Segment Shared Protection in Mesh Communications Networks with Bandwidth Guaranteed Tunnels

Pin-Han Ho, IEEE Member, János Tapolcai, IEEE Member, and Tibor Cinkler, IEEE Member

Abstract – This paper focuses on the problem of dynamic survivable routing for segment shared protection (SSP) in mesh communication networks provisioning bandwidth guaranteed tunnels. With SSP, a connection is settled by concatenating a series of *protection domains*, each of which contains a working and protection segment-pair behaving as a self-healing unit for performing local restoration whenever the working segment is subject to any unexpected interruption. We first discuss the advantages of using SSP – the ability of shortening the restoration time as well as achieving a higher throughput by saving spare capacity required for 100% restorability; then the survivable routing problem is formulated into an Integer Linear Programming (ILP), where the switching/merging node-pair of each protection domain along with the corresponding least-cost working and protection segment-pair can be jointly determined for a dynamically arrived connection request. A novel approach of *arc-reversal transformation* is devised to deal with the situation that the working segments of two neighbour protection domains may overlap with each other by more than a single node. Due to a very high computation complexity induced in solving the ILP, a novel heuristic algorithm is proposed, named Cascaded Diverse Routing (CDR), to allocate protection domains for a connection request by performing diverse routing across a set of pre-defined candidate switching/merging node-pairs. Experiments are conducted on five two-connected network topologies to verify the ILP and the CDR algorithm. We first find out the best diameter of protection domains for the CDR scheme in each network topology. Using the results of best diameters, CDR is compared with two reported schemes, namely PROMISE and OPDA. We demonstrate in the simulation results that the path shared protection schemes are outperformed by the SSP schemes in terms of blocking probability under all possible arrangements in the experiment, and that CDR yields better performance than PROMISE and OPDA due to the extra efforts in manipulating the location of working segments at the expense of longer computation time.

Keywords: segment shared protection (SSP), survivable routing, working and protection paths, integer linear programming, Shared Risk Link Group (SRLG).

I. INTRODUCTION

Survivability has emerged as one of the most important issues in the design of modern communications networks with bandwidth guaranteed tunnels. Survivable routing is recognized as one of the best strategies to equip the networks with service continuity by pre-planning link-disjoint or node-disjoint protection paths for working capacity. With a diversely routed working-protection path-pair, once the working path is subject to any unexpected interruption, the corresponding service can be restored by switching over the working traffic to the protection path such that the failure is circumvented. With the emergence of some commercially applications and delay-sensitive services addressing stringent requirements on data integrity and service continuity, the design of survivable routing algorithms should not only be both capacity- and computation-efficient, but also minimize the possible restoration time for a specific connection, such that the maximum benefits can be gained in the operation of carrier networks.

Segment shared protection (SSP) is one of the best approaches to meet the above design requirements, where a connection is provisioned by concatenating a series of protection domains, each of which contains a working and protection segment-pair behaving as a self-healing unit for performing local restoration when the working segment is subject to any unexpected interruption. As shown in Fig. 1, when the working path segment of protection domain 2 is impaired unexpectedly (e.g., either link E-F, F-G, G-H, or H-I is cut), the restoration is performed locally within protection domain 2 such that the affected flow switches over to the backup segment at node E (called switching node of the protection domain) and merges back to the original working path at node J (called merging node of the protection domain).

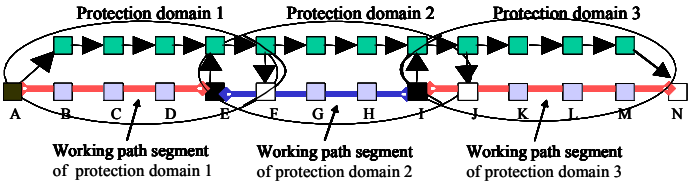


Fig. 1. An illustration of SSP.

