

Approximating Optimal Spare Capacity Allocation by Successive Survivable Routing

Yu Liu David Tipper Peerapon Siripongwutikorn

Department of Information Science and Telecommunications

University of Pittsburgh, Pittsburgh, PA 15213, USA

{yuliu, tipper, peerapon}@tele.pitt.edu

Abstract—Spare capacity allocation (SCA) is an important part of fault tolerant network design. In the spare capacity allocation problem one seeks to determine where to place spare capacity in the network and how much spare capacity must be allocated to guarantee seamless communications services survivable to a set of failure scenarios (e.g., any single link failure). Formulated as a multi-commodity flow integer programming problem, SCA is known to be NP-hard. In this paper, we provide a two-pronged attack to approximate the optimal SCA solution: unravel the SCA structure and find an effective algorithm. First, a literature review on the SCA problem and its algorithms is provided. Second, an integer programming (InP) model for SCA is provided. Third, a simulated annealing algorithm using the above InP model is briefly introduced. Next, the structure of SCA is modeled by a matrix method. The per-flow based backup path information are aggregated into a square matrix, called the spare provision matrix (SPM). The size of the SPM is the number of links. Using the SPM as the state information, a new adaptive algorithm is then developed to approximate the optimal SCA solution termed successive survivable routing (SSR). SSR routes link-disjoint backup paths for each traffic flow one at a time. Each flow keeps updating its backup path according to the current network state as long as the backup path is not carrying any traffic. In this way, SSR can be implemented by shortest path algorithms using advertised state information with complexity $O(\#Link^2)$. The analysis also shows that SSR is using a necessary condition of the optimal solution. The numerical results show that SSR has near optimal spare capacity allocation with substantial advantages in computation speed.

Keywords—network survivability, spare capacity allocation, survivable routing, network optimization, multi-commodity flow, approximation algorithm, online algorithm, simulated annealing

I. INTRODUCTION

NETWORK survivability techniques have been proposed to guarantee the seamless communications services in the face of network failures. Most techniques use centralized planning and are developed for circuit-switched networks such as public switched telephone networks [1], SONET/SDH [2], [3], [4], ATM [5], [6], [7], WDM networks [8], [9]. However, circuit-switched backbone networks are being replaced or overlapped with packet-switched networks which provide better manageability of bandwidth granularity and connection types. This architecture migration has been significantly accelerated by the explosive expansion of Internet services. In addition to traditional requirements of cost-effectiveness and service survivability, the increasing Internet traffic and its flexible QoS requirements have brought in many new issues such as scalability, dynamic and distributed bandwidth provisioning for fluctuating traffic. Therefore, it is of increasing importance and necessity for network survivability research to catch up with this trend. Some initial efforts have been made in this direction, for example, various preplanned fault-tolerant routing schemes are studied in RSVP-based IntServ services [10] and IP/MPLS

backbone networks [11]. 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 bandwidth, it is the interest of this paper to have redundancy minimization as an optimization criterion without deteriorating the benefits of preplanned backup path routing. We call this scheme *successive survivable routing*. Our numerical results show that this method is not only feasible, scalable, adaptive, and fast, but also near optimal in redundancy reduction.

This paper is organized as follows. Section II overviews existing survivability techniques and SCA algorithms. In Section III, an Integer Programming (InP) formulation is introduced and its best solution with hop-limited path sets is solved by a commercial Branch and Bound (BB) solver AMPL/CPLEX. By relaxing the integrity of the design variables, we also determine a Linear Programming model which provides the infeasible lower bound of the InP optimal solution. Next, a Simulated Annealing (SA) algorithm is implemented in Section IV to approximate the optimal solution.

The main contribution is in Section V and VI. The structure of SCA is modeled with a novel matrix method approach. The per-flow based backup path information is aggregated into a square matrix, called the spare provision matrix (SPM). The size of the SPM is the number of links. Using the SPM as the state information, a new adaptive algorithm termed successive survivable routing (SSR) is then developed to approximate the optimal SCA solution. SSR routes link-disjoint backup paths for each traffic flow one at a time. Each flow keeps updating its backup path according to the current network state as long as the backup path is not carrying any traffic. In this way, SSR can be implemented by shortest path algorithms using advertised state information with complexity $O(\#Link^2)$ instead of per-flow based information. It is an on-line algorithm to approach the InP optimal solution.

In Section VII, numerical results for a range of networks are used to compare the proposed SSR with other algorithms, namely: Linear Programming lower bound (LP), Integer Programming (InP) using a branch and bound (BB) solution, Simulated Annealing (SA), Survivable Routing (SR), Sharing with Partial Information (SPI) [11], and the Resource Aggregation Fault Tolerance (RAFT) method [10]. These results show that SSR has advantages in terms of its effective approximation to the optimal SCA, fast computation speed, distributed implementation, and adaptive to traffic fluctuations as well as network evolution. Section VIII summarizes our conclusions.

II. RELATED WORK

A. Traditional Network Survivability Techniques

Traditional network survivability techniques have two aspects, survivable network design and network restoration [12]. These two phases are complementary to each other and cooperate to achieve seamless service upon failures.

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 is given. It ensures enough spare capacity for the physical network or the virtual network to recover from a failure via traffic rerouting. For example, given a SONET/SDH mesh-type network topology, the normal traffic demand with their flow allocation, the problems are how much spare capacity should be provisioned and where it should be located in order for the network to tolerate a specified set of failure scenarios (e.g., loss of any single link). The term *mesh-type* does not imply that the network topology is a full mesh, but rather that the network nodes are at least two-connected [13].

In network restoration, affected traffic demand pairs are rerouted upon failure to backup paths that have enough spare capacity provisioned in the survivable network design phase. Compared to dynamic fault-tolerant routing where no spare capacity is pre-allocated before failure, pre-planning spare capacity not only guarantees service restoration, but also minimizes the duration and range of the failure impact. In packet-switched networks, such service guarantees are especially important because backlog traffic accumulated during the failure restoration phase might introduce significant congestion [14], [15]. Pre-planning spare capacity can mitigate or even avoid this congestion. On the other hand, reserving additional spare capacity increases the network redundancy.

