Department of Information Science and Telecommunications, University of Pittsburgh, Pittsburgh, PA 15260 USA (e-mail: tipper@tele.pitt.edu).

P. Siripongwutikorn is with the Computer Engineering Department, King Mongkut's University of Technology, Thonburi, Bangkok 10140, Thailand (e-mail: peerapon@cpe.kmutt.ac.th).

Digital Object Identifier 10.1109/TNET.2004.842220

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

TABLE II
ACRONYMS

SCA	Spare Capacity Allocation
SPM	Spare Provision Matrix
SSR	Successive Survivable Routing
SR	Survivable Routing
RAFT	Resource Aggregation for Fault Tolerance [2]
SPI	Sharing with Partial Information [3]
BB	Branch and Bound used in [63]
LP	Linear Programming relaxation
InP	Integer Programming
NS	No-Sharing scheme [3]
FMT	Fault Management Table [2]
FID	Failure-Independent path restoration
FD	Failure-Dependent path restoration

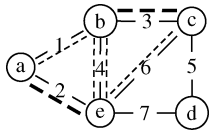

Fig. 1. Example five-node network, links are indexed numerically and nodes are indexed alphabetically.

other SCA algorithms, show that SSR has a near optimal spare capacity allocation with substantial advantages in computation speed.

The remainder of this paper is organized as follows. Section II gives a review on general network survivability techniques. Section III introduces the spare provision matrix based model of SCA. Section IV extends this model to directed networks and considers protection from a set of arbitrary failures. Then, the SSR algorithm is developed in Section V. Section VI gives the results of a numerical comparison between SSR and other algorithms for protecting single link failures. Section VII extends the matrix model and gives SSR numerical results for protecting all single node failures. Section VIII concludes the paper.

II. Background and Related Work

A. Traditional Network Survivability Techniques

Traditional network survivability techniques have two aspects, survivable network design and network restoration [7].

Survivable network design refers to the incorporation of survivability strategies into the network design phase in order to mitigate the impact of a set of specific failure scenarios. *Spare capacity allocation* is the major component in dimensioning a survivable network when the network topology and traffic demand are given. It ensures enough spare capacity for all demands on the physical or virtual network to recover from one of the failure scenarios via traffic rerouting. In this paper, we use a "traffic demand" same as a "flow".

For example, given a mesh SONET/SDH network topology, the demand with their traffic routed through given working path, the problems are how much spare capacity should be provisioned and where it should be located and shared by backup paths in order for the network to tolerate a specified set of failure scenarios (e.g., loss of any single link).

The term *mesh* does not imply that the network topology is a full mesh, but rather that the network is at least two-connected [8], [9]. The "two-connected" in this paper is equivalent to two-edge-connected when considering single link failures. There are at least two edge disjoint paths between any pair of nodes. For protecting single node failures, it requires two-node-connectivity.

Spare capacity sharing allows backup paths to share their capacity on their common links if their working paths are disjoint from the protected failures. This enables the minimization of the total spare capacity, but introduces a complicated combinatorial structure.

Example 1 – Spare Capacity Sharing: In the five-node network in Fig. 1, there are two working paths a-e and b-c (dashed lines), with their backup paths a-b-e and b-e-c (dotted lines) respectively. If single link failures are protected, the spare capacity on link 4 (b-e) can be shared by these two backup paths.

In *network restoration* phase, traffic demands interrupted by a failure are rerouted to their backup paths that have enough spare capacity provisioned in the design phase. Compared to dynamic fault-tolerant routing where no spare capacity is pre-allocated before failure, pre-planning and reserving enough spare capacity not only guarantees service restoration, but also reduces the duration and range of the failure impact. This is critical in backbone transport networks. In high speed packet-switched networks, such guarantee is also very important because the large backlog traffic accumulated during the failure restoration phase might introduce significant congestion [10], [11]. Pre-planning spare capacity can mitigate or even avoid this congestion.

Therefore, the recent interest in survivable network design has been concentrated on pre-planning cost-efficient spare capacity at a certain survivability level or restoration level. The *survivability level* gauges the percentage of restorable network

traffic upon a failure. In this paper, a 100% survivability level is always used. The partial survivability level can be dealt by a set of scale parameters on the backup path capacities.

The *network redundancy* is measured by the ratio of the total spare capacity over the total working capacity. In mesh-type networks, when working paths are the shortest hop paths, no less than 100% redundancy could be achieved when backup paths reserve dedicated bandwidth. However, the redundancy can be reduced by sharing spare capacity reservations among different backup paths. This scheme is called shared path restoration scheme. In the share path restoration cases of this paper, the redundancy can be as low as 35% to 70%.

A self healing ring (SHR) has 100% "redundancy" [12], [13]. This "redundancy" is allocated without the knowledge of traffic demands. It is different from the above definition. Since the real traffic on the ring might not take the shortest hop path, neither working nor spare capacity might be minimized. From the perspective of utilization, ring will never be better than mesh.

A *failure scenario* includes all simultaneously failed links or nodes that need to be protected. The failure scenarios where only one link can fail at a time are considered in Section III. Next, this assumption is then generalized to consider multiple *arbitrary failure* scenarios. Each of them includes multiple links or nodes. A concept, called *shared risk link group* (SRLG), supports the restoration of multiple component failures [5], [6]. The SCA problem for arbitrary failure addresses the design problem with consideration of SRLG.

A *node failure*, as a special arbitrary failure, is discussed in Section VII. Each node failure is transformed to include all links adjacent to this node. In the SCA for node failures, some demands with one-hop working paths will need link disjoint backup ones. The considered failure scenarios should consider all single link and node failures. In addition, a demand has to be protected from any single node failures excluding their source/destination nodes. Consequently, various demands will be resilient to different sets of failure scenarios.

Restoration schemes can be classified as either *link restoration* or *path restoration* according to the initialization locations of the rerouting process. In link restoration, the nodes adjacent to a failed link are responsible for rerouting all affected traffic demands. Thus it only patches around the failed link in original paths. In contrast, in path restoration, the end nodes whose traffic demands are traversing the failed link initiate the rerouting process. When the reserved spare capacity can be shared among different backup paths, it is called shared path/link restoration. In general, path restoration requires less total spare capacity reservation than link restoration scheme [14].

The selection of backup paths in path restoration can be *failure-dependent* (FD) when different failures are protected by different backup paths. Hence, the failure response depends on which failure scenario happens. On the contrary, a *failure-independent* (FID) path restoration scheme requires only one backup path to be *failure-disjoint* from the working path. The restoration does not need the knowledge of failure as long as this failure has been predicted and protected. These two schemes are also called the state-dependent and the state-independent path restoration in [15], [16]. The requirement of *failure-disjoint* guarantees backup and working paths will not be disrupted simultaneously by any single failure. For single link failures, the scheme is also called *path restoration with link-disjoint routes*. This failure-independent scheme requires less signaling support and is easier to implement at the trade-off of possibly more spare capacity than failure-dependent path restoration. An example for this scheme on an MPLS network is using a single backup Label Switched Path (LSP) to protect a working LSP and sharing reservation among different backup LSP's. Another MPLS implementation is using secondary explicit route (ER) of an LSP to protect its primary ER and subscribe enough TE bandwidth to be shared by secondary ER's. This paper concentrates on the failure-independent path restoration scheme. The extension for the failure-dependent scheme is in [17], [18].

A problem that arises in the failure-independent path restoration scheme is the existence of *trap topology* [19], [20]. In a trap topology, the working path may block all possible link-disjoint backup paths although the network topology is two-connected. For example on Network 6 in Fig. 11, when the traffic demand between nodes 13 and 15 has a working path routed via nodes 1 and 22, this path does not have any link-disjoint backup path available, although the network is two-connected.

There are two ways to avoid this dilemma. One way is to select multiple partially link-disjoint backup paths to protect different segments of the working path. However, the resulting path restoration scheme changes to be failure-dependent. The other way is to modify the working path to render a link-disjoint backup path possible. It is equivalent to routing working and backup paths interactively. The *augmenting path algorithm* for the max-flow problem [21] can be modified to serve this purpose. It routes each flow on a modified network with the same topology where all links have one-unit capacity and the traffic demand asks for two units. The algorithm can find two link-disjoint paths. The shorter one is for working and the other is for backup. Although this method introduces longer working paths, it is an effective method to keep failure-independent path restoration feasible. Thanks to the rare occurrence of the trap topology [19], the increased length on working path is negligible for overall network capacity. A similar trap topology issue for single node failures has been solved through a node split process [22]. Related modifications are discussed for various purposes [23]–[25]. For trap topology issues of arbitrary failures, some special cases have been discussed in [24]. Although several practical methods are available [26], [27], no general fast algorithm exists to assure the complete avoidance of this dilemma. It is a topic under study.

A closely related topic for the trap topology is the survivable topology design problems [28]. Great interests have been seen in multi-layer topology design and multicast tree protection recently. A logical topology design in multi-layer networks is modeled as an integer programming problem in [29]. It is generalized for arbitrary failures and represented in a matrix model in [17]. They considered the failure propagation effect where one lower layer failure will affect multiple upper layer failures in multi-layer networks. This topic has been discussed earlier in [30]–[32]. An algorithm to design redundant trees for single link failures is introduced in [33]. These results provide preliminary

foundations of spare capacity allocation for multicast traffic and on multi-layer networks.

B. SCA Algorithms

Previous research on spare capacity allocation of mesh-type networks adopts the problem context above and uses either mathematical programming techniques or heuristics to determine the spare capacity allocation as well as backup paths for all traffic demands. Multi-commodity flow models have been widely used to formulate spare capacity allocation problems in different networks like SONET/SDH [34]–[38], ATM [15], [37], WDM [39], [40], and IP/MPLS [41]. However, the resulting Integer Programming (InP) formulation was know as *NP-Hard* [3], [15], [37]. We further prove that SCA is *NP-Complete* in [17]. Due to the rapid increase of the solution space size with the network size, the optimal solution become unsolvable in polynomial time in many realistic networks. Thus fast heuristic methods are needed.