Comparing with its counterpart – path shared protection [10-18], segment shared protection has been reported to achieve a better throughput by maximizing the extent of spare capacity resource sharing [2,7,21]. It can also impose a stringent limitation on the restoration time for a specific

application by constraining the length/hop-count of the working and protection segment in each protection domain. The major difficulties of implementing SSP lie in the dependency between the working and spare capacities as well as the exponentially enlarged design space with the network size in identifying a set of switching/merging node-pairs to form protection domains for a connection request. Thus, most previous studies focus on heuristic approaches to solve the problem. In [2], a framework known as Short Leap Shared Protection (SLSP) is proposed, which implements SSP by pre-assigning a series of switching/merging node-pairs along a given working path. The capacity-efficiency with SLSP is expected to be further improved if the working and backup segments can be jointly determined. In [3], an algorithm is developed to find the working path first followed by its backup path segments. The study is characterized by the fact that the spare capacity sharing is not considered until the physical routes of the backup path segments are defined. In [4] and [5], the authors propose two similar dynamic algorithms to switchover working traffic for each link from its immediate upstream node and merge back to the original path at any of the downstream nodes. Both studies do not impose any limitation on the lengths of the backup paths, and may not be able to guarantee the restoration time when a failure occurs. Note that the unnecessarily lengthy backup paths have been witnessed to impair the overall performance even if they share spare capacity with the other backup paths [10,12].

The study of [6] provides an algorithm for computing QoS paths with restoration, which is characterized by considering multiple link metrics in searching the working and protection segments. This study does not consider resource sharing and adopt exhaustive searching for those protection segments for the working path. The study in [7] proposes an Integer Linear Programming (ILP) formulation for SSP to determine the switching/merging node-pairs along the working path given in advance. The algorithm is characterized by the efforts of inspecting all possible number of protection domains for a connection request and all possible combinations of allocation under each given number of protection domains. Since the formulation needs to have the working path first before the protection segments can be derived, the solution may be far from the optimal if the working path is not well selected. The study in [8] takes a very similar approach to that in [7]. The algorithm finds a protection segment for each link along the working path given in advance, in which "backtrack" by D hops is allowed, where D can be an arbitrary positive integer or infinity. In the study of [9], a heuristic algorithm is proposed for segment shared protection, called Optimal Protection Domain Allocation (OPDA), where a graph transformation algorithm, called *transferred graph of cycles*, is devised. Since the algorithm needs to enumerate and align simple cycles in the network topology as candidate cycles for a given working path, it may impair the capacity efficiency if only a limited number of cycles can be inspected within the allowed computation time. It is notable that all of the above schemes deal with work and protection paths separately, which are also known as Active Path First (APF) [10].

To jointly determine a working and protection path-pair, some studies focus on developing a programming-based solution [4,11-13]. One of the key issues in formulating the problem under the *complete routing information scenario* [4] (i.e., the algorithm is aware of all the working and protection paths along each link) is that the routing of the working path and the dependency between working and spare capacity along each link must be handled in a single step. Besides, the consideration of multiple states for the protection path to consume spare capacity along each link may easily leave the formulation nonlinear [4]. Therefore, to our best knowledge a linear programming-based solution that can optimally solve the least-cost working and shared protection path-pair according to the current link-state has never been reported before, let alone the case of segment shared protection, which is considered much more complicated.

The study in [4] provides an Integer Programming (InP) in the complete routing information scenario for solving the end-to-end working and shared protection path-pair according to the current link-state. The formulation is nonetheless nonlinear and cannot be solved by most of the commercially available LP solvers. The study in [11] provides a heuristic-based ILP formulation to deal with the problem, where two scaling parameters are introduced to avoid the nonlinearity possibly induced when multiple states of spare capacity along each link are considered. The study in [12,13] focuses on the complete routing information case and provides an ILP formulation that can solve the problem with two states of spare capacity along each link for the protection path – either sharable or non-sharable, where the link bandwidth constraint can never be addressed. It is clear that an ILP formulation for segment shared protection considering all possible states of spare capacity consumption along each link in the complete routing information scenario has never been seen before.

This paper contributes in formulating the problem into an ILP working under the complete routing information scenario, in which the location of the switching/merging nodes and the corresponding working and protection segment-pairs for a connection request can be derived in a single step according to the current link-state. This is the first linear formulation that can handle the dependency between working and spare capacity for segment shared protection, where a novel method of arc-reversal transformation is devised to deal with the

situation that working segments of two neighbour protection domains may overlap with each other by more than a single node (as shown in Fig. 1). To avoid the nonlinearity possibly incurred when dealing with the multiple states for a protection segment to take spare capacity, a graph transformation technique is devised to facilitate the formulation.

