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, a integer programming 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 of $O(\#Link^2)$. 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 communication 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 *survivable routing*. Our numerical results show that this method is not only feasible, scalable, adaptive, and fast, but also near optimal in redundancy reduction.

Supported in part by DARPA under No. F30602-97-1-0257 and NSF grant NCR-9506652

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 of $O(\#Link^2)$ instead of per-flow based information. It is an on-line algorithm to approach the InP optimal solution utilizing necessary conditions of the 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 assignment 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* doesn't 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]. Preplanning 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-disjointed 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 assignment 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. Unfortunately, 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 assignment 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 subproblems.

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 stoping 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 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 approximated 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 scheme in this paper. However, in [11] it is claimed that per-flow based information is necessary for SCI, unlike the SSR 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 the spare capacity sharing operation between 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 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-disjointed 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 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 includes link l, 0 otherwise
$x^{r,p}$	Binary decision variable, equals to 1 if flow r uses p for its backup
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:

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

s.t.

$$\sum_{p \in \mathcal{P}^r} x^{r,p} = 1 \quad \forall r \in \mathcal{D} \tag{2}$$

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

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

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 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 constraints (3), which will pick the maximum required spare bandwidth of each link under any single link failure scenario. 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 in search-based heuristics, such as simulated annealing.

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

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 InP model of Section III.

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. Basically, T decreases as the search proceeds and hence gradually reducing p_a. As T approaches zero, the search reduces to a greedy search and will be trapped in the nearest local optima. The generally accepted stopping criteria 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 nth temperature update, and α is an arbitrary constant between 0 and 1. Usually, T is updated periodically every number of accepted moves U. The parameter α determines how slowly T decreases. Values of α between 0.9 and 0.95 are typical. Parameters 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 search space of the 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 smallest product is chosen for the next move operation. It can be shown that by using this method, the probability of selecting ith backup path in the path set $\mathcal{P}^r$ is $p_i = \frac{1/i}{\sum_{j=1}^{P^r} 1/j}$. Since the backup paths in the path set are ordered in ascending with the number of hops, those with smaller number of 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. Note that since the objective function in (5) is the minimization, non-improving moves result in $O_c \leq O_n$ and hence yield $p_a \leq 1$.

The backup path altering process above does not need additional information about the topology. Instead, it simply changes the index of the backup path of a randomly chosen flow. This shields the best possible moving direction which might need additional shortest path calculation. Therefore, we trade off simplicity with optimality in the SA algorithm.

V. Spare provision matrix method

The restoration scheme considered in this paper is the path restoration with link disjoint routes, where the backup path of a flow is link disjointed from its working path. The failure scenarios considered are all single link failures. Given network topology, traffic flows and their working paths in SCA, the objective 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

$\boldsymbol{A} = \{\boldsymbol{a}_r\} = \{a_{rj}\}$	working path-link incidence matrix
$\boldsymbol{B} = \{\boldsymbol{b}_r\} = \{b_{ri}\}$	backup path-link incidence matrix
$\boldsymbol{m} = \{m_r\}$	vector of traffic demands
$\boldsymbol{M} = \text{Diag}(\boldsymbol{m})$	diagonal matrix of traffic demands
$\boldsymbol{C} = \{c_{ij}\}$	spare provision matrix
$\boldsymbol{C}^r = \{c^r_{ij}\}$	contribution of flow r to $\boldsymbol{C}$
$\boldsymbol{s} = \{s_i\}$	vector of spare capacity reservations
W, S	total working, spare capacity
$\eta = S/W$	network redundancy

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

The *spare provision matrix* $\boldsymbol{C} = \{c_{ij}\}_{L\times L}$ can be found in two ways. The first 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 $\boldsymbol{s} = \{s_i\}_{L\times 1}$ in (7). The “row-max” operator returns a column vector, with each entry being the maximum item in each row of a given matrix.

$$\boldsymbol{C} = \boldsymbol{B}^T \boldsymbol{M} \boldsymbol{A} \tag{6}$$

$$\boldsymbol{s} = \text{row-max}\, \boldsymbol{C} \tag{7}$$

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

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

$$\boldsymbol{C} = \sum_{r=1}^{D} \boldsymbol{C}^r \tag{9}$$

From these equations (6)-(9), per-flow based information in $\boldsymbol{A}$ and $\boldsymbol{B}$ is replaced by $\boldsymbol{C}$ 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.