Relaxation methods are widely used to approximate InP solutions. Herzberg *et al.* [34] formulate a linear programming (LP) model and treat spare capacity as continuous variables. A rounding process is used afterward to obtain the final integer spare capacity solution which might not be feasible. They use hop-limited restoration routes to scale their LP problem. This technique is also used to input candidate paths into InP formulation when Branch and Bound (BB) is employed for searching the near optimal solution [35], [37]. Lagrangian relaxation with subgradient optimization is used by Medhi and Tipper [42]. The Lagrangian relaxation usually simplifies a hard original problem by dualizing the primal problem and decomposing its dual problem into multiple sub-problems easier to solve. Subgradient optimization is used to iteratively derive solutions between the primal and dual problems until the solution gap converges.

Genetic Algorithm (GA) based methods have been proposed for SCA as well [38], [42]–[44]. GA evolves the current population of "good solutions" toward the optimality by using carefully designed crossover and mutation operators. One advantage of GA approach is the ability to incorporate nonlinear functions into the algorithm, such as modular link cost. Additionally, the computation time can be easily controlled allowing the approach to scale to large networks. There are many other heuristic methods reported in the last decade, including Tabu Search [45], Simulated Annealing (SA) [39], Spare Link Placement Algorithm (SLPA) [37], Iterated Cutsets InP-based Heuristics (ICH) [46], Max-Latching Heuristics [37], Subset relaxation [47], and the column generation method [48]. Several reviews are given in [1], [33], [37].

All of the above methods are still in the pre-planning phase which can only be implemented centrally. A distributed scheme, Resource aggregation for fault tolerance (RAFT), is proposed by Dovrolis [2] for IntServ services using the resource ReSerVation Protocol (RSVP) [49]. Another routing based heuristic is given to pre-plan virtual connections on ATM networks in [15]. Each flow will route its backup path(s) individually. The link metric used in the routing algorithm is a heuristic value which has not considered the chance of sharing spare capacity. Two dynamic routing schemes with restoration, called *Sharing with Partial routing Information* (SPI) and *Sharing with Complete routing Information* (SCI) were introduced in [3]. In SPI, backup paths are routed by the shortest path algorithm while the spare resource reduction is approximated by using a heuristic link cost function. SPI is simple and fast, but as shown in our numerical results, the redundancy that SPI achieves is not very close to the optimal solutions. The SCI scheme is similar to the survivable routing (SR) scheme in this paper. However, it is claimed that the per-flow based information is necessary for SCI, unlike the SR scheme here.

Recently, implementations of shared path protection scheme have been seen in [50] and [51].

C. SCA Structure

The structure of the SCA problem has been investigated along with the algorithm discoveries. The max-latching hypothesis was introduced in [37] and [52] to speed up the heuristic for spare capacity design with span restoration. The span restoration is used in SONET/SDH networks to recover any single span failure. It is equivalent to the link restoration in this paper. A square matrix structure is first introduced with elements given the spare capacity requirement on one span when the other span fails. It is also called the *forcer* relationship since a span was forced to provide enough spare capacity due to multiple other span failures. This concept is further used in [53] to solve the express route planning problem in span (link) restorable networks. The breakpoint to reduce the total spare capacity is to break these "forcer" links and reroute the flows over them. The relationship focused on pair-wise link relationships in link restoration. The matrix method in this paper extends this concept to consider the spare capacity sharing relationship among different demands using path restoration. The *channel dependency graph* in [54] shows the dependency relations between links on working and backup paths in a dual graph. It provides an important hint for the SCA structure.

The *fault management table* (FMT) method is the building structure in the resource aggregation fault tolerant (RAFT) scheme [2]. It provides a local data matrix to store the spare capacity sharing information among different flows. It is very difficult to share the FMT information globally since it is per-flow based and hence, not scalable with the network size and the number of flows. An equivalent mathematical formulation of FMT is given in [42].

A two-dimensional array between failed links and links with spare capacity is used by Cwilich *et al.* to build a routing based algorithm called "LOCAL" [55]. The method of finding spare capacity is also specified using this array. Then, the LOCAL algorithm uses part of the information to build routing metrics to route backup paths. Recently, similar two-dimensional relationships have also been used for several routing based algorithms in [3], [4], [56]–[58]. These papers still concentrate on single link failures. The spare capacity sharing structure for general failure cases is given in this paper.

III. A Spare Provision Matrix Based SCA Model

In this section, the spare capacity allocation (SCA) problem is targeted to protect any single link failure using

failure-independent (FID) path restoration. It is also called *path restoration with link-disjoint routes*, where a backup path is always link-disjoint from its working path. The SCA objective here is to minimize the total spare capacity when all traffic demands require a 100% survivability or restoration level.

A network is represented by an undirected graph with N nodes, L links and R flows. The physical link capacity is assumed unlimited in this paper. This assumption simplifies the SCA problem and allow us to concentrate on its essential characters. Note that the model and algorithm can be generalized to incorporate capacitated links by adding constraints or using nonlinear link cost functions.

A set of matrix-based definitions and an optimization model are given first. An example is shown in Figs. 1 and 2.

A flow r, $1 \le r \le R$ is specified by its origin/destination node pair $(o(r), d(r))$ and bandwidth m_r. Working and backup paths of flow r are represented by two $1 \times L$ binary row vectors $\boldsymbol{p}_r = \{p_{rl}\}$ and $\boldsymbol{q}_r = \{q_{rl}\}$ 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 stacks of these row vectors, forming two $R \times L$ matrices $\boldsymbol{P} = \{p_{rl}\}$ and $\boldsymbol{Q} = \{q_{rl}\}$ respectively. Let $\boldsymbol{M} = \mathrm{Diag}(\{m_r\}_{R\times 1})$ denote the diagonal matrix representing the bandwidth units of all flows. Note that if the protection level of flows is under/above 100%, the elements in $\boldsymbol{M}$ can be adjusted by a set of scale parameters to reserve partial/additional spare capacities on backup paths.

The undirected network topology is represented by the node link incidence matrix $\boldsymbol{B} = (b_{nl})_{N\times L}$ where $b_{nl} = 1$ if and only if node n is the origin or destination of link l. The flow node matrix is $\boldsymbol{D} = (d_{rn})_{R\times N}$, where $d_{rn} = 1$ iff $o(r) = n$ or $d(r) = n$. In undirected networks, both $\boldsymbol{B}$ and $\boldsymbol{D}$ are binary matrices. In directed networks used in the next section, they are generalized to use "-1" to mark the destination node of a link or a flow, same as the notations in the graph theory [8], [9].

We let $\boldsymbol{G} = \{g_{lk}\}_{L\times K}$ denote the *spare provision matrix* whose elements g_{lk} are the minimum spare capacity required on link l when link k fails. Note that $K = L$ when protecting all single link failures. Given the backup paths $\boldsymbol{Q}$, demand bandwidth matrix $\boldsymbol{M}$, and working path $\boldsymbol{P}$, the spare provision matrix can be determined as in (3). The minimum spare capacity required on each link is denoted by the column vector $\boldsymbol{s} = \{s_l\}_{L\times 1}$ which is found in (2). The function $\max$ in (2) asserts that an element in $\boldsymbol{s}$ is equal to the maximum element in the corresponding row of $\boldsymbol{G}$. It is equivalent to $\boldsymbol{s} \ge \boldsymbol{G}$ in this optimization model, where the operator $\ge$ between a column vector $\boldsymbol{s}$ and a matrix $\boldsymbol{G}$ guarantees that any element in $\boldsymbol{s}$ is always not less than any elements in the corresponding row of $\boldsymbol{G}$. 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 ϕ_l denote the cost function of spare capacity on link l. $\boldsymbol{\phi} = \{\phi_l\}_{L\times 1}$ is a column vector of these cost functions and $\boldsymbol{\phi}(\boldsymbol{s})$ gives the cost vector of the spare capacities on all links. The total cost of spare capacity on the network is $\boldsymbol{e}^T\boldsymbol{\phi}(\boldsymbol{s})$, where $\boldsymbol{e}$ is unit column vector of size L. For simplicity, in this section, we assume all cost functions are identity functions. Then the objective of SCA is reduced to minimize the total spare capacity in

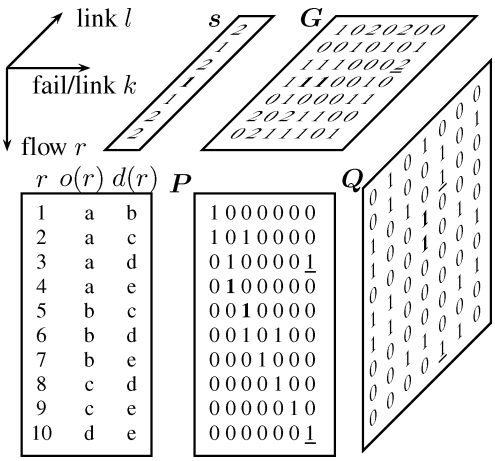

Fig. 2. SCA structure for failure-independent path restoration on the five-node network in Fig. 1.