Therefore, the major interest in survivable network design has been concentrated on providing cost-efficient spare capacity reservation at a certain survivability level. The *survivability level* gauges the percentage of restorable network traffic upon a failure. In this paper, a 100% survivability level is considered unless it is explicitly noted otherwise. The *network redundancy* is measured by the ratio of the total spare capacity over the total working capacity. It depends highly on the network topology as well as the spare capacity allocation algorithms used. For example, a self healing ring (SHR) topology has 100% redundancy [2], [3]. In mesh-type networks, when working paths are the shortest hop paths, more than 100% redundancy is needed in reserving backup paths. However, the redundancy can be reduced by sharing spare capacity reservations and this reduction is feasible only when failure scenarios are non-overlapping in the network. A *failure scenario* includes all the simultaneously failed network devices or components. In this paper, we consider the failure scenarios where only one network link can fail at any one time.

Restoration schemes can be classified as either *link restoration* and *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 the affected traffic flows. Thus it only patches around the failed link in the original path. In contrast, in path restoration, the end nodes whose traffic is traversing the failed link initiate the rerouting process. In general, path restoration requires less spare capacity reservation than link restoration [16]. Moreover, the selection of backup paths can be *failure dependent* when different failures are protected by different backup paths. A *failure independent* path restoration scheme requires a backup path which is link-disjoint from the working path. This approach requires less signaling support and is easier to implement at the cost of requiring more spare capacity than failure dependent path restoration. In this paper, we use link-disjoint backup paths in failure independent path restoration.

A problem that arises in the failure independent path restoration scheme is the existence of *trap topology* [17], [4]. In a trap topology, the working path may block all the possible link-disjoint backup paths although the network topology is two-connected. For example, when the traffic flow between node 13 and node 15 in Network 6 in Fig. 8 has 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 is to select multiple partially link-disjoint backup paths to protect the working path. However, the resulting restoration scheme is failure dependent. The other way is to modify the working path to render a link-disjoint backup path possible. It is equivalent to routing the working and backup paths simultaneously. The *augmenting path algorithm* can be used for this purpose [18]. It uses a modified network with the same topology where all links have one-unit capacity and the traffic demand from the source to the destination is two units. The algorithm can find two link-disjoint paths and we can use the shorter one for working and the other for backup. Although such working paths might not be the shortest hop paths, they are the shortest among all working paths protected by link-disjoint backup paths. Although this method introduces longer working paths and disturbs traffic on the working paths already been routed, it is still a practical method thanks to the rare occurrence of the trap topology [17] and we use this method to deal with trap topologies.

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 path allocation for all traffic flows. Multi-commodity flow (MCF) models have been widely used to formulate spare capacity allocation problems in different networks like SONET/SDH [19], [20], [21], [6], [22], ATM [6], [7], WDM [8], [9], and IP/MPLS [23]. In these models, pre-calculated path sets for all traffic demand pairs are used to compose the search space for the design variables and the objective is to minimize the total spare capacity required for the restoration from specific failure scenarios. Unfor-

unately, the resulting Integer Programming (InP) formulation is *NP-hard*. Due to the rapid increase in the path set size with the network size, the above models will not scale to many realistic situations. Thus heuristic methods are needed to approach the optimum.

Relaxation methods are widely used to approximate InP solutions. Herzberg et. al. [19] formulate a linear programming (LP) model for the spare capacity allocation problem and treat spare capacity as continuous variables. A rounding process is used 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 can also be extended to input InP formulation when Branch and Bound (BB) is employed for searching the optimal solution [20], [6]. Lagrangian relaxation with subgradient optimization are used by Medhi and Tipper [24]. The Lagrangian relaxation usually simplifies a hard original problem by dualizing the constraints and decomposing it into multiple easier sub-problems. Subgradient optimization is used to iteratively solve the dual variables in these sub-problems.

Genetic Algorithm (GA) based methods are proposed in [24] and [22]. GA evolves the current population of “good solutions” toward the optimality by using carefully designed crossover and mutation operators. Al-Rumaih et. al. [22] encodes the link spare capacity in the chromosomes. The crossover operator is to swap partial spare capacities of two parent links. The mutation operator is to randomly vary spare capacity. The crossover and mutation repeat until feasible offspring are achieved. Then the elitist offspring will evolve into next generation until a stopping condition is met. In [24], the indices of the backup paths in the path set for demand pairs are encoded into the chromosome. The crossover is simply to exchange the backup path indices between two solutions and the mutation is to change the backup path indices of certain traffic demands. Their results show certain resistivity to local optimum in the search space.

There are many other heuristic methods reported including Tabu search [25], Simulated Annealing (SA) [8], Spare Link Placement Algorithm (SLPA) [6], Iterated Cutsets InP-based Heuristics (ICH) [26], Max-Latching Heuristics [6], Subset relaxation [27], and the column generation method [28].

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 [10] for next generation Internet (NGI) IntServ services using the Resource Reservation Protocol (RSVP) [29]. It utilizes a Fault Management Table (FMT) on each link to keep track of all the traffic flows having this link included in their backup paths. The shared spare capacity can be calculated with this FMT. The routing is independently carried out and thus the total spare capacity is not minimized.

Two similar dynamic routing schemes with restoration, Sharing with Partial routing Information (SPI) and Sharing with Complete routing Information (SCI) were introduced recently [11]. In SPI, the backup path routing is based on the shortest path algorithm while the resource minimization is ap-

proximated by using modified link costs in routing. Although SPI is simple and fast, as shown in our numerical results, the redundancy that SPI achieves is not very close to the optimal results. The SCI scheme is similar to the survivable routing (SR) scheme in this paper. However, in [11] it is claimed that per-flow based information is necessary for SCI, unlike the SR scheme proposed here.

C. SCA Structure

The structure of the SCA problem has recently been investigated in a few works.

The *channel dependency graph* was introduced by Duato to analyze network fault tolerance when failure protection routes exist in a parallel computing system [30]. Though it concentrates on the question of how many faults a fault tolerant routing function can deal with, the dependency relations between links on working and backup paths is shown through a dual graph. This provides an important hint for the SCA problem structure we used here. The *forcer concept* was proposed by Grover and Li [31]. It utilizes the relationship between the working and spare capacity reservations to solve the express route planning problem in link restorable networks. This link relationship focuses on pair wise relations among links but not the whole structure of spare capacity sharing among flows. The *fault management table* (FMT) method is the building foundation for the resource aggregation fault tolerant (RAFT) scheme [10]. It provides a local data structure to store the spare capacity sharing information among different flows. It is very difficult to use FMT to share these information globally since such information is per-flow based and hence, not scalable with the network size and the number of flows. An equivalent mathematical formulation of FMT is provided by Medhi and Tipper [24].