Although the ILP can yield the least-cost (or optimal) working and protection segment-pairs for a connection request, the solving of the ILP nonetheless takes an intolerably large amount of computation time. Therefore, this paper also introduces a novel heuristic algorithm for solving the problem, called Cascaded Diverse Routing (CDR), aiming to trade the optimality in performance with the computation complexity. The basic idea of CDR is to pre-define a set of candidate switching/merging node-pairs, between each of which a diverse routing algorithm, called Iterative Two-Step-Approach (ITSA) [12], is conducted to find the corresponding working and protection segment pair. We compare CDR with the heuristic approaches reported in [7] and [9] through simulation, namely PROMISE and OPDA, and examine the *offset of optimality* in each case referring to the solutions from the ILP formulation.

This paper is organized as follows. In Section II, an overview on segment shared protection and spare capacity resource sharing is conducted, where the cost functions for both working and protection segments are defined. In Section III, the ILP formulation for solving the segment shared protection problem is presented. Section IV presents the heuristic algorithm, Cascaded Diverse Routing (CDR). In Section V, CDR is compared with two other reported schemes through simulation and is verified by the results from the ILP solution. Section VI concludes this paper.

II PROBLEM DEFINITION

A *Concepts of Shared Risk Link Group (SRLG)*

Shared Risk Link Group (SRLG) is defined as a group of network elements (i.e., either links, nodes, physical devices, software/protocol identities, or a mix of which, etc) subject to the same risk of single failure. In practical cases an SRLG may contain multiple seemingly unrelated and arbitrarily selected links/nodes. We define that a working path is *involved* in an SRLG if it traverses through any network element that belongs to the SRLG. A path may be involved in multiple SRLG's. A working path is said to be *SRLG-disjoint* with its protection path if the two paths are not involved in any common SRLG. In this study, the SRLG-disjointedness for a working and protection path-pair is the major effort of achieving 100% restorability for the working data flows under the single failure scenario.

Without loss of generality, this study focuses on the case that each arc in the network topology is an SRLG. Under such a premise, it is easy to see that a working path traversing through H hops will be involved in no more than H different SRLG's. We raise the assumption that the probability of each physical conduit to be subject to a failure is independent. In other words, to achieve 100% restorability, it is sufficient and necessary that every link traversed by the working path is protected by at least one link-disjoint protection path.

In this study, all the working paths are assumed to be loop-less. A working path may contain multiple protection domains, each of which has a protection path. The spare capacity taken by backup path segments is called *spare channels* that are only reserved but not configured during the normal operation. Therefore, the spare channels can also be used by some best-effort traffic that can tolerate a service interruption. In the event that a failure occurs that interrupts a working path (such as a fibre-cut or loss of signal due to the failure of any network element), the switching fabric structures in the nodes along the corresponding protection path are configured by prioritized signalling followed by traffic switchover to recover the original service supported by the working path. Therefore, the protection paths of different working paths can reserve the same spare channels if the working paths are not involved in a common SRLG, and are considered to share the same risk of single failure. In this study, two working paths share the same risk of single failure if and only if they take any common arc in the network. In other words, whether or not two protection paths can share a spare channel depends on the physical location of their working lightpaths. The dependency is the reason for the existence of SRLG constraint [1]. A simple example is shown in Fig. 2. W1 and P1 form a working and protection path-pair. The protection path of W2 (any other working path) should exclude the possibility of using any of the spare channels taken by P1 because W2 traverses link A-B, which shares the same risk of a single failure with W1.

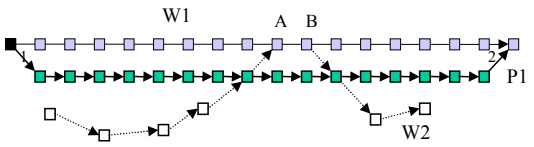


Fig. 2. An example to illustrate the SRLG constraint.

For language precision, in the following context a "link" is always directional while an "arc" is un-directional composed of two directional links of the same ending nodes. We assume that a link is physically bundled and is comprised of several independent communication channels that provision data flows. Therefore, a failure defined in this paper is limited to a link cut that is possibly caused by a rodent bite or careless construction/maintenance efforts. We take an assumption of *single failure scenario*, where the algorithm only deal with the

situation that a single link in the total network is unexpectedly interrupted at a moment.