(1). Given the notation and definitions (summarized in Table I and Table II), the spare capacity allocation problem can be formulated as follows.

$$\min_{\boldsymbol{Q},\boldsymbol{s}} \quad \boldsymbol{e}^T\boldsymbol{s} \tag{1}$$

$$\text{s.t.} \quad \boldsymbol{s} = \max \boldsymbol{G} \tag{2}$$

$$\boldsymbol{G} = \boldsymbol{Q}^T\boldsymbol{M}\boldsymbol{P} \tag{3}$$

$$\boldsymbol{P} + \boldsymbol{Q} \le 1 \tag{4}$$

$$\boldsymbol{Q}\boldsymbol{B}^T = \boldsymbol{D} (\mathrm{mod}\ 2) \tag{5}$$

$$\boldsymbol{Q} : binary. \tag{6}$$

The objective function in (1) is to minimize the total spare capacity by selection of the backup paths and spare capacity allocation. Constraints (2) and (3) calculate the spare capacity vector $\boldsymbol{s}$ and the spare provision matrix $\boldsymbol{G}$. Constraint (4) guarantees that each backup path is link-disjoint from its working path. Flow conservation constraint (5) guarantees that backup paths given in $\boldsymbol{Q}$ are feasible paths of flows in an *undirected* network. This type of constraint is also called the mass balance constraint [21]. Only source and destination nodes of a flow have nonzero traffic accumulation, while its intermediate nodes only allow traffic passing.

The above optimization model in (1)–(6) is an *arc-flow* multicommodity flow model. It is *NP-complete* [17]. Compared with the *path-flow* formulations in [35], [42], this arc-flow model needs additional constraints to find feasible backup paths, but pre-calculated backup path sets are not necessary. These differences make it solvable using iterative backup path routing.

Example 2 – Matrix Method: In the five-node undirected network in Fig. 1, the network load is a full mesh of symmetrical unit-bandwidth traffic demands. Their indices, source and destination node are listed in the left bottom table in Fig. 2. Their bandwidth matrix forms $\boldsymbol{M} = \boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix of size R. Shortest hop routes are used for the working paths $\boldsymbol{P}$. We consider the case of any single link failure. Currently, the backup paths are assumed found in $\boldsymbol{Q}$. The spare provision matrix is given as $\boldsymbol{G} = \boldsymbol{Q}^T\boldsymbol{P}$. These matrices are shown in Fig. 2. Their indexing variables, link l, failed link k, and flow r, are ordered as given in the top left corner of the figure.

First let us revisit Example 1, two flows with working/backup paths, a-e/a-b-e and b-c/b-e-c, are recorded in the rows 4 and 5 of $\boldsymbol{P}$ and $\boldsymbol{Q}$. One unit spare capacity on link 4 is shared by the two backup paths, as shown by the elements " **1** " for g_{42} and g_{43} in $\boldsymbol{G}$, as marked in bold font. Moreover, we can find that two more backup paths can share this unit of spare capacity. They are flow 1 (a-b/a-e-b) and flow 9 (c-e/c-b-e). These spare capacity sharing is clearly captured by g_{41} and g_{46}.

Furthermore, the total spare capacity reservation is given by the summation of all elements in vector $\boldsymbol{s}$ as $S = \boldsymbol{e}^T\boldsymbol{s} = 11$. The element g_{57} of $\boldsymbol{G}$ equals to "$\underline{2}$", listed in seventh column and the fifth row from the bottom, marked by an underline. It presents the spare capacity required on link 5 when link 7 fails. In this case, flow 3 and 10 will be affected as shown by "$\underline{1}$" in column 7 of $\boldsymbol{P}$. The backup paths of flow 3 and 10 both use link 5 as shown in the fifth column of $\boldsymbol{Q}$. Then the total bandwidth of these two flows is the required bandwidth in g_{57}.

IV. Model for Arbitrary Failures and Link Cost

In this section, we generalize the matrix SCA model to protect any arbitrary failures. An arbitrary failure scenario includes all simultaneously failed links or nodes that need to be protected. For a failed node, all its adjacent links are marked as fail instead.

We characterize K failure scenarios in a binary matrix $\boldsymbol{F} = \{\boldsymbol{f}_k\}_{K\times 1} = \{f_{kl}\}_{K\times L}$. The row vector $\boldsymbol{f}_k$ in $\boldsymbol{F}$ is for failure scenario k and its element f_{kl} equals one iff link l fails in scenario k. In this way, each failure scenario includes a set of links that will fail simultaneously in this scenario. We also denote a flow failure incidence matrix $\boldsymbol{U} = \{\boldsymbol{u}_r\}_{R\times 1} = \{u_{rk}\}_{R\times K}$, where $u_{rk} = 1$ iff flow r will be affected by failure k, and $u_{rk} = 0$ otherwise. A flow tabu-link matrix $\boldsymbol{T} = \{\boldsymbol{t}_r\}_{R\times 1} = \{t_{rl}\}_{R\times L}$ has $t_{rl} = 1$ iff the backup path of flow r could not use link l, and $t_{rl} = 0$ otherwise. We can find $\boldsymbol{U}$ and $\boldsymbol{T}$ given $\boldsymbol{P}$ and $\boldsymbol{F}$ as shown in (7) and (8) respectively. Note that, a binary matrix multiplication operation "$\odot$" is used in these two equations. It modifies the general addition in $1 + 1 = 2$ to Boolean addition in $1 + 1 = 1$ [59]. Using this binary operator, the complicated logical relations among link, path and failure scenarios are simplified into two matrix operations.

$$\boldsymbol{U} = \boldsymbol{P} \odot \boldsymbol{F}^T \tag{7}$$

$$\boldsymbol{T} = \boldsymbol{U} \odot \boldsymbol{F}. \tag{8}$$

The *spare provision matrix* $\boldsymbol{G} = \{g_{lk}\}_{L\times K}$ is given in (11). Its element g_{lk} gives the minimum spare capacity required on link l when failure k happens. The minimum spare capacities required on links are given by the column vector $\boldsymbol{s} = \{s_l\}_{L\times 1}$ in (10), which is equivalent as (2) as explained earlier.

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

$$\min_{\boldsymbol{Q},\boldsymbol{s}} \quad \boldsymbol{e}^T\boldsymbol{\phi(s)} \tag{9}$$

$$\text{s.t.} \quad \boldsymbol{s} \geq \boldsymbol{G} \tag{10}$$

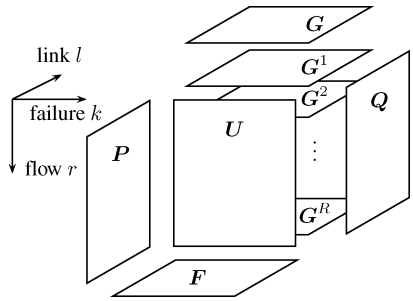

Fig. 3. SCA structure for protecting arbitrary failures.

$$\boldsymbol{G} = \boldsymbol{Q}^T\boldsymbol{M}\boldsymbol{U} \tag{11}$$

$$\boldsymbol{T} + \boldsymbol{Q} \leq 1 \tag{12}$$

$$\boldsymbol{Q}\boldsymbol{B}^T = \boldsymbol{D} \tag{13}$$

$$\boldsymbol{Q} : \textit{binary}. \tag{14}$$

The objective function in (9) 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 nonlinear link cost functions. Constraints (10) and (11) calculate $\boldsymbol{s}$ and $\boldsymbol{G}$. Constraint (12) guarantees that backup paths will not use any link which might fail simultaneously with their working paths. Flow conservation constraint (13) guarantees that backup paths given in $\boldsymbol{Q}$ are feasible paths of flows in a *directed* network. This constraint is different from (5) for undirected networks, because the node link matrix $\boldsymbol{B}$ and flow node matrix $\boldsymbol{D}$ are not binary matrices anymore. The topology is given by the node-link incidence matrix $\boldsymbol{B} = (b_{nl})_{N\times L}$ where $b_{nl} = 1$ *or* -1 if and only if node n is the origin *or* destination node of link l. $\boldsymbol{D} = (d_{rn})_{R\times N}$ is the flow node incidence matrix where $d_{rn} = 1$ or -1 iff $o(r) = n$ or $d(r) = n$. $\boldsymbol{D}$ can be further separated by two binary matrices $\boldsymbol{D}^o$ and $\boldsymbol{D}^d$ to indicate the source and destination nodes respectively: $d^o_{rn} = 1$ ($d^d_{rn} = 1$) iff $o(r) = n$ ($d(r) = n$). It gives $\boldsymbol{D}^o - \boldsymbol{D}^d = \boldsymbol{D}$. These two binary matrices will be used in Section VII for node failures.

The above matrix-based SCA model for protecting any arbitrary failure scenarios, has the problem structure illustrated in Fig. 3. Pre-calculated flow failure incidence matrix $\boldsymbol{U}$, instead of working path matrix $\boldsymbol{P}$, is used to calculate spare provision matrix $\boldsymbol{G}$ in (11).

In the discussion above, the spare provision matrix $\boldsymbol{G}$ plays a critical role in the SCA problem. Another way to compute $\boldsymbol{G}$ is through the aggregation of per-flow based information of working and backup paths. First, the contribution of a single traffic demand r to $\boldsymbol{G}$ is given by $\boldsymbol{G}^r = \{g^r_{lk}\}_{L\times K}$ in (15), where $\boldsymbol{u}_r$ and $\boldsymbol{q}_r$ are the rth row vectors in $\boldsymbol{U}$ and $\boldsymbol{Q}$. The spare provision matrix $\boldsymbol{G}$, thus, is calculated in (16). This structure is sketched in Fig. 3.

$$\boldsymbol{G}^r = m_r(\boldsymbol{q}_r^T\boldsymbol{u}_r), \quad \forall r, 1 \leq r \leq R \tag{15}$$

$$\boldsymbol{G} = \sum_{r=1}^{R} \boldsymbol{G}^r. \tag{16}$$

Using above matrices, per-flow based information in $\boldsymbol{Q}$ is replaced by $\boldsymbol{G}$ as the stored network state information for spare capacity sharing. 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 and makes it possible for distributed implementation. Moreover, this spare provision matrix has other good properties, such as privacy and transparency among traffic demands, in an open network environment.

V. Successive Survivable Routing

The successive survivable routing (SSR) algorithm is a heuristic algorithm to solve the SCA problem. In the SCA problem, the working paths are given. The backup paths need to be found to protect their working paths. The spare capacity reserved by these backup paths are shared in order to minimize the total cost of the spare capacity.