From the above discussion, the spare capacity allocation problem is still an important and challenging problem.

III. INTEGER PROGRAMMING MODEL

Consider a directed graph $(\mathcal{N}, \mathcal{L})$, where \mathcal{N} and \mathcal{L} are the node set and the link set for the network. The total numbers of nodes and links in the network are $N = |\mathcal{N}|$ and $L = |\mathcal{L}|$ respectively. For a link $l \in \mathcal{L}$, let w_l denote the unit spare capacity cost. In this paper, we use $w_l = 1$ for minimum hop routing. Let \mathcal{D} denote the set of traffic flows, where $\mathcal{D} \subset (\mathcal{N} \times \mathcal{N})$ and $D = |\mathcal{D}| \leq N \times (N - 1)$ if we only consider bidirectional point-to-point traffic flows. Traffic flow $r (r \in \mathcal{D})$ requires m_r units of traffic demands with 100% restoration level for a set of specific failure scenarios.

The path restoration with link-disjoint restoration scheme can be represented as a *path-link* model where the design variables are indexed by the backup paths for traffic flows and the links in the networks. For each $r \in \mathcal{D}$, let \mathcal{P}^r denote a candidate set of loop-free backup paths which are link-disjoint from a given working path. The path set \mathcal{P}^r is precalculated from the network topology and the working paths selected. The p -th path in path set \mathcal{P}^r is represented by the binary path-link indicator $\delta_l^{r,p}$ which will take “one” if and only if path p uses link l . Here

we consider only the case of any single link failure. We let \mathcal{D}_f denote the set of traffic flows affected upon the failure of link f . \mathcal{D}_f is easily determined from the set of working path traffic utilizing link f . Let $x^{r,p}$ be a binary decision variable which equals one when path p in \mathcal{P}^r is used to protect the working traffic of flow r . Further we let s_l denote the required spare capacity on link l to protect against any single link failure.

TABLE I
NOTATION OF THE INP MODEL

$\mathcal{N}, \mathcal{L}, \mathcal{D}$	Node set, link set and flow set
N, L, D	Numbers of nodes, links and flows
i, j, l	Indices of links
r	Index of flows
w_l	Unit spare capacity cost of link $l, l \in \mathcal{L}$
m_r	Traffic demand of flow $r, r \in \mathcal{D}$
\mathcal{D}_f	Set of affected flows upon failure of link $f, f \in \mathcal{L}$
\mathcal{P}^r	The link-disjoint backup path set for flow $r, r \in \mathcal{D}$
$\delta_l^{r,p}$	1 if path p in the backup path set \mathcal{P}^r for flow r uses link l , 0 otherwise
$x^{r,p}$	Binary decision variable, equals to 1 if flow r uses p for its backup path
s_l	Integer variable denoting spare capacity needed on link l

Given the notation above (summarized in Table I), the spare capacity allocation problem can be given as following:

$$\min_{s_l, x^{r,p}} \sum_{l \in \mathcal{L}} w_l s_l \quad (1)$$

$$\text{s.t.} \quad \sum_{p \in \mathcal{P}^r} x^{r,p} = 1 \quad \forall r \in \mathcal{D} \quad (2)$$

$$s_l \geq \sum_{r \in \mathcal{D}_f} (m_r \sum_{p \in \mathcal{P}^r} \delta_l^{r,p} x^{r,p}), \quad \forall l, f \in \mathcal{L}, l \neq f \quad (3)$$

$$s_l = \text{integer}, x^{r,p} = 0 \text{ or } 1 \quad (4)$$

The objective function in (1) is to minimize the total cost of spare capacity in the network. Constraints (2) and (4) guarantee that there is one and only one link disjoint backup path selected for each flow for each failure case. The total spare capacity needed on each link is determined by constraint (3), which picks the maximum of required spare bandwidth for each link under any single link failure. This InP formulation is *NP-hard*. It is topology dependent and does not scale with N, L and D .

By removing s_l , we reach an equivalent formulation with objective function (5) subject to constraints (2) and (4). This new objective function [24] can be evaluated easier using search-based heuristics, such as simulated annealing.

$$\min \sum_{l \in \mathcal{L}} w_l \left[\max_{f \in \mathcal{L} - \{l\}} \left\{ \sum_{r \in \mathcal{D}_f} (m_r \sum_{p \in \mathcal{P}^r} \delta_l^{r,p} x^{r,p}) \right\} \right] \quad (5)$$

IV. SIMULATED ANNEALING

Simulated Annealing (SA) is a stochastic hill-climbing heuristic search method. It allows the search to explore a larger

area in the search space without being trapped in local optima prematurely, and the best solution found during the search is regarded as the final result [32]. We use SA to solve the above InP model for comparison with our SSR algorithm.

The probability of accepting non-improving moves, p_a , is a function of two parameters - the difference between the objective values δ and the control temperature T . This probability is generally given by $p_a = e^{-\frac{\delta}{T}}$ where T is determined by a so-called annealing, or cooling, scheme which will be discussed later. T decreases as the search proceeds and hence p_a is gradually reduced. As T approaches zero, the search reduces to a greedy search and can be trapped in the nearest local optima. The generally accepted stopping criterion is that the search experiences no improvement in the best objective value found so far for some number of moves.

There are many possible annealing schemes for the update of T . In our case, we use the annealing scheme $T_n = \alpha \times T_{n-1}$ where T_n is the temperature at the n th temperature update, and α is an arbitrary constant between 0 and 1. Usually, T is updated periodically every accepted U moves. The parameter α determines how slowly T decreases. Typically, α is between 0.9 and 0.95. Parameter tuning for U, α , and the initial value of T (T_{\max}) is critical to the SA performance. The detailed SA parameters are given in Section VII with their related numerical results.

The move operation dictates how one obtains another solution in the neighborhood of a current solution. A solution is represented by a set of selected link-disjoint backup path $p \in \mathcal{P}^r$, one for each demand pair $r \in \mathcal{D}$. We define a move operation as selecting a traffic demand pair uniformly from \mathcal{D} and altering its current backup path in the current solution. In order to select a new backup path for the move, we assign each backup path a weight which is simply its index in the path set \mathcal{P}^r . Then, we multiply the weight of each path above with a standard uniform $[0 - 1]$ variate. The backup path with the smallest product is chosen for next move. It can be shown that by using this method, the probability of selecting i th backup path in the path set \mathcal{P}^r is $p_i = \frac{1/i}{\sum_{j=1}^{|\mathcal{P}^r|} 1/j}$. Since backup paths in path sets are ordered ascendingly with the number of hops, those with fewer hops (smaller index) are more likely to be selected.