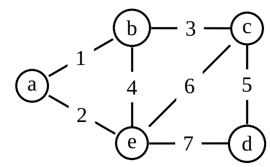

Fig. 1. Example five-node network

TABLE III

Matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ and $\boldsymbol{s}$ in the five-node network

Flows			$\boldsymbol{A}$	$\boldsymbol{B}$		$\boldsymbol{C}$	$\boldsymbol{s}$
r	o_r	d_r	1234567	1234567	l	1234567	
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 in the spare capacity sharing operation. Considering only unit traffic demands, we set $\boldsymbol{M} = \boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix of size D. Then, $\boldsymbol{C} = \boldsymbol{B}^T \boldsymbol{A}$. The matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{s}$ are given in Table III. Note that all the diagonal elements of $\boldsymbol{C}$ are zeroes because the working and backup paths for each flow are link disjointed. The total spare capacity reservation is given by $S = \boldsymbol{e}^T \boldsymbol{s} = 11$, where $\boldsymbol{e}$ is the unit column vector with length L. We can also get the total working capacity reservation $W = \boldsymbol{e}^T \boldsymbol{M} \boldsymbol{A} \boldsymbol{e} = 13$ and the total spare capacity reservation without spare capacity sharing $S' = \boldsymbol{e}^T \boldsymbol{M} \boldsymbol{B} \boldsymbol{e} = 23$. Compared S to S', the total spare capacity reservation is reduced significantly with the spare capacity sharing operation.

The matrix method for more complicated SCA problems, such as cases with directed links, node failures, link modularity, and multi-layer networks, are discussed in [33].

VI. Successive survivable routing

In SSR, each traffic flow routes its working path first, then its backup path in the source node next. The reason of sequential selection of working and backup paths is that they have different levels of importance. The working path is used most of the time while the backup path is evoked only after failures. It is more cost-effective to optimize working paths than backup paths over time. This also makes the protocol implementation simpler. Based on the above matrix method, the SSR implementation in the source node of a flow r is given in Fig. 2.

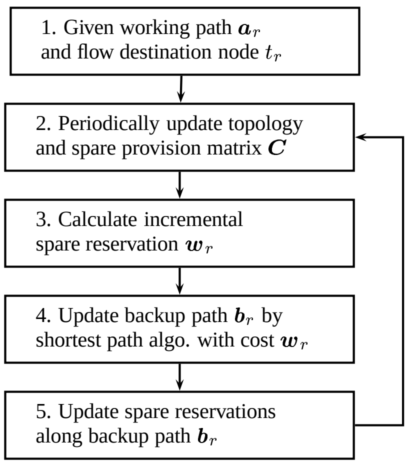

Fig. 2. Flow chart of the SSR algorithm in the source node of flow r

In step 1, each flow is initialized by giving its working path $\boldsymbol{a}_r$ and destination node t_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 network state. In this paper, flows update their backup paths sequentially. This assumption can be achieved when the period is long enough comparing to the state synchronization time and traffic arrival intervals. But too long period might affect the response time of the protocol. How to decide this period is left for future work.

Keeping $\boldsymbol{C}$ up-to-date is important for protocol efficiency. There are two methods in collecting $\boldsymbol{C}$ over the network. The first method is a link-based method. Each link calculates its capacity reservation row vector in $\boldsymbol{C}$ from the working path of a flow whose backup path passing it. The information about working paths will be discarded after that. A node collects all such row vectors from its adjacent links to form a link state packet with packet size of $O(NL)$. The packet is then broadcasted to other nodes.

The second method is a node-based method. The source node of a flow r calculates its contribution matrix $\boldsymbol{C}^r$. Each node aggregates all $\boldsymbol{C}^r$ locally, then disseminates such information though link state packets. Comparing to the link-based method, it does not require to spread the working path information along the backup path, but it increases the size of the link state packets from $O(NL)$ to $O(L^2)$.

Both of the methods collect the state information $\boldsymbol{C}$ at the space complexity of $O(L^2)$. The per-flow based information is not required for backup path routing and reservation. This advantage makes SSR suitable for distributed implementation.