Under this problem definition, SSR solves the original multi-commodity flow problem by partitioning it into a sequence of single flow problems. Using a random order of flows, SSR finds backup paths one by one. Since different random orders might produce different solutions, the best results among multiple cases with different random orders will be selected as the approximation solution of the optimization problem. For each flow within a random case, SSR routes its backup path using shortest path algorithm, with a set of special link metrics that are calculated as the cost of incremental spare capacity.

In Fig. 4, a flow chart of the SSR implementation at the source node $o(r)$ of a flow r is given.

Step 1 initiates SSR for flow r with its working path $\boldsymbol{p}_r$ and the failure matrix $\boldsymbol{F}$. Then, $\boldsymbol{u}_r$ and $\boldsymbol{t}_r$, which are the rows for flow r in $\boldsymbol{U}$ and $\boldsymbol{T}$, are calculated in (7) and (8).

Step 2 periodically collects current network state information. These information includes the spare provision matrix $\boldsymbol{G}$. 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 $\boldsymbol{G}$ should be long enough to guarantee the stability of the algorithm. The discussion of how to keep $\boldsymbol{G}$ synchronized is discussed after we finish the introduction of the SSR algorithm.

In Step 3, the vector of link metrics $\boldsymbol{v}_r$ used for the shortest path algorithm is first calculated as follows:

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

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

(c) Let $\boldsymbol{q}_r^* = \boldsymbol{e} - \boldsymbol{t}_r$, which assumes the backup path is using all nontabu links. Then, we can find the vector of *link metrics* for flow r as

$$\begin{aligned} \boldsymbol{v}_r &= \{v_{rl}\}_{L\times 1} \\ &= \boldsymbol{\phi}(\boldsymbol{s}^*(\boldsymbol{e} - \boldsymbol{t}_r)) - \boldsymbol{\phi}(\boldsymbol{s}^{-r}), \end{aligned} \tag{17}$$

where $\boldsymbol{t}_r$ is the binary flow tabu-link vector of flow r. The element v_{rl} is the cost of the incremental spare capacity on link l if this link is used on the backup path.

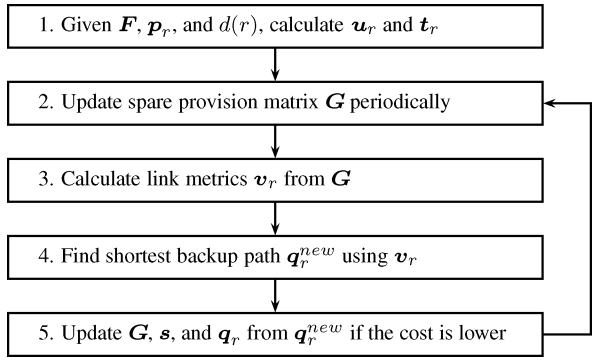

Fig. 4. SSR Flow chart at the source node of flow r.

After given the vector of link metrics, Step 4 first excludes all the tabu links marked in $\boldsymbol{t}_r$, then uses a shortest path algorithm with link metrics $\boldsymbol{v}_r$ to find an updated backup path $\boldsymbol{q}_r^{new}$.

In Step 5, the original backup path $\boldsymbol{q}_r$ is replaced by the new path $\boldsymbol{q}_r^{new}$ when it has a lower path cost based on the link metrics $\boldsymbol{v}_r$:

$$\boldsymbol{q}_r = \boldsymbol{q}_r^{new}, \text{ when } \boldsymbol{v}_r^T \boldsymbol{q}_r > \boldsymbol{v}_r^T \boldsymbol{q}_r^{new}.$$

Then the spare provision matrix $\boldsymbol{G}$ and the spare capacity vector $\boldsymbol{s}$ are updated to reflect this change 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. This will 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 IETF RFC 3209 [60]. In the off-line centralized implementation, the tear-down and setup of the backup paths might be postponed until the final backup paths are determined.

The objective here is not only to route an eligible backup path, but also to minimum total cost and eventually pre-plan spare capacity and provision survivable services. Hence, we call this backup path finding process *survivable routing* (SR).

After Step 5, SSR will continue to Step 2 to start the next backup path update for another flow.

This iterative process keeps improving the total cost of spare capacity. Thus the algorithm is called *successive survivable routing* (SSR).

A termination condition after Step 5 can be added as an option to decide whether to stop the algorithm. If there is no backup path update or a threshold number of backup path updates is reached, the algorithm will stop. Otherwise, The algorithm continues to update backup paths for the changing network status. Because the above iteration keeps reducing the objective function, SSR can converges quickly on a stable network. This fast convergence has been shown in the numerical results next.

Example 3 – Find a Backup Path in SSR: The Example 2 in Fig. 1 is used here to illustrate how a backup path is found to protect single link failures. The objective is to minimize total spare capacity. The current network already has 10 flows. Their working and backup paths are shown in Fig. 2. The current spare provision matrix $\boldsymbol{G}$ is also shown in Fig. 5.

Assume a new flow 11 from node a to b requires one unit demand. The shortest hop path a-b is the working path. To protect

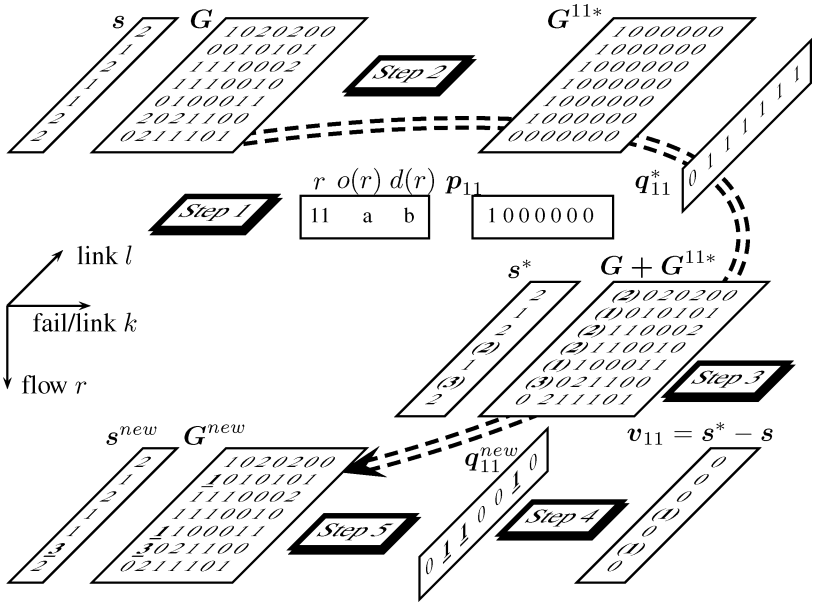

Fig. 5. Find a backup path of flow 11 using successive survivable routing algorithm on the five-node network in Figs. 1 and 2.

single link failures, we have $\boldsymbol{t}_{11} = \boldsymbol{u}_{11} = \boldsymbol{p}_{11}$ in Step 1. In Step 2, we assume current $\boldsymbol{G}$ for 10 flows is shown in Fig. 2. Since flow 11 does not have a backup path, Step 3(a) is omitted. Next, a "fake" backup path vector $\boldsymbol{q}_{11}^*$ is used to generate the vector $\boldsymbol{s}^*$ in Step 3(b) and 3(c). The difference between a pair of corresponding elements of $\boldsymbol{s}$ and $\boldsymbol{s}^*$ shows the additional spare capacity required if this link is used in the backup path. $\boldsymbol{v}_{11}$ records this difference. In this case, it is called the incremental spare capacity vector.

In Step 4, the objective to minimize total spare capacity on the network has been partitioned to minimization of the additional spare capacity used for each backup path. We need to find a path which requires minimum additional spare capacity. Hence, elements in $\boldsymbol{v}_{11}$ are used as the link metrics in the shortest path algorithm to find a new backup path. In addition, the tabu links of this flow is removed. Then the new backup path vector $\boldsymbol{q}_{11}^{new}$ is found on link 2-6-3, or nodes a-e-c-b. This backup path is not the shortest hop path. However, it requires minimum additional spare capacity, i.e., one unit. This helps to minimize total spare capacity through spare capacity sharing. A new spare provision matrix $\boldsymbol{G}^{new}$ and a new spare capacity vector $\boldsymbol{s}^{new}$ are updated in Step 5. They will be used to find or update other backup paths.

Synchronization of the Spare Provision Matrix

Keeping $\boldsymbol{G}$ up-to-date is important for the efficiency of a distributed protocol as introduced in Step 2. There are two methods for collecting $\boldsymbol{G}$ over the network.

The first one is link based. The l-th row vector of $\boldsymbol{G}$, $\boldsymbol{g}_l$, is given in (18), $\boldsymbol{Q}_l$ is the l-th column vector in the backup path matrix $\boldsymbol{Q}$. It is stored at the source node n, $b_{nl} = 1$ of link l. It represents the required spare capacities for different failure scenarios on this link. The maximum element of this vector is the required spare capacity $s_l = \max \boldsymbol{g}_l$. This operation requires the working path information to be included in its backup path reservation.

$$\boldsymbol{g}_l = \boldsymbol{Q}_l^T \boldsymbol{M} \boldsymbol{U}. \tag{18}$$

Once all these row vectors are up-to-date, a source node n will collect all the row vectors in a vector set $\{\boldsymbol{g}_l : \forall l, b_{nl} = 1\}$, and exchange it with other nodes through an information synchronization process as those in link state routing protocols. Each advertised packet will have a size of at most $N \times K$. In this way, the spare provision matrix $\boldsymbol{G}$ can be distributively calculated over the network.