If the new objective value (O_n) after the move is better (smaller) than the current objective value (O_c), we accept the new solution. Otherwise, we accept it with probability $p_a = e^{(\frac{O_c - O_n}{T})}$ where T depends on the temperature period of the annealing schedule described earlier. Since we minimize the objective function in (5), non-improving moves result in $O_c \leq O_n$ and thus $p_a \leq 1$.

The above backup path altering process does not need additional information about the topology. Instead, it simply changes the indices of backup paths of a randomly chosen flow. This does not guarantee best search direction at each move, but it greatly improves the simplicity by avoiding additional shortest path calculations.

V. SPARE PROVISION MATRIX METHOD

In this section, a matrix method approach is given to present the SCA problem. The restoration scheme considered is the path restoration with link disjoint routes. We consider single link failures. Given network topology, traffic flows and their working paths in SCA, the objective of SCA is to minimize the total spare capacity reservation in the network. The decision variables are backup paths. In order to minimize this objective function, backup paths can share capacity on common links if their working paths are link disjointed. For example, there are two flows in the network in Fig. 1, and the link-disjointed working paths are a-b-c and e-d. If the backup paths are a-e-c and e-c-d, they can share the spare capacity reservations on link 6 (e-c).

TABLE II
NOTATION OF THE MATRIX METHOD

$\mathbf{A} = \{\mathbf{a}_r\} = \{a_{rj}\}$	Working path-link incidence matrix
$\mathbf{B} = \{\mathbf{b}_r\} = \{b_{ri}\}$	Backup path-link incidence matrix
$\mathbf{m} = \{m_r\}$	Vector of traffic demands
$\mathbf{M} = \text{Diag}(\mathbf{m})$	Diagonal matrix of traffic demands
$\mathbf{C} = \{c_{ij}\}$	Spare provision matrix
$\mathbf{C}^r = \{c_{ij}^r\}$	Contribution of flow r to \mathbf{C}
$\mathbf{s} = \{s_i\}$	Vector of spare capacity reservations
W, S	Total working, spare capacity
$\eta = S/W$	Network redundancy
o_r, d_r	Origin/destination nodes of flow r
$\mathbf{v}_r = \{v_{ri}\}$	Incremental spare reservation vector of flow r

Our notation is summarized in Table I and II. A network is represented by a graph of N nodes and L links with D traffic flows. Both links and flows are numbered sequentially. The link capacity is unlimited for the simplicity of analysis. Without loss of generality, we assume that links and flows are undirected. Each flow $r, 1 \leq r \leq D$ is specified by its origin/destination node pair (o_r, d_r) and traffic demand m_r , where $o_r < d_r, \forall 1 \leq r \leq D$ due to our undirected flow assumption. Working and backup paths are represented by two $1 \times L$ binary row vectors $\mathbf{a}_r = \{a_{rl}\}$ and $\mathbf{b}_r = \{b_{rl}\}$. The l -th element in these vectors equals to one if and only if the corresponding path uses link l . Path-link incidence matrices for working and backup paths are the collections of all the incidence vectors, forming $D \times L$ matrices $\mathbf{A} = \{a_{rl}\}$ and $\mathbf{B} = \{b_{rl}\}$ respectively. The traffic demand matrix $\mathbf{M} = \text{Diag}(\{m_r\}_{D \times 1})$ is a diagonal matrix.

The *spare provision matrix* $\mathbf{C} = \{c_{ij}\}_{L \times L}$ can be found in two ways. One way is given in (6). The value of c_{ij} is the minimum spare capacity required on link i when link j fails. Moreover, the spare capacity reservations on the links are given by the column vector $\mathbf{s} = \{s_i\}_{L \times 1}$ in (7). The ‘‘row-max’’ operator returns a column vector, with each entry being the maximum item in each row of a given matrix.

$$\mathbf{C} = \mathbf{B}^T \mathbf{M} \mathbf{A} \quad (6)$$

$$\mathbf{s} = \text{row-max } \mathbf{C} \quad (7)$$

The other way to find \mathbf{C} is through aggregating per-flow based information, including working and backup paths. First, the contribution of a single traffic flow r to \mathbf{C} is given by $\mathbf{C}^r = \{c_{ij}^r\}_{L \times L}$ in (8), where \mathbf{a}_r and \mathbf{b}_r are the row vectors in \mathbf{A} and \mathbf{B} representing flow r 's working and backup paths respectively. The spare provision matrix \mathbf{C} is thus given in (9).

$$\mathbf{C}^r = m_r (\mathbf{b}_r^T \mathbf{a}_r), \quad r = 1, \dots, D \quad (8)$$

$$\mathbf{C} = \sum_{r=1}^D \mathbf{C}^r \quad (9)$$

From these equations (6)-(9), per-flow based information in \mathbf{A} and \mathbf{B} is replaced by \mathbf{C} as the stored network state information. The space complexity is decreased from $O(DL)$ down to $O(L^2)$ and it is independent of the number of flows. This improves the scalability of the spare capacity sharing. It also provides privacy and transparency among traffic flows in an open network environment.

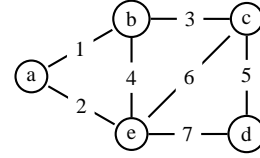


Fig. 1. Example five-node network

TABLE III
MATRICES \mathbf{A} , \mathbf{B} , \mathbf{C} AND \mathbf{s} IN THE FIVE-NODE NETWORK

r	Flows		\mathbf{A}	\mathbf{B}	l	\mathbf{C}	\mathbf{s}
	o_r	d_r					
1	a	b	1000000	0101000	1	0211101	2
2	a	c	1010000	0100101	2	2021100	2
3	a	d	0100001	1010100	3	0100011	1
4	a	e	0100000	1001000	4	1110010	1
5	b	c	0010000	0001010	5	1110002	2
6	b	d	0010100	1100001	6	0010101	1
7	b	e	0001000	1100000	7	1020200	2
8	c	d	0000100	0000011			
9	c	e	0000010	0011000			
10	d	e	0000001	0000110			

The five-node network in Fig. 1 is given as an example of the SPM method. Considering only unit traffic demands, we set $\mathbf{M} = \mathbf{I}$ where \mathbf{I} is the identity matrix of size D . Then, $\mathbf{C} = \mathbf{B}^T \mathbf{A}$. The matrices \mathbf{A} , \mathbf{B} , \mathbf{C} and \mathbf{s} are given in Table III. Note that all the diagonal elements of \mathbf{C} are zeroes because working and backup paths for each flow are link disjointed. The total spare capacity reservation $S = \mathbf{e}^T \mathbf{s} = 11$, where \mathbf{e} is the unit column vector of length L . We also get the total working capacity reservation $W = \mathbf{e}^T \mathbf{M} \mathbf{A} \mathbf{e} = 13$ and the total spare capacity reservation without spare capacity sharing $S' = \mathbf{e}^T \mathbf{M} \mathbf{B} \mathbf{e} = 23$. Compared to S' , the total spare capacity reservation S is reduced significantly due to spare capacity sharing.