In step 3, we define following notations to calculated the incremental spare reservation vector. Given the spare provision matric $\boldsymbol{C}$, the backup path $\boldsymbol{b}_r$ and $\boldsymbol{C}^r$ for flow r, we define $\boldsymbol{C}^- = \boldsymbol{C} - \boldsymbol{C}^r$ and $\boldsymbol{s}^- =$ row-max $(\boldsymbol{C}^-)$ as the state information after flow r is removed. Assuming an alternate backup path for flow r is $\boldsymbol{b}_r^+$, we have $\boldsymbol{C}^{r+}(\boldsymbol{b}_r^+) = m_r \boldsymbol{b}_r^{+T} \boldsymbol{a}_r$. This produces a spare capacity reservation vector as $\boldsymbol{s}^+(\boldsymbol{b}_r^+) =$ row-max $(\boldsymbol{C}^- + \boldsymbol{C}^{r+}(\boldsymbol{b}_r^+))$. Finally, when $\boldsymbol{b}_r^+ = \boldsymbol{e}$ the *incremental spare reservation* vector for traffic flow r is defined by

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

If the original $\boldsymbol{C}$ is an optimal solution, we know that any alternate backup path $\boldsymbol{b}_r^+$ will result $S^+ \geq S$. The optimal backup path $\boldsymbol{b}_r$ should result the minimum $\boldsymbol{e}^T \boldsymbol{b}_r^+$ among all backup paths $\boldsymbol{b}_r^+ \leq \boldsymbol{e} - \boldsymbol{a}_r$. Hence, the minimization of the incremental spare capacity is a *necessary condition* of the global optimal SCA solution.

In step 4, the possible better backup path is calculated by using the shortest path algorithm with $\boldsymbol{v}_r$ as the link cost. We call this scheme as *Survivable Routing* (SR). It has a low complexity at $O(N \log N)$. This makes the resulted SSR a fast algorithm.

In step 5, if the backup path is changed in the former step, the spare capacity reservations along the path will be updated accordingly. Since the backup path and its spare capacity are not used unless failure happens, it is possible to modify current backup paths as well as the reserved spare capacity to improve the global cost-effectiveness according to the changing traffic requirements and network states.

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

When there is no backup path update in the network, the algorithm stops at a local optimum where no flow can improve the global solution individually. In this way, the SSR reaches an approximation of the global optimum through iterative backup path routing and state information collection procedure.

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 $\bar{d}$ from 2.31 to 4.4 as given in Table IV. Without loss of generality, we assume symmetrical traffic flows between the same node pair are on the same path but reversed directions. All flows have one unit bandwidth demand in the eight networks, i.e., $m_r = 1, \forall r \in \mathcal{D}$. This unit traffic demand is only for easy comparison. For network 3 and 5, the results with demands between one and five units are also given in cases 3b and 5b in Table IV–VI and Fig 11.

The best total spare capacity allocations come from Branch and Bound with hop-limited backup path sets solved in the commercial software AMPL/CPLEX [34], [35]. The computation time for the smaller networks is within hours on a SUN Ultra

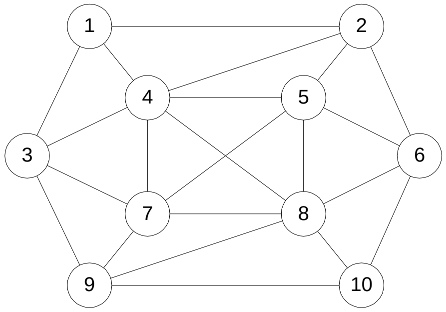

Fig. 3. Network 1 (10 nodes, 22 links, $\bar{d} = 4.4$)

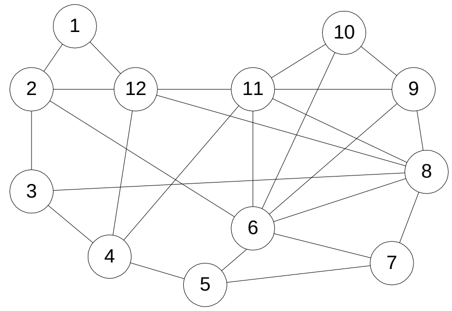

Fig. 4. Network 2 (12 nodes, 25 links, $\bar{d} = 4.17$)

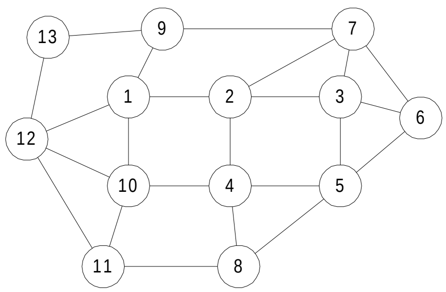

Fig. 5. Network 3 (13 nodes, 23 links, $\bar{d} = 3.54$)