The second method is node based. Since a node n has the working and backup path information of all flows it originated, it is easy to find a partial spare provision matrix $\boldsymbol{G}(n)$ to include all the contributions $\boldsymbol{G}^r$ of its originated flows $\forall r, d_{rn} = 1$ as given in (19). Then this node disseminates this parial information $\boldsymbol{G}(n)$ through information advertisement 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 include working paths with their backup path reservations. Hence it uses less signaling support.

$$\boldsymbol{G}(n) = \sum_{1 \le r \le R, d(rn)=1} \boldsymbol{G}^r. \tag{19}$$

Both methods can synchronize the spare provision matrix $\boldsymbol{G}$ 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 $\boldsymbol{G}$ 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 centralize 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 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 scale. Each flow can find a backup path first, then update it later to reduce the total cost of spare capacity. Note that longer timing requirement will further alleviate the scalability problem of the

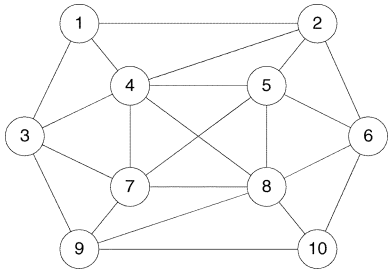

Fig. 6. Network 1 ($N = 10$, $L = 22$).

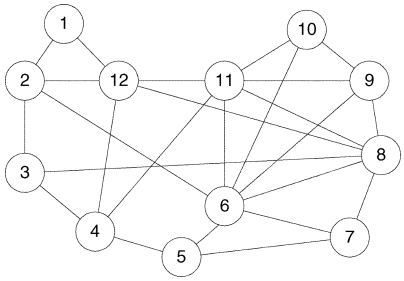

Fig. 7. Network 2 ($N = 12$, $L = 25$).

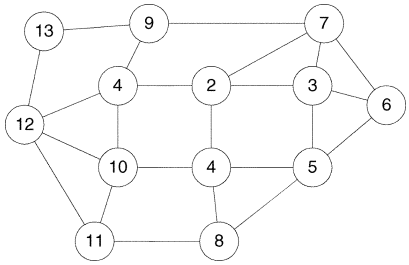

Fig. 8. Network 3 ($N = 13$, $L = 23$).

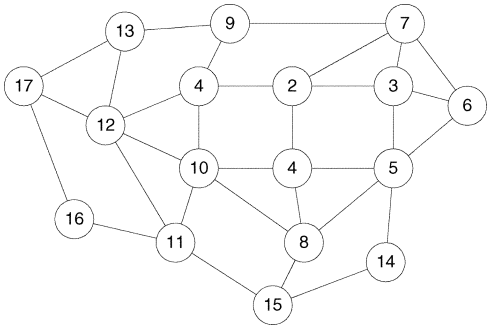

Fig. 9. Network 4 ($N = 17$, $L = 31$).

above state information synchronization process. This topic is important and requires further study [61].

VI. Numerical Results for Link Failures

Eight network topologies shown in Figs. 6–13 are used to assess the proposed SSR algorithm. The networks have average node degrees $\bar{d}$ ranged from 2.31 to 4.4 as given in Table III. Without loss of generality, we assume symmetrical traffic demands between any node pairs. All flows have one unit bandwidth demand, i.e., $m_r = 1, \forall r, 1 \leq r \leq R$. For Network 3 and 5, We also provide results when demands are varied between one and five units in cases 3b and 5b. The objective of SCA is to minimize the total spare capacity as

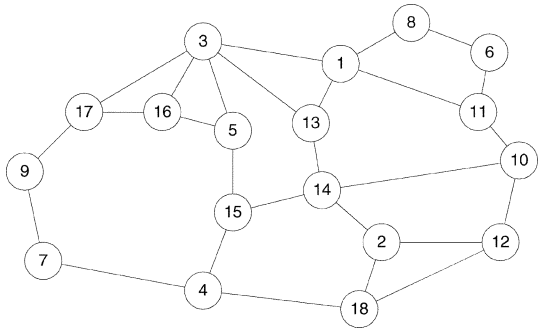

Fig. 10. Network 5 ($N = 18$, $L = 27$).

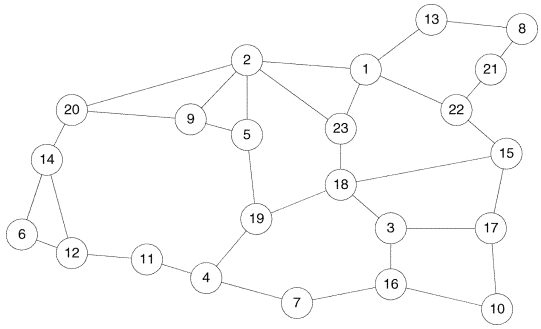

Fig. 11. Network 6 ($N = 23$, $L = 33$).

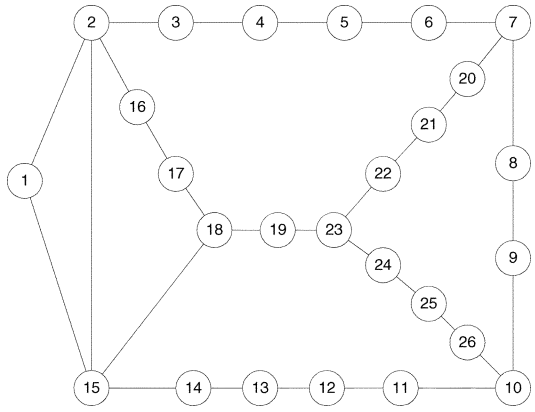

Fig. 12. Network 7 ($N = 26$, $L = 30$).

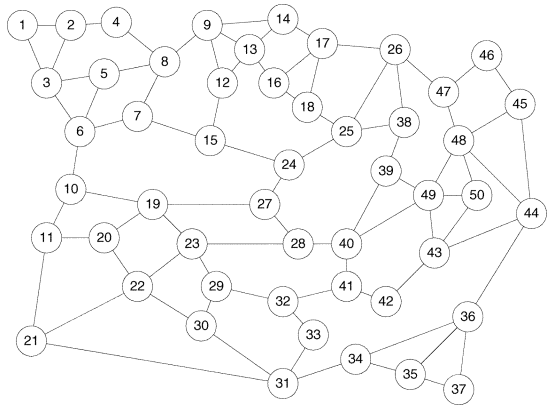

Fig. 13. Network 8 ($N = 50$, $L = 82$).

TABLE III
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	3.00	2.87	2.31	3.28
R	90	132	156	272	306	506	650	2450

Note: the number of link L here is the number of undirected links. Same definition applies to Fig.6–13.

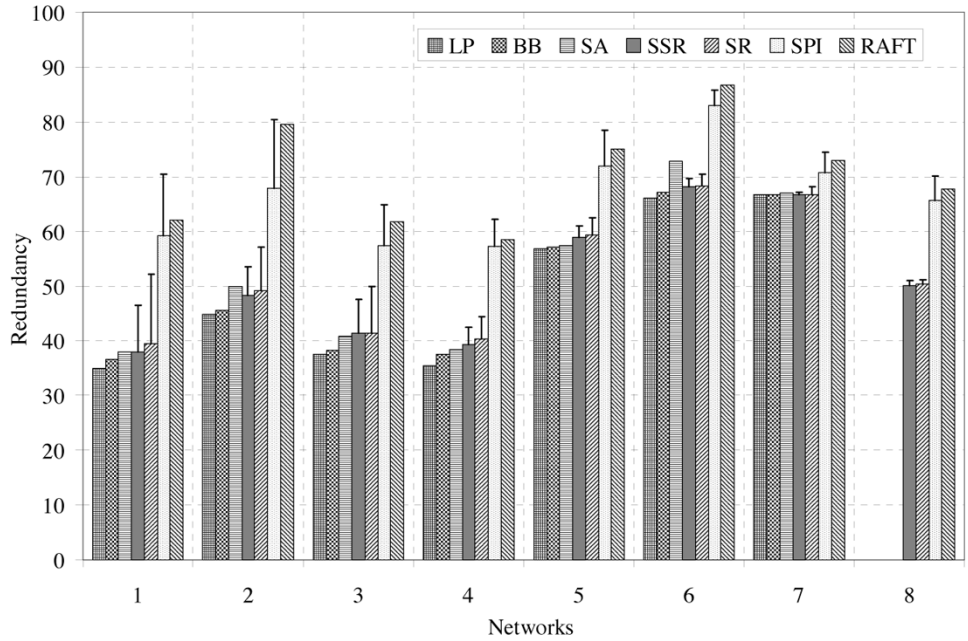

Fig. 14. Comparison of redundancy $\eta = S/W$ over networks for single link failures. The error bars on SSR, SR and SPI give the ranges of 64 results from random flow sequences.

shown in (1). These assumptions are selected for the ease of comparison among networks.

The total spare capacities and their total CPU times are given in Table IV. The network redundancies are plotted in Fig. 14. For Network 3, the redundancy versus time is plotted in Fig. 15 as an example to show the trade-off between time and optimality of SSR solutions. We conclude our results as follows.

SSR Finds Near Optimal Solutions The achieved redundancies from all algorithms can be roughly ordered as:

$$\text{LP} \le \text{optimal solutions} = \text{BB} \le \text{SA} \le \text{SSR} \le \text{SR} \ll \text{SPI} \approx \text{RAFT} \ll \text{NS}.$$

The optimal solutions are given by Branch and Bound (BB) and lower-bounded by Linear Programming relaxation (LP). Their gaps are very narrow. Simulated Annealing (SA) provides good approximation to the optimal solutions with a longer execution time. At the other extreme, NS does not provide spare capacity sharing. Consequently it gives the highest redundancies, which is above 100%. There are small gaps in redundancies between BB and SSR. They are less than 4%. Hence, SSR has achieved solutions very close to optimal ones.