The matrix method for more complicated SCA problems, such as cases with directed links, node failures, no-linear link cost function, and multi-layer networks, are discussed in [33].

VI. SUCCESSIVE SURVIVABLE ROUTING

In SSR, each traffic flow first routes its working path then routes its backup path at its source node. The reason for the sequential routing of working and backup paths is their different levels of importance. Working paths carry traffic most of the time whereas backup paths are used only upon their working path failures. Based on the above matrix method, the SSR implementation at the source node of a flow r is given in Fig. 2.

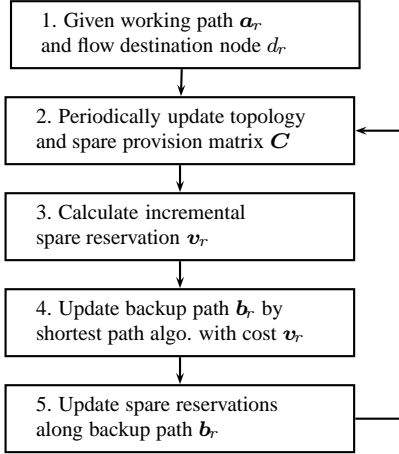


Fig. 2. Flow chart of the SSR algorithm at the source node of flow r

In step 1, each flow is initialized by giving its working path \mathbf{a}_r and destination node d_r .

In step 2, current network state information is collected and updated periodically. The update period should be long enough to guarantee the stability of the algorithm.

Keeping \mathbf{C} up-to-date is important for the efficiency of a distributed protocol. There are two methods in collecting \mathbf{C} over the network. One is link-based and the other is node-based. In the link-based method, each link calculates its corresponding row vector in \mathbf{C} from all working paths of those flows whose backup paths use this link. Every node collects all such row vectors from its outward links to form a link state packet of size $O(NL)$. The packet is then broadcasted to other nodes. In the node-based method, the source node of flow r calculates its contribution matrix \mathbf{C}^r . Each node aggregates all \mathbf{C}^r locally, then 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 $O(NL)$ to $O(L^2)$, it does not require spreading working path information along backup paths. Both methods collect the state information \mathbf{C} with space complexity $O(L^2)$. The per-flow based information is not required for backup path routing and reservation. This advantage makes SSR possible for distributed implementation.

In step 3, we define the following notation to calculate incremental spare reservation vectors. Given \mathbf{C} , \mathbf{b}_r and \mathbf{C}^r , we

define $\mathbf{C}^- = \mathbf{C} - \mathbf{C}^r$ and $\mathbf{s}^- = \text{row-max}(\mathbf{C}^-)$ as the state information after \mathbf{b}_r is removed. Let \mathbf{b}_r^+ denote an alternative backup path for flow r . we have $\mathbf{C}^{r+}(\mathbf{b}_r^+) = m_r \mathbf{b}_r^{+T} \mathbf{a}_r$. This produces a new spare capacity reservation vector $\mathbf{s}^+(\mathbf{b}_r^+) = \text{row-max}(\mathbf{C}^- + \mathbf{C}^{r+}(\mathbf{b}_r^+))$. Finally, let $\mathbf{b}_r^+ = \mathbf{e} - \mathbf{a}_r$, the *incremental spare reservation vector* for traffic flow r is

$$\mathbf{v}_r = \{v_{ri}\}_{L \times 1} = \mathbf{s}^+(\mathbf{e} - \mathbf{a}_r) - \mathbf{s}^-. \quad (10)$$

If the original \mathbf{C} is an optimal solution, we know that any alternative backup path \mathbf{b}_r^+ has $S^+ \geq S$. The optimal backup path \mathbf{b}_r should lead to minimum $\mathbf{e}^T \mathbf{s}^+(\mathbf{b}_r^+)$ among all backup paths $\mathbf{b}_r^+ \leq \mathbf{e} - \mathbf{a}_r$. Hence, the minimization of the incremental spare capacity is a *necessary condition* for the global optimal SCA solution.

In step 4, a possible better backup path is calculated by using the shortest path algorithm with link cost \mathbf{v}_r . We call this scheme *Survivable Routing (SR)*. It has the same time complexity as shortest path algorithms in each flow and space complexity $O(L^2)$.

In step 5, if the backup path is changed in step 4, the spare capacity reservations along the path will be updated accordingly.

After this step, the algorithm returns to step 2 to start next backup path update.

When there is no backup path update in the network, the algorithm reaches a local optimum. Through this iterative backup path routing and state information collection, SSR finds a near optimal solution. The time complexity of SSR depends not only on the complexity of SR, but also on the convergence of flow iterations. Because the above backup path update will not increase S and S is integer, SSR can converge within a bounded number of iterations.

VII. NUMERICAL EXPERIMENTS

Eight network topologies shown in Fig. 3–10 are used to assess the proposed SSR algorithm. The networks have an average node degree \bar{d} ranged from 2.31 to 4.4 as given in Table IV. Without loss of generality, we assume symmetrical traffic flows between any node pairs. All flows have one unit bandwidth demand in all eight networks, i.e., $m_r = 1, \forall r \in \mathcal{D}$. This unit traffic demand is selected for the ease of comparison among networks. For network 3 and 5, We also provide results when demands are between one and five units in cases 3b and 5b.