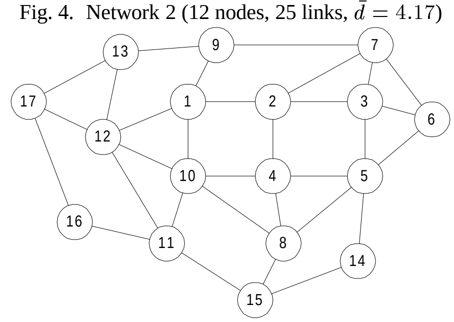

Fig. 6. Network 4 (17 nodes, 31 links, $\bar{d} = 3.65$)

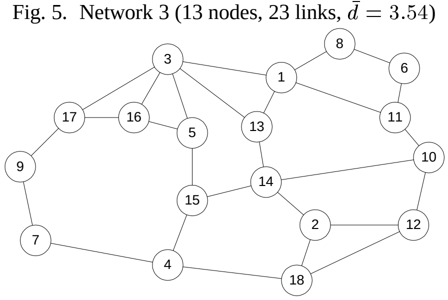

Fig. 7. Network 5 (18 nodes, 27 links, $\bar{d} = 3.00$)

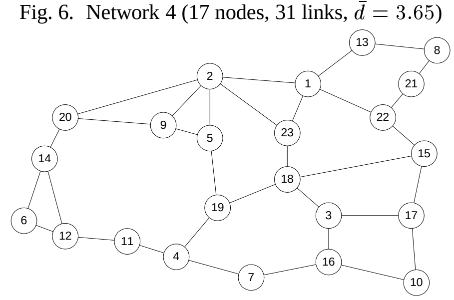

Fig. 8. Network 6 (23 nodes, 33 links, $\bar{d} = 2.87$)

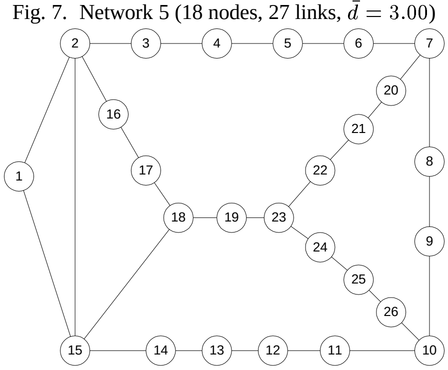

Fig. 9. Network 7 (26 nodes, 30 links, $\bar{d} = 2.31$)

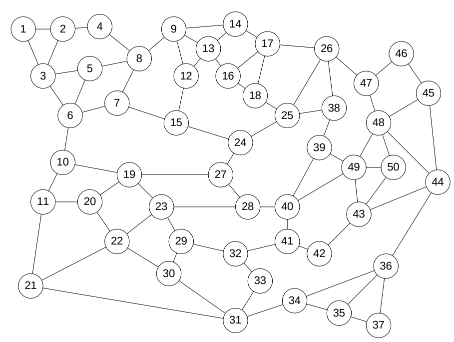

Fig. 10. Network 8 (50 nodes, 82 links, $\bar{d} = 3.28$)

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

TOTAL SPARE CAPACITY

Network	1	2	3	4	5	6	7	8	3b	5b
Working	142	224	324	640	826	1670	2732	11104	972	2430
LP	49.7	100.5	121.5	227	469	1103	1820	-	381	1340
BB	52	102	124	240	472	1122	1820	-	382	1346
SA	54	112	132	246	474	1216	1830	-	-	-
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

TABLE VI

EXECUTION TIME (IN SECOND)

Network	1	2	3	4	5	6	7	8	3b	5b
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	0.5	1	1.2	3.2	3	6.5	7.3	191.8	1.34	3.67
SR	0.2	0.3	0.3	0.6	0.7	1.3	1.8	25.5	0.32	0.73
SPI	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

Enterprise server with 4GB memory and 250MHz UltraSparc CPU. The results from linear programming (LP) relaxation are also given. In LP, the backup path sets include all possible paths[1]. The decision variables $x^{r,p}$ in the InP model of Section III are relaxed to non-integer value. Hence, the results from LP might not be feasible for the InP model but it provides a lower bound of the optimal solution.

Simulated Annealing (SA) algorithm introduced in Section IV is also used to find near optimal solutions. The parameters include the annealing schedule constant $\alpha = 0.9$, the maximum number of no improvement $b = 2000$, and the initial temperature $T_{max} = 100$. In Table VI, the simulated annealing takes less time in the SUN server than BB but it still needs a lot of time for convergence.