SSR is Fast and Scales The computation times for these algorithms are significantly different. BB takes tens of minutes to hours and it cannot scale to larger networks, such as Network 8. SA is faster in speed than BB, but it still needs parameter tuning

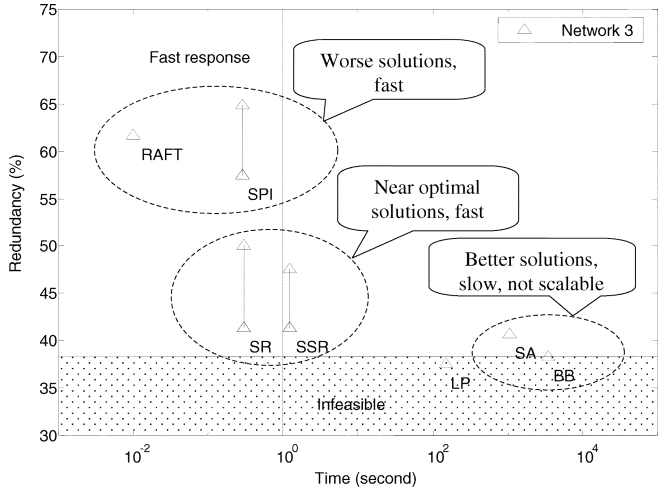

Fig. 15. Comparison of redundancy $\eta = S/W$ versus CPU time of different SCA algorithms for single link failures on Network 3.

and it takes minutes to converge. RAFT is very fast but its solutions are far from optimal. SSR gives very good near-optimal solutions for all networks in very short time. For the first seven networks, SSR takes less than three seconds to find all 64 solutions for one network. For Network 8, it takes about 3.2 minutes.

RAFT is Preferred to SPI RAFT and SPI find close solutions. SPI requires on-line link metric calculation in backup path

TABLE IV
NUMERICAL RESULTS FOR LINK FAILURES

Network	1	2	3	4	5	6	7	8	3b	5b
W	142	224	324	640	826	1670	2732	11104	972	2430
[1] Total spare capacity S										
[2] LP	49.7	100.5	121.5	[3] 227	469	1103	1820	-	381	1340
BB	52	102	124	240	472	1122	1820	-	382	1346
[4] SA	54	112	132	246	474	1216	1830	-	426	1374
[5] SSRmin	54	108	134	252	486	1138	1822	5568	410	1394
SSRmax	66	120	154	272	504	1164	1834	5654	476	1462
SRmin	56	110	134	258	490	1142	1822	5592	432	1422
SRmax	74	128	162	284	516	1176	1862	5670	516	1512
SPImin	84	152	186	366	594	1386	1932	7286	584	1770
SPImax	100	180	210	398	648	1434	2032	7792	658	1936
RAFT	88	178	200	374	620	1448	1994	7516	626	1810
NS	198	326	456	898	1308	2708	5638	16154	1338	3836
Total CPU time (in second)										
LP	52	380	150	1500	650	2100	41	-	140	610
BB	7100	5500	3500	11000	3700	16000	230	-	2000	2900
SA	957	4031	1062	4600	2772	1020	1571	-	-	-
SSRsum	0.5	1	1.2	3.2	3	6.5	7.3	191.8	1.34	3.67
SRsum	0.2	0.3	0.3	0.6	0.7	1.3	1.8	25.5	0.32	0.73
SPIsum	0.2	0.27	0.29	0.52	0.54	1.11	1.42	21.3	0.3	0.63
RAFT	0.01	0.01	0.01	0.03	0.03	0.07	0.07	0.76	0.01	0.03
NS	0.01	0.01	0.01	0.03	0.03	0.07	0.07	0.75	0.01	0.03

[1]LP, BB and SA are run on a SUN Ultra Enterprise server with 4GB memory and 250MHz UltraSPARC CPU. SSR, SR, SPI, RAFT and NS are run on a Pentium III 533MHz PC. [2]LP and BB use AMPL/CPLEX [62], [63]; all others are coded in C++. [3]In LP, the backup path sets include all possible paths. An exception is network 4, where the LP lower bound is not found in one day and the limited path set is used instead. [4]SA is introduced in [64], [17]. [5]For SSR, SR and SPI, we assume that the state information can be synchronized immediately and all ßows are determined before the algorithm starts. We use 64 random number seeds for generating ßow sequences for backup path updates. For these 64 results, we list their maximum and minimum total spare capacities and the sum of their CPU times in the above.

routing, while RAFT is much simpler as it uses hop counts as link metrics. Hence, a simpler algorithm, RAFT, is preferred.

Results are Network Topology Dependent The network topology is an important factor for SCA. The sparser networks tends to have higher redundancies and smaller differences between SSR and BB results. On the other hand, the denser networks can achieve lower network redundancy around 40% where the difference between SSR and BB redundancies goes up to 4%.

Flow Sequence in SSR is an Important Factor The maximum and minimum redundancies in 64 different SSR random cases provide ranges between 0.4% and 8.5%. This indicates that the flow sequence to update backup paths is a critical factor for the SSR algorithm. Although our preliminary study on the flow sequences based on bandwidth and/or hop count does not show any significant effects yet [17], it might still be a topic for future study.

SSR Converges Quickly In the first 7 networks, it takes each flow less than 4 backup path iterations before SSR terminates. In Network 8, this iteration number increases to 10. This convergence speed is fast.

SR is Simple and Efficient SR achieves very good results – only slightly worse than SSR. SR does not reroute backup routes iteratively. This advantage might be more suitable for the distributed backup path routing.

In short, SSR is fast to achieve surprisingly good approximations to the optimal SCA solutions.

In fact, we can use the matrix method to explain why SSR achieves such good results comparing to RAFT. RAFT routes backup path through a minimum hop route. The corresponding operation in matrix $\boldsymbol{G}^r$ is to minimize the summation of its elements since the working path is given. Consequently, minimizing the summation of all elements in $\boldsymbol{G}$ is the objective for RAFT. This operation is equivalent to minimize a lower bound η_{LB1} of network redundancy η, as given in (20). Apparently, reducing the lower bound does not necessarily reduce the redundancy itself.

$$\eta = \frac{S}{W} = \frac{\boldsymbol{e}^T \boldsymbol{s}}{W} \geq \frac{1}{W} \frac{\boldsymbol{e}^T \boldsymbol{G} \boldsymbol{e}}{L-1} = \eta_{LB1}. \tag{20}$$

On the other hand, SSR computes its solutions directly based on a necessary condition of the optimal solution. In an optimal solution, a backup path of a flow has the minimum incremental cost comparing to other alternative backup paths. Otherwise, if another backup path has lower incremental spare cost, we can use it to replace the current optimal backup path to achieve a even lower feasible solution. It is inconsistent with the optimality of the original optimal solution.

This analysis might help to understand why the partial information based schemes [3], [57] have less capacity efficiency.

In conclusion, SSR is a "greedy" search algorithm with special designed "directions"(flow) and "steps"(incremental cost). It partitions the original multi-commodity problem into multiple single-commodity subproblems and iteratively solves them to get a good approximation solution. It has better chance in finding better solutions than RAFT.

VII. NODE FAILURES

In SCA problem for node failures, flows with single-link working paths require link disjoint backup paths. For this

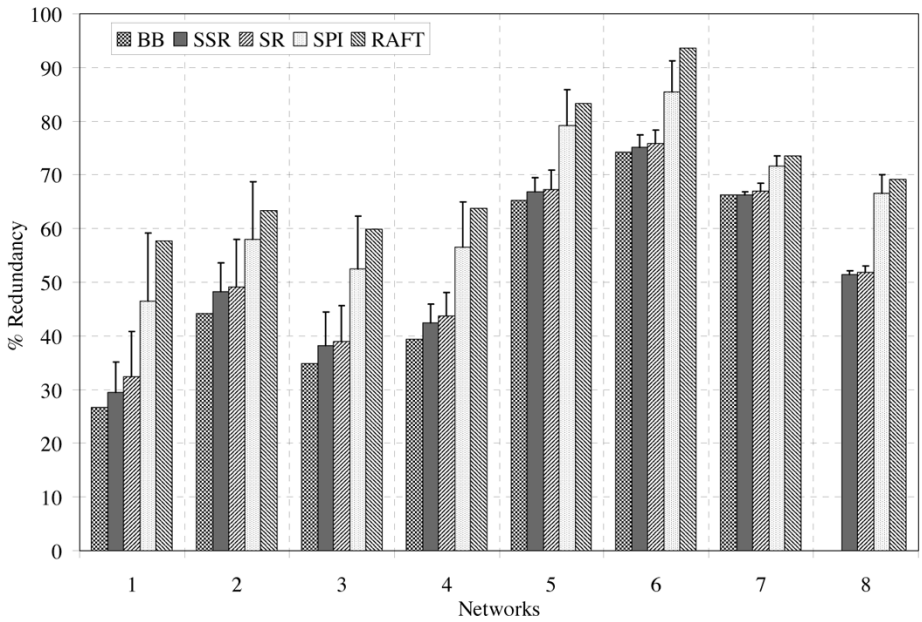

Fig. 16. Comparison of redundancy $\eta = S/W$ in SCA for node failures.