The total spare capacities and their total CPU times are given in Table V. The network redundancies are plotted in Fig 11. From these results, we derive the following conclusions:

SSR finds near optimal solutions. The redundancies from all algorithms can be roughly ordered as: $\text{LP} \leq \text{optimal solutions} \leq \text{BB} \leq \text{SA} \leq \text{SSR} \leq \text{SR} \ll \text{SPI} \approx \text{RAFT} \ll \text{NS}$. The optimal solutions are upper bounded by BB and lower bounded by LP. The gaps are very narrow in all networks. SA provides good approximation to the optimal solutions. At the other extreme, NS does not have spare capacity sharing and consequently it has the highest redundancy ratio, which is above 100%. Since there are small redundancy gaps from LP to SSR, which are less than 4%. SSR has achieved very good near optimal solutions.

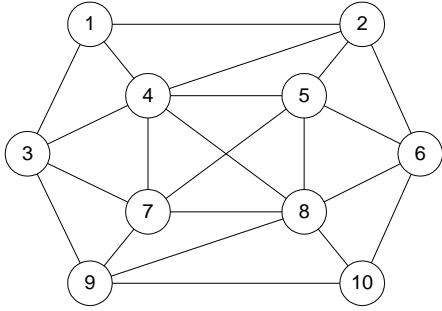


Fig. 3. Network 1 (10 nodes, 22 links, $\bar{d} = 4.4$)

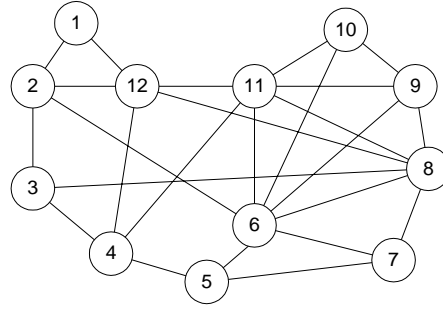


Fig. 4. Network 2 (12 nodes, 25 links, $\bar{d} = 4.17$)

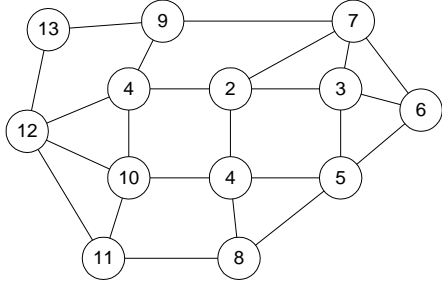


Fig. 5. Network 3 (13 nodes, 23 links, $\bar{d} = 3.54$)

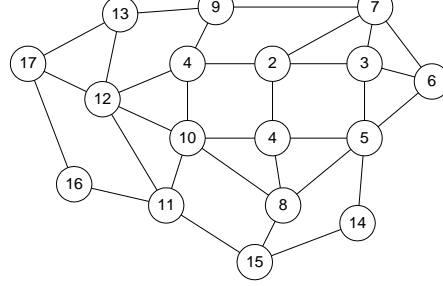


Fig. 6. Network 4 (17 nodes, 31 links, $\bar{d} = 3.65$)

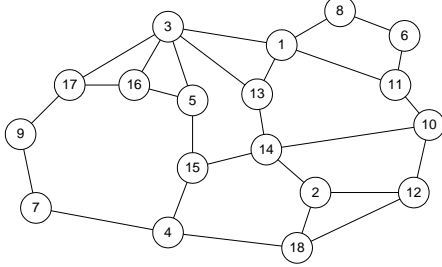


Fig. 7. Network 5 (18 nodes, 27 links, $\bar{d} = 3.00$)

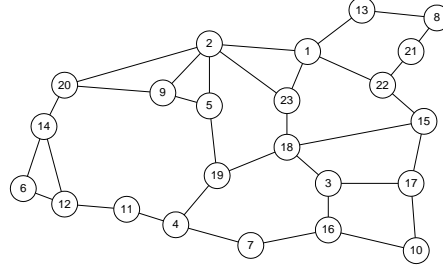


Fig. 8. Network 6 (23 nodes, 33 links, $\bar{d} = 2.87$)

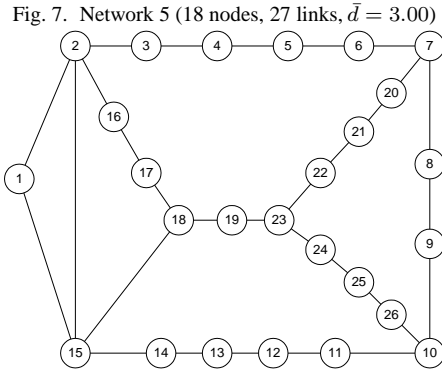


Fig. 9. Network 7 (26 nodes, 30 links, $\bar{d} = 2.31$)

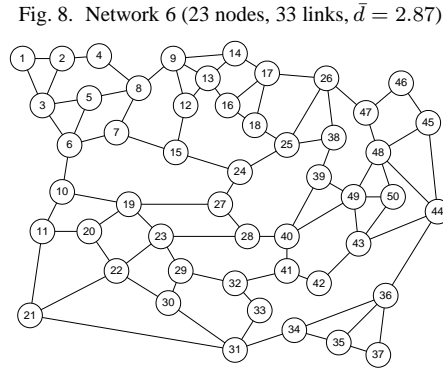


Fig. 10. Network 8 (50 nodes, 82 links, $\bar{d} = 3.28$)

TABLE IV
NETWORK INFORMATION

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

TABLE V
NUMERICAL RESULTS

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	-	-	-
⁵ SSR _{min}	54	108	134	252	486	1138	1822	5568	410	1394
SSR _{max}	66	120	154	272	504	1164	1834	5654	476	1462
SR _{min}	56	110	134	258	490	1142	1822	5592	432	1422
SR _{max}	74	128	162	284	516	1176	1862	5670	516	1512
SPI _{min}	84	152	186	366	594	1386	1932	7286	584	1770
SPI _{max}	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	-	-	-
SSR _{sum}	0.5	1	1.2	3.2	3	6.5	7.3	191.8	1.34	3.67
SR _{sum}	0.2	0.3	0.3	0.6	0.7	1.3	1.8	25.5	0.32	0.73
SPI _{sum}	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

¹LP, 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. ²LP and BB use AMPL/CPLEX [34], [35]; all others are coded in C++. ³In 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. ⁴The parameters of SA are $\alpha = 0.9$, $b = 2000$, and $T_{max} = 100$. ⁵For SSR, SR and SPI, we assume that the state information can be synchronized immediately and all flows are determined before the algorithm starts. We use 64 random number seeds for generating flow 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.