For SSR, we assume that the state information can be synchronized immediately and all flows are determined before the algorithm starts. The sequence of the flow updates in SSR is an important factor. Different sequences of flow will results different solution. We use 64 different seeds for generating random number sequences. In this way, we can collect the statistically significant results, the maximum and minimum solutions among 64 results, in Table V. These random seeds are also used in the SR and SPI schemes. The achieved redundancy levels of all schemes except NS are plotted in Fig 11. In SSR, SR and SPI schemes, the minimum redundancies from 64 random cases are the top of the solid columns and the maximums are the top of the error bars. These schemes are executed for all 64 random cases in a Pentium III 533MHz machine. The total execution time for all 64 cases is given in Table VI. RAFT and No Sharing (NS) [11] schemes are run in the same machine, but the execution time recorded is only for single case.

[1] An exception is network 4 where the LP lower bound is not found in one day and the limited path set is used instead.

In experiments of first seven networks, SSR achieves stable solutions after at most four backup path update for all flows. For the larger network 8, the maximum number of update is ten. It means that SSR converges very fast.

Among all the algorithms, BB gives the best feasible solutions. The optimal solutions are bounded narrowly between BB and LP. In the other end, NS does not provide spare capacity sharing and consequently it requires the highest redundancy ratio, which is above 100%.

Among all other schemes supporting spare capacity sharing, RAFT and SPI are the next highest schemes. Their results are close. SPI requires on-line link metrics calculation, while RAFT is much simpler since it uses hop count as link cost. SSR is much lower than SPI and RAFT and slightly lower than SR. It improves the total spare capacity by iteratively updating backup routes upon an SR solution. Furthermore, the results of SSR is slightly higher than SA and BB. SSR is only 1% to 4% more than BB. Considering the small gap from BB to the optimal solution, SSR is very close to the optimal SCA solution as well.

The network topology is also an important factor. The sparser networks such as network 7 have higher redundancy. The difference between SSR and BB results are closer. On the other hand, the denser networks can achieve lower network redundancy around 40% where the difference between SSR and BB results expands up to 4%.

The maximum value and minimum results in 64 different SSR random cases provide ranges from 0.4% to 8.5%. This indicates