TABLE V
NUMERICAL RESULTS FOR NODE FAILURES

Network	1	2	3	4	5	6	7	8
W	142	224	324	640	826	[1] 1686	2732	11104
Total spare capacity S								
BB	38	99	113	252	539	1252	1812	-
SSRmin	42	108	124	272	552	1268	1812	5720
SSRmax	50	120	144	294	574	1306	1826	5800
SRmin	46	110	126	280	556	1280	1832	5764
SRmax	58	130	148	308	586	1320	1872	5894
SPImin	66	130	170	362	654	1440	1958	7394
SPImax	84	154	202	416	710	1540	2008	7780
RAFT	82	142	194	408	688	1578	2010	7690
NS	198	326	456	910	1324	2736	5652	16278
Total CPU time (in second)								
BB	60	130	720	1700	130	5900	41	-
SSRsum	3.25	3.63	3.84	6.51	5.94	8.6	14.73	293.43
SRsum	0.59	0.63	0.71	1.08	1.17	2.25	2.81	38.18
SPIsum	0.58	0.64	0.66	0.98	1.03	1.96	2.36	28.96
RAFT	0.02	0.02	0.02	0.04	0.04	0.08	0.11	0.92
NS	0.02	0.02	0.02	0.04	0.04	0.08	0.1	0.89

The experiment setups are similar to those in Table III except node failures are protected. [1] The total working capacity is higher because some working paths are adjusted to guarantee they have node-disjoint backup paths.

reason, the failure scenarios here consider all single nodes and links. The matrix $\boldsymbol{F} = [\boldsymbol{B}^T|\boldsymbol{I}_L]^T$, where $\boldsymbol{I}_L$ is an identity matrix of size L to indicate all single link failures on undirected network. In addition, each flow cannot avoid the failures of its source or destination nodes. Such failures are removed from the flow failure adjacent matrix $\boldsymbol{U}$ in (21) in replacement of (7). For this purpose, we introduce two matrices $\boldsymbol{D}^o$ and $\boldsymbol{D}^d$ to indicate relations of flows and their source and destination nodes respectively, where $d^o_{rn} = 1$ iff $o_r = n$ and $d^d_{rn} = 1$ iff $d_r = n$. The zero matrix $\boldsymbol{0}_L$ is a square matrix of size L.

$$\boldsymbol{U} = \boldsymbol{P} \odot \boldsymbol{F}^T - [\boldsymbol{D}^o + \boldsymbol{D}^d|\boldsymbol{0}_L]. \tag{21}$$

Eight networks in Figs. 6–13 and the same experiment setups in Section VI are used again for numerical experiments.

Several algorithms are compared on different networks. Numerical results are summarized in Table V and their network redundancies are drawn on Fig. 16. The total spare capacities found by these algorithms can be sorted as: $\text{Optimal solution} = \text{BB} < \text{SSR} < \text{SR} \ll \text{SPI} \approx \text{RAFT} \ll \text{NS}$. The ranges of redundancies SSR found is still within 4% from the optimal solution found by BB. Moreover, SSR is very fast comparing to other algorithms like SR, SPI and RAFT. Since the CPU times for SSR, SR and SPI are the summation of 64 independent running cases, the time for a single case is lower than a few seconds. Hence, these three algorithms also belong to fast algorithms as RAFT.

In Fig. 17, redundancies versus CPU times of these algorithms on Network 6 is plotted as an example for the tradeoff. SSR achieves good trade-off between optimality and solution time. All these conclusions are very similar to those for protecting link failure in Section VI. These results demonstrate that SSR algorithm is still a good approximation algorithm for the node failure resilient spare capacity allocation problem.

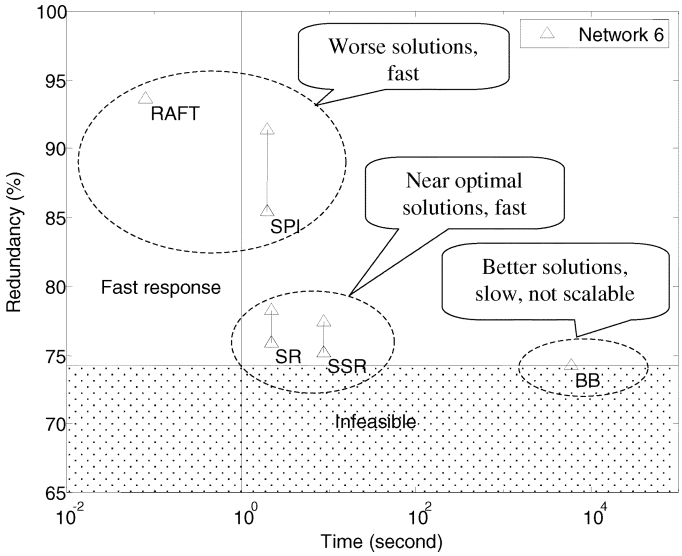

Fig. 17. Comparison of redundancy versus CPU time of different SCA algorithms for node failures on Network 6.

VIII. Summary

In this paper, the NP-complete spare capacity allocation (SCA) problem using share path restoration is studied. The complicated structure for spare capacity sharing among different backup paths is captured by a spare provision matrix. This matrix aggregates per-flow based information and provides sufficient information for spare capacity sharing. Based on the matrix model, the optimal SCA solution is approximated by a fast and efficient algorithm, called successive survivable routing (SSR). Both the matrix-based model and SSR algorithm are also extended for the general cases which protect arbitrary failure scenarios and use nonlinear link cost functions. The numerical results shows that SSR is fast and finds near optimal solutions.

References

[1] W. D. Grover, *Mesh-Based Survivable Transport Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking*. New York: Prentice-Hall, 2003.

[2] C. Dovrolis and P. Ramanathan, "Resource aggregation for fault tolerance in integrated service networks," *ACM Comput. Commun. Rev.*, vol. 28, no. 2, pp. 39–53, 1998.

[3] M. Kodialam and T. V. Lakshman, "Dynamic routing of bandwidth guaranteed tunnels with restoration," in *Proc. IEEE INFOCOM*, Mar. 2000.

[4] ——, "Dynamic routing of restorable bandwidth-guaranteed tunnels using aggregated network resource usage information," *IEEE/ACM Trans. Networking*, vol. 11, no. 3, pp. 399–410, Jun. 2003.

[5] S. Chaudhuri, G. Hjalmtysson, and J. Yates. (2000) Control of Lightpaths in an Optical Network. IETF. [Online]. Available: draft-chaudhuri-ip-olxc-control-00.txt

[6] R. Doverspike and J. Yates, "Challenges for MPLS in optical network restoration," *IEEE Commun. Mag.*, vol. 39, no. 2, pp. 89–96, Feb. 2001.

[7] L. Nederlof, K. Struyue, C. Shea, H. Misser, Y. Du, and B. Tamayo, "End-to-end survivable broadband networks," *IEEE Commun. Mag.*, vol. 9, pp. 63–70, 1995.

[8] R. Diestel, *Graph Theory*, 2 ed: Springer-Verlag, 2000, vol. 173, Graduate Textbooks in Mathematics.

[9] R. Sedgewick, *Algorithms*, 2 ed. Reading, MA: Addison-Wesley, 1988.

[10] W.-P. Wang, D. Tipper, B. Jæger, and D. Medhi, "Fault recovery routing in wide area packet networks," in *Proc. 15th Int. Teletraffic Congr.*, Washington, DC, Jun. 1997.

[11] B. Jæger and D. Tipper, "Prioritized traffic restoration in connection oriented QoS based networks," *Comput. Netw.*, vol. 26, no. 18, pp. 2025–2036, Dec. 2003.

[12] T.-H. Wu, *Fiber Network Service Survivability*. Boston, MA: Artech House, 1992.

[13] ——, "Emerging technologies for fiber network survivability," *IEEE Commun. Mag.*, vol. 33, no. 2, pp. 58–74, Feb. 1995.

[14] R. Doverspike and B. Wilson, "Comparison of capacity efficiency of DCS network restoration routing techniques," *J. Netw. Syst. Manag.*, vol. 2, no. 2, pp. 95–123, 1994.

[15] Y. Xiong and L. G. Mason, "Restoration strategies and spare capacity requirements in self-healing ATM networks," *IEEE/ACM Trans. Networking*, vol. 7, no. 1, pp. 98–110, Feb. 1999.

[16] Y. Xiong and L. Mason, "Comparison of two path restoration schemes in self-healing networks," *Comput. Netw.*, vol. 38, no. 5, 2002.

[17] Y. Liu, "Spare Capacity Allocation: Model, Analysis and Algorithm," Ph.D., Sch. Information Sciences, Univ. Pittsburgh, PA, 2001.

[18] Y. Liu and D. Tipper, "Spare capacity allocation for nonlinear cost and failure-dependent path restoration," in *3rd Int. Workshop on Design of Reliable Communication Networks (DRCN)*, Budapest, Hungary, Oct. 7–10, 2001.

[19] D. A. Dunn, W. D. Grover, and M. H. MacGregor, "Comparison of k-shortest paths and maximum flow routing for network facility restoration," *IEEE J. Select. Areas Commun.*, vol. 2, no. 1, pp. 88–99, Jan. 1994.

[20] W. D. Grover, "Distributed restoration of the transport network," in *Telecommunications Network Management into the 21st Century, Techniques, Standards, Technologies and Applications*, S. Aidarous and T. Plevyak, Eds. New York: IEEE Press, 1994, ch. 11, pp. 337–417.

[21] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms and Applications*. New York: Prentice-Hall, 1993.

[22] J. W. Suurballe, "Disjoint paths in a network," *Networks*, vol. 4, pp. 125–145, 1974.

[23] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, pp. 325–336, 1984.

[24] R. Bhandari, *Survivable Networks Algorithms for Diverse Routing*. Boston, MA: Kluwer, 1999.

[25] Y. Liu and D. Tipper, "Successive survivable routing to protect node failures," in *Proc. IEEE Global Communications Conf.*, San Antonio, TX, Nov. 2001.

[26] E. Oki, N. Matsuura, K. Shiomoto, and N. Yamanaka, "A disjoint path selection scheme with shared risk link groups in GMPLS networks," *IEEE Commun. Lett.*, vol. 6, no. 9, pp. 406–408, Sep. 2002.

[27] D. Xu, Y. Xiong, C. Qiao, and G. Li, "Trap avoidance and protection schemes in networks with shared risk link groups," *J. Lightwave Technol.*, vol. 21, no. 11, pp. 2683–2693, Nov. 2003.

[28] M. Stoer, *Design of Survivable Networks*. New York: Springer-Verlag, 1992, vol. 1531, Lecture Notes in Mathematics.