SSR has fast computation and good scalability. The computation times for these algorithms are significantly different. BB takes hours and cannot scale to larger networks such as network 8. SA has faster speed comparing to BB, but it still needs parameter tuning and it takes minutes to converge. RAFT is very fast but the results are far from the optimal solution. SSR gives very good near optimal solutions for all networks in seconds level.

RAFT is preferred to SPI. SPI requires on-line link metric calculation in backup path routing, while RAFT is much simpler as it uses hop counts as link metrics.

Flow sequence in SSR is an important factor. The maximum and minimum redundancies in 64 different SSR random cases provide ranges from 0.4% to 8.5%. This indicates that the flow sequence of updating the backup paths is a critical factor for the SSR algorithm. It is an interesting topic for further research.

SSR has fast convergence. It takes SSR less than four backup path updates for all flows in the first 7 networks and less than ten updates in network 8.

In short, SSR is a fast algorithm that achieves surprisingly good approximation to the optimal SCA solutions. In fact, we can use the SPM method to explain why SSR achieves such good results by comparing it to RAFT. RAFT routes backup paths through minimum hop paths. The corresponding operation in matrix C^r is to minimize the summation of its elements after working paths are given. Consequently, minimizing the summation of all elements in C becomes the objective. This operation is equivalent to minimize a lower bound of network

redundancy η , as given below:

$$\eta = \frac{S}{W} = \frac{e^T s}{W} \geq \frac{1}{W} \frac{e^T C e}{L - 1}. \quad (11)$$

Apparently, reducing the lower bound of redundancy does not necessarily reduce the achievable redundancy itself. On the other hand, SSR computes its solutions based on the necessary condition for η optimization directly. This condition is discussed below (10). Hence, SSR has better chance in approximating the optimal solution than RAFT.

VIII. SUMMARY

This paper first overviews the spare capacity allocation problem, which is an NP-hard multi-commodity flow problem. Then we introduce an Integer Programming model and a Simulated Annealing algorithm. These provide the basis for the later comparison with our two-pronged attack to SCA: modeling the SCA structure in a novel matrix method and constructing an effective approximation algorithm termed successive survivable routing.

The matrix method aggregates the state information for backup path routing to a square matrix, called the spare provision matrix (SPM), based on which the Successive Survivable Routing (SSR) algorithm is developed by utilizing a necessary condition for redundancy optimization. From extensive numerical results and comparisons, we show that this algorithm computes near optimal SCA solutions extremely fast. Its flow-based optimization allows the computation to be distributed over the

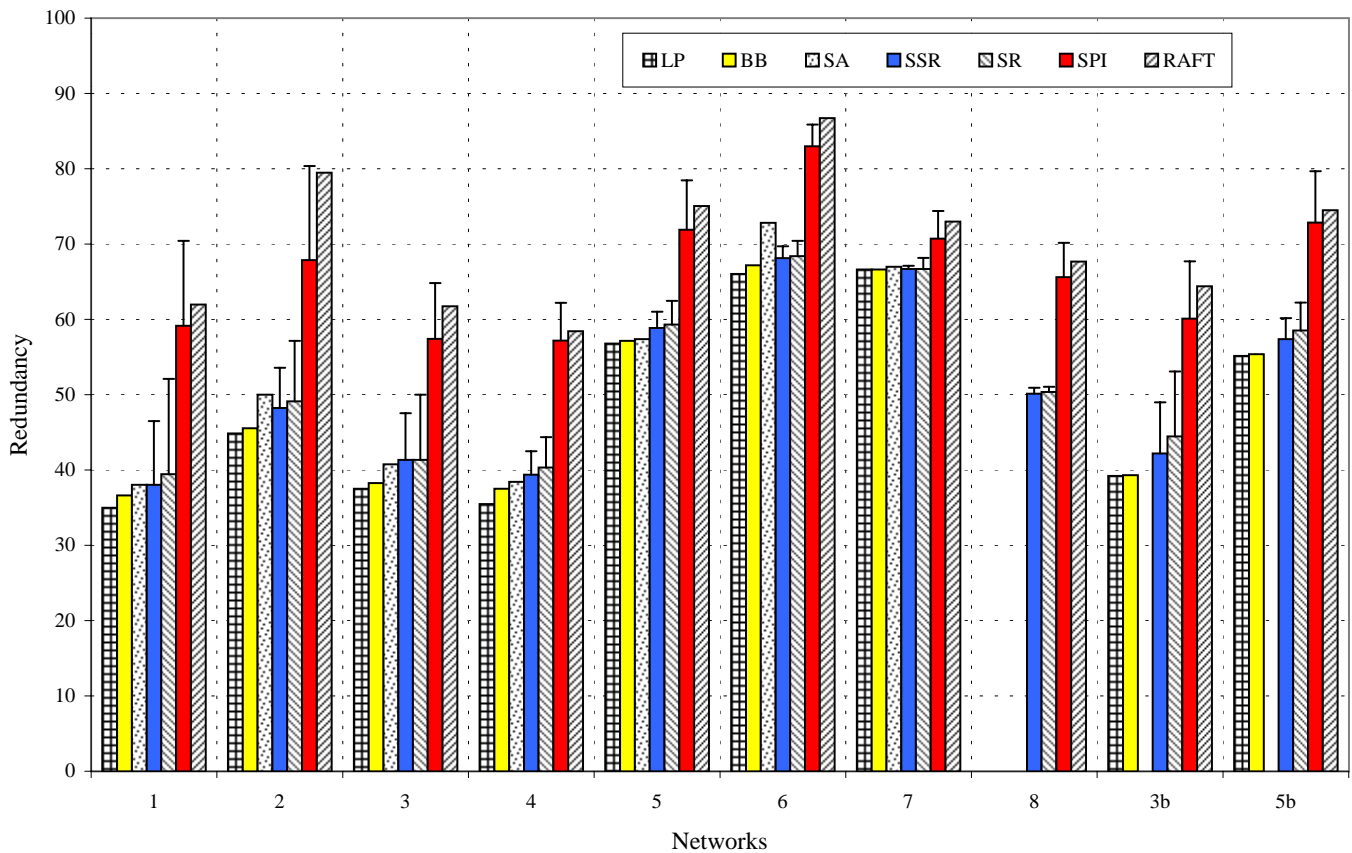


Fig. 11. Comparison of redundancy $\eta = S/W$ over networks. The error bars on SSR, SR and SPI give the ranges of 64 results from random flow sequences.

network. It also has good scalability, low complexity and adaptive to traffic fluctuations as well as network evolution. These features make SSR an excellent algorithm for spare capacity allocation in mesh type networks.