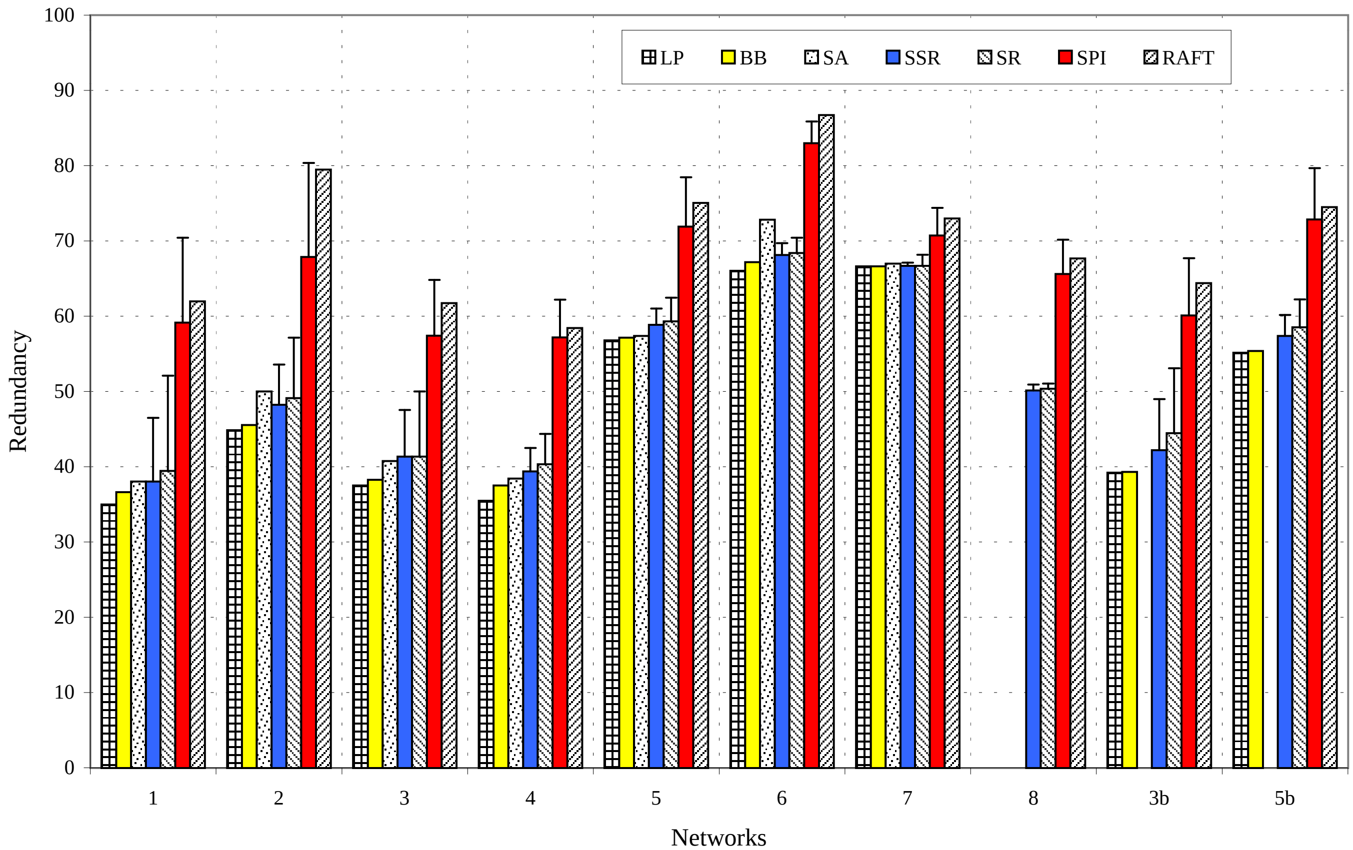

Fig. 11. Comparison of redundancy

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.

Moreover, the computation time for different algorithms are significantly different. BB takes hours to compute in most networks and can not scale to larger networks, such as network 8. SA has faster speed comparing to BB and less optimality. 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 solutions to all networks in seconds and it achieves very good near optimal solutions.

From above results, we conclude that SSR achieves surprisingly good approximation results with very fast speed.

We use the matrix method to explain why SSR achieves such surprisingly good results comparing to RAFT.

First, RAFT routes backup paths through minimum hop paths. The corresponding operation in matrix $\boldsymbol{C}^r$ is to minimize the summation of its elements after the working path is given. Consequently, Minimizing the summation of all elements in $\boldsymbol{C}$ is the objective. This operation is equivalent to minimize the rightmost formula in (11) which is a lower bound of the network redundancy. In Section VII, RAFT can not achieve good redundancy ratio compared to SSR and BB solutions. The reason can be explained as: reducing the lower bound of redundancy does not necessary reduce the achievable redundancy itself.

$$\eta = \frac{S}{W} = \frac{\boldsymbol{e}^T \boldsymbol{s}}{W} \geq \frac{1}{W} \frac{\boldsymbol{e}^T \boldsymbol{C} \boldsymbol{e}}{L-1} \tag{11}$$

In SSR, we utilize the necessary condition discussed after (10) to find a *partial* optimal for a single traffic flow under the given network state. Hence, SSR has better chance in approximating the optimal solution than RAFT. We use "partial" here to distinguish it from the global optimal solution for the multi-commodity flow SCA problem.

One more benefit from using the incremental spare reservation vector in (10) as link cost is the possible iteration procedure to approximate the global optimal solution. The property of the reserved spare capacity is also making this iteration procedure possible. The backup path and its spare capacity are not used unless failure happens. It is reasonable to update current backup paths as well as the spare capacity to improve the cost-effectiveness when the traffic demand and network state are changed. Hence, beyond above survivable routing, we can achieve such iterations by successively updating the backup paths. This leads to the successive survivable routing algorithm.

VIII. Conclusion

This paper first overviews the spare capacity allocation problem. It is an NP-hard multi-commodity flow problem. Next, an Integer Programming model is formulated and solved by a simulated annealing approximation algorithm.

The major contribution of the paper is a two-pronged attack to SCA: model the SCA structure in a novel matrix method approach and construct 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. Based on the SPM, the Successive Survivable Routing (SSR) algorithm is given as an approximation algorithm for SCA. It has the advantages in distributed implementation, easy protocol complexity, and adaptive to traffic fluctuations as well as network evolution. The extensive numerical results show that SSR has advantages in terms of effective approximation to the optimal solutions and fast computation speed. Moreover, we also use the SPM to justify why SSR can achieve better results than the other algorithm RAFT – a necessary condition of the optimal SCA solution was used in SSR.

In conclusion, SSR is an excellent candidate to allocate spare capacity for the survivable network services in the mesh type networks.

ACKNOWLEDGMENT

Thanks to Stella 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 *Proceddings 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, Mar. 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.