[29] E. Modiano and A. Narula-Tam, "Survivable routing of logical topologies in WDM networks," in *Proc. IEEE INFOCOM*, Apr. 2001.

[30] O. Crochat, J.-Y. Le Boudec, and O. Gerstel, "Protection interoperability for WDM optical networks," *IEEE/ACM Trans. Networking*, vol. 8, no. 3, pp. 384–395, Jun. 2000.

[31] P. Demeester and M. Gryseels, "Resilience in multilayer networks," *IEEE Commun. Mag.*, vol. 37, no. 8, pp. 70–76, Aug. 1999.

[32] R. Doverspike, "Trends in layered network management of ATM, SONET and WDM technologies for network survivability and fault management," *J. Netw. Syst. Manag.*, vol. 5, pp. 215–220, 1997.

[33] M. Médard, S. G. Finn, R. A. Barry, and R. G. Gallager, "Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs," *IEEE/ACM Trans. Networking*, vol. 7, no. 5, pp. 641–652, Oct. 1999.

[34] M. Herzberg, S. J. Bye, and U. Utano, "The hop-limit approach for spare-capacity assignment in survivable networks," *IEEE/ACM Trans. Networking*, vol. 3, no. 6, pp. 775–784, Dec. 1995.

[35] R. Iraschko, M. MacGregor, and W. Grover, "Optimal capacity placement for path restoration in STM or ATM mesh survivable networks," *IEEE/ACM Trans. Networking*, vol. 6, no. 3, pp. 325–336, Jun. 1998.

[36] M. Herzberg, D. Wells, and A. Herschtal, "Optimal resource allocation for path restoration in mesh-type self-healing networks," in *Proc. 15th Int. Teletraffic Congr.*, vol. 15, Washington, DC, Jun. 1997.

[37] W. D. Grover, R. R. Iraschko, and Y. Zheng, "Comparative methods and issues in design of mesh-restorable STM and ATM networks," in *Telecommunication Network Planning*, P. Soriano and B. Sanso, Eds. Boston, MA: Kluwer, 1999, pp. 169–200.

[38] A. Al-Rumaih, D. Tipper, Y. Liu, and B. A. Norman, "Spare capacity planning for survivable mesh networks," in *Proceedings IFIP–TC6 Networking 2000*, vol. 1815, Lecture Notes in Computer Science (LNCS), Paris, France, May 2000, pp. 957–968.
[39] B. Van Caenegem, W. Van Parys, F. De Turck, and P. M. Demeester, "Dimensioning of survivable WDM networks," *IEEE J. Select. Areas Commun.*, vol. 16, no. 7, pp. 1146–1157, Sep. 1998.
[40] S. Ramamurthy and B. Mukherjee, "Survivable WDM mesh networks, part I – Protection," in *Proc. IEEE INFOCOM*, New York, Mar. 1999.
[41] T. H. Oh, T. M. Chen, and J. L. Kennington, "Fault restoration and spare capacity allocation with QoS constraints for MPLS networks," in *Proc. IEEE Global Communications Conf.*, vol. III, Nov. 2000, pp. 1731–1735.
[42] D. Medhi and D. Tipper, "Some approaches to solving a multi-hour broadband network capacity design problem with single-path routing," *Telecommun. Syst.*, vol. 13, no. 2, pp. 269–291, 2000.
[43] K.-T. Ko, K.-S. Tang, C.-Y. Chan, K.-F. Man, and S. Kwong, "Using genetic algorithms to design mesh networks," *IEEE Comput. Mag.*, vol. 30, no. 8, pp. 56–60, Aug. 1997.
[44] L. T. M. Berry, B. A. Murtagh, G. McMahon, S. Sugden, and L. Welling, "An integrated GA-LP approach to communication network design," *Telecommun. Syst.*, vol. 12, pp. 265–280, 1999.
[45] C.-C. Shyur, T.-C. Lu, and U.-P. Wen, "Applying tabu search to spare capacity planning for network restoration," *Comput. Oper. Res.*, vol. 26, no. 10, pp. 1175–1194, Oct. 1999.
[46] H. Sakauchi, Y. Nishimura, and S. Hasegawa, "A self-healing network with an economical spare-channel assignment," in *Proc. IEEE Global Communications Conf.*, Nov. 1990, pp. 438–442.
[47] J. L. Kennington and M. W. Lewis, "Models and Algorithms for Creating Restoration Paths in Survivable Mesh Networks," Southern Methodist Univ., Dept. Computer Sci. Eng., 99-CSE-5, 1999.
[48] M. H. MacGregor, W. D. Grover, and K. Ryhorchuk, "Optimal spare capacity preconfiguration for faster restoration of mesh networks," *J. Netw. Syst. Manag.*, vol. 5, no. 2, pp. 159–171, 1997.
[49] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification," IETF, RFC 2205, 1997.
[50] E. Bouillet, P. Mishra, J.-F. Labourdette, K. Perlove, and S. French, "Lightpath re-optimization in mesh optical networks," in *Proc. Eur. Conf. Networks & Optical Communications (NOC'02)*, Darsmstadt, Germany, Jun. 2002.
[51] P. Charalambous, G. Ellinas, C. Dennis, E. Bouillet, J.-F. Labourdette, A. A. Akyamaç, S. Chaudhuri, M. Morokhovich, and D. Shales, "A national mesh network using optical cross-connect switches," in *Proc. Optical Fiber Communication Conference (OFC'03)*, Atlanta, GA, Mar. 2003.
[52] W. D. Grover, V. Rawat, and M. H. MacGregor, "Fast heuristic principle for spare capacity placement in mesh-restorable SONET/SDH transport networks," *Electron. Lett.*, vol. 33, no. 3, pp. 195–196, Jan. 1997.
[53] W. D. Grover and D. Y. Li, "The forcer concept and express route planning in mesh-survivable networks," *J. Netw. Syst. Manag.*, vol. 7, no. 2, pp. 199–223, 1999.
[54] J. Duato, "A theory of fault-tolerant routing in wormhole networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 8, no. 8, pp. 790–802, Aug. 1997.
[55] S. Cwilich, M. Deng, D. F. Lynch, and S. J. Phillips, "Algorithms for restoration planning in a telecommunications network," in *Algorithm Engineering and Experimentation Int. Workshop (ALENEX'99)*, vol. 1619, Lecture Notes in Computer Science 1619, 1999, pp. 194–209.
[56] X. Su and C.-F. Su, "An online distributed protection algorithm in WDM networks," in *Proc. IEEE Int. Conf. Communications*, 2001, pp. 1571–1575.
[57] C. Qiao and D. Xu, "Distributed partial information management (DPIM) schemes for survivable networks–Part I," in *Proc. IEEE INFOCOM*, 2002, pp. 302–311.
[58] D. Xu, C. Qiao, and Y. Xiong, "An ultra-fast shared path protection scheme–Distributed partial information management, part II," in *Proc. 10th Int. Conf. Network Protocols (ICNP)*, Nov. 2002, pp. 344–353.
[59] B. Kolman, R. C. Busby, and S. Ross, *Discrete Mathematical Structures*. New York: Prentice-Hall, 1996.
[60] D. O. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," IETF, RFC 3209, 2001.
[61] S. Darisala, A. Fumagalli, P. Kothandaraman, M. Tacca, L. Valcarenghi, M. Ali, and D. Elie-Dit-Cosaque, "On the convergence of the link-state advertisement protocol in survivable WDM mesh networks," in *Proc. 7th IFIP Working Conf Optical Network Design and Modeling (ONDM 2003)*, Budapest, Hungary, Feb. 2003.
[62] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*. San Francisco, CA: Scientific Press, 1993.
[63] *CPLEX User Manual v6.0*, ILOG, Inc., 1998.
[64] Y. Liu, D. Tipper, and P. Siripongwutikorn, "Approximating optimal spare capacity allocation by successive survivable routing," in *Proc. IEEE INFOCOM*, Anchorage, AL, Apr. 2001, pp. 699–708.

Yu Liu (S'96–M'02) received the B.S. degree in information science and technology from Xi'an Jiaotong University, China, in 1993, the M.S. degree in communications and electronic systems from Tsinghua University, China, in 1996, and the Ph.D. in telecommunications from the University of Pittsburgh, Pittsburgh, PA, in 2001.

His research interests include network survivability and optimization, queueing theory, embedded and distributed systems. Since 2001, he has been a Software Engineer at OPNET Technologies, developing automated design solutions for MPLS traffic engineering, multi-layer network design, capital expenditure optimization, link dimensioning, and topology planning in the SP Guru product.

David Tipper (S'78–M'88–SM'95) received the B.S.E.E. degree from Virginia Tech, Blacksburg, and the M.S. and Ph.D. degrees from the University of Arizona, Tucson.

He is an Associate Professor in the Department of Information Science and Telecommunications at the University of Pittsburgh. Prior to joining Pitt in 1994, he was an Associate Professor of Electrical and Computer Engineering at Clemson University, Clemson, SC. His current research interests are network design and traffic restoration procedures for survivable networks, network control (i.e., routing, flow control, etc.), performance analysis, wireless and wired network design. His research has been supported by grants from various government and corporate sources such as NSF, DARPA, NIST, IBM, and AT&T.

Prof. Tipper has been on numerous conference technical committees, including serving as the Technical Program Chair of the Fourth IEEE International Workshop on the Design of Reliable Communication Networks (DRCN 2003). He is currently a member of the editorial board of the *Journal of Network and Systems Management*.

Peerapon Siripongwutikorn (S'98–M'03) received the M.S. and Ph.D. degrees in telecommunications from the University of Pittsburgh, Pittsburgh, PA, in 1998 and 2003, respectively.

He is currently with the Department of Computer Engineering, King Mongkut's University of Technology, Thonburi, Thailand. His current research interests include dynamic resource allocation and control of communication networks, network performance analysis, and network survivability.