ACKNOWLEDGMENT

Thanks to Stella X. Yu, Miguel Labrador Perez and anonymous reviewers for their valuable comments.

REFERENCES

- [1] J. C. McDonald, "Public network integrity – avoiding a crisis in trust," *IEEE Journal on Selected Areas of Communications*, vol. 12, no. 1, pp. 5–12, Jan. 1994.
- [2] Tsong-Ho Wu, *Fiber Network Service Survivability*, Artech House, 1992.
- [3] T.-H. Wu, "Emerging technologies for fiber network survivability," *IEEE Communications Magazine*, vol. 33, no. 2, pp. 58–74, Feb. 1995.
- [4] W.D. Grover, "Distributed restoration of the transport network," in *Telecommunications Network Management into the 21st Century, Techniques, Standards, Technologies and Applications*, Salah Aidarous and Thomas Plevyak, Eds., chapter 11, pp. 337–417. IEEE Press, 1994.
- [5] Byung Han Ryu, Masayuki Murata, and Hideo Miyahara, "Design method for highly reliable virtual path based ATM networks," *IEICE Transactions on Communications*, vol. E79-B, no. 10, pp. 1500–1514, 10 1996.
- [6] 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 B.Sanso, Ed., pp. 169–200. Kluwer Academic Publishers, 1999.
- [7] Yijun Xiong and Lorne 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.
- [8] B. Van Caenegem, W. Van Parys, F. De Turck, and P. M. Demeester, "Dimensioning of survivable WDM networks," *IEEE Journal on Selected Areas of Communications*, vol. 16, no. 7, pp. 1146–1157, Sept. 1998.
- [9] S. Ramamurthy and B. Mukherjee, "Survivable WDM mesh networks, part I - protection," in *Proceedings of IEEE INFOCOM'99*, New York, USA, March 21-25 1999, IEEE, IEEE Press.
- [10] 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.
- [11] Murali Kodialam and T.V. Lakshman, "Dynamic routing of bandwidth guaranteed tunnels with restoration," in *Proceeding of IEEE INFOCOM*, Mar. 2000.
- [12] L. Nederlof, Kris Struyue, Howard Misser Chris Shea, Yonggang Du, , and Braulio Tamayo, "End-to-end survivable broadband networks," *IEEE Communications Magazine*, pp. 63–70, 9 1995.
- [13] Robert Sedgewick, *Algorithms*, Readings, Mass.: Addison-Wesley, 2 edition, 1988.
- [14] W.-P. Wang, D. Tipper, B. Jæger, and D. Medhi, "Fault recovery routing in wide area packet networks," in *Proceedings of 15th International Teletraffic Congress*, Washington, DC, June 1997.
- [15] B. Jæger and D. Tipper, "On fault recovery priority in ATM networks," in *Proceedings of 1998 IEEE International Communications Conference (ICC'98)*, Atlanta, GA, June 1998.
- [16] R. Doverspike and B. Wilson, "Comparison of capacity efficiency of DCS network restoration routing techniques," *Journal of Network and System Management*, vol. 2, no. 2, pp. 95–123, 1994.
- [17] D. Anthony Dunn, Wayne D. Grover, and Mike H. MacGregor, "Comparison of k-shortest paths and maximum flow routing for network facility restoration," *IEEE Journal on Selected Areas of Communications*, vol. 2, no. 1, pp. 88–99, 1 1994.
- [18] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms and Applications*, Prentice Hall, New York, 1993.
- [19] M. Herzberg, S. Bye, and U. Utano, "The hop-limit approach for spare-

- capacity assignment in survivable networks," *IEEE/ACM Transactions on Networking*, pp. 775–784, Dec. 1995.
- [20] 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*, pp. 325–336, June 1998.
- [21] M. Herzberg, D. Wells, and A. Herschtal, "Optimal resource allocation for path restoration in mesh-type self-healing networks," in *Proceedings of 15th International Teletraffic Congress*, Washington, D.C., June 1997, vol. 15.
- [22] 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.
- [23] Tae H. Oh, Thomas M. Chen, and Jeffery L. Kennington, "Fault restoration and spare capacity allocation with QoS constraints for MPLS networks," in *Proceeding of IEEE Global Communications Conference*, Nov. 2000, vol. III, pp. 1731–1735.
- [24] Deep Medhi and David Tipper, "Some approaches to solving a multi-hour broadband network capacity design problem," *Telecommunication Systems*, vol. 13, no. 2, pp. 269–291, 2000.
- [25] Ching-Chir Shyur, Ta-Chien Lu, and Ue-Pyng Wen, "Applying tabu search to spare capacity planning for network restoration," *Computers & Operations Research*, vol. 26, no. 10, pp. 1175–1194, October 1999.
- [26] H. Sakauchi, Y. Nishimura, and S. Hasegawa, "A self-healing network with an economical spare-channel assignment," in *Proceeding of IEEE Global Communications Conference*, Nov. 1990, pp. 438–442.
- [27] Jeffery L. Kennington and Mark W. Lewis, "Models and algorithms for creating restoration paths in survivable mesh networks," Tech. Rep. 99-CSE-5, Dept. of Computer Science and Engineering, Southern Methodist University, 1999.
- [28] M.H. MacGregor, W.D. Grover, and K. Ryhorchuk, "Optimal spare capacity preconfiguration for faster restoration of mesh networks," *Journal of Network and System Management*, vol. 5, no. 2, pp. 159–171, 1997.
- [29] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification*, Internet Engineering Task Force, Sept. 1997.
- [30] J. Duato, "A theory of fault-tolerant routing in wormhole networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 8, no. 8, pp. 790–802, Aug. 1997.
- [31] W. D. Grover and D. Y. Li, "The forcer concept and express route planning in mesh-survivable networks," *Journal of Network and System Management*, vol. 7, no. 2, pp. 199–223, 1999.
- [32] C. Reeves, *Modern heuristic techniques for combinatorial problems*, Oxford: Blackwell Scientific Publishing, 1993.
- [33] Yu Liu, *Spare capacity allocation method, analysis and algorithms*, Ph.D. dissertation, University of Pittsburgh, Apr. 2001.
- [34] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: a Modeling Language for Mathematical Programming*, Scientific Press, San Francisco, 1993.
- [35] ILOG, *Using the CPLEX Callable Library*, ILOG, Inc., 1998.