B An Overview on SSP

The advantage of using SSP compared with path shared protection lie in not only the reduction of restoration time, but also the achievement of larger degree of sharing. The following is an example to show the resultant capacity saving and reduced restoration time in using SSP compared with path shared protection. In Fig. 3, the network contains working path W1 (A-B-C-D-E-F-G) and its link-disjoint protection path along (A-H-C-J-E-K-G). Let W2 (C-D-E) is being allocated with its protection path. In case path shared protection is adopted, as shown in Fig. 3(a), the spare capacity taken by the protection path of W1 can never be shared by the protection path of W2 since both W1 and W2 are involved in a common SRLG. For SSP shown in Fig. 3(b), on the other hand, W1 is segmented into multiple segments, each of which is assigned with a switching/merging node-pair and a protection segment. For example, for the second protection domain, each switching and merging node is C and E, respectively. Therefore, the protection path of W2 can share the spare capacity taken by the protection segments in the first and the third protection domain. In this case, the number of working paths in an SRLG is reduced such that the total amount of non-sharable spare capacity in the network for a specific working segment is reduced, which yields better throughput. Because the restoration can be performed locally in each protection domain, the propagation time of signalling messages can be largely reduced.

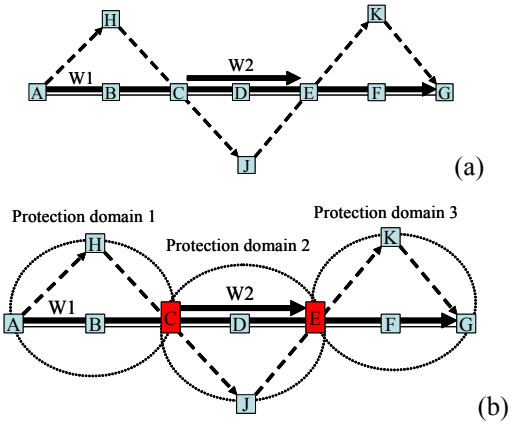


Fig. 3. A comparison between the path shared protection and SSP; (a) path shared protection; (b) SSP with switching-merging pairs (A, C), (C, E), and (E, G) for the first, second, and third protection domain, respectively.

In this study, the link-based shared protection [2] is taken as a special case of SSP, in which every link along the working path behaves as a working path segment with a switching/merging node-pair for the corresponding protection segment. The overhead in using segment shared protection compared with path shared protection is as below. Firstly, larger computation complexity is taken to solve the problem due to the efforts in determining the switching/merging node-pair of each protection domain. Secondly, a new suite of signalling mechanism must be defined.

The signalling effort with SSP is briefly described as follows. After a fault on the working path occurs, it is localized immediately by the downstream neighbour node, which notifies the switching node of the protection domain to activate a traffic switchover. For the example in Fig. 1, a fault on link C-D is localized by its immediate downstream node D. A fault on link E-F is localized by the immediate downstream node F. In the former case, node D sends notification indicator signal (NIS) to notify node A and F that a fault occurred in their protection domain. In the latter case, node F sends an NIS to node E and J for a fault notification. In the case that a failure occurs to the link or node covered by two neighbour protection domains (e.g., link E-F), the protection domain close to the source node is in charge of the failure. After receiving the NIS, the merging node (i.e. A or E) immediately sends a wake-up packet to activate the configuration of switching fabric in each node along the corresponding backup segment of the corresponding protection domain, and then the traffic can be switched over to the protection path.

With the above signalling mechanism, it is clear that every node must additionally keep track of the switching node of each working path traversing through the node in the corresponding protection domain. Fault localization [22] is necessary such that the downstream node of a failure can activate the failure recovery process, in which a higher requirement on hardware responsiveness and control complexity is needed. The largely increased computation complexity in using SSP compared with path shared protection is also a non-trivial problem that should be solved before the scheme can be practically applied.

C Definition of Cost Functions

This section defines cost function and the link-state for solving a protection segment of a working path segment (or termed *spare link-state*). Given a network $G(N,E)$ with N and E being the set of nodes and directional links, respectively. The capacity along link j, $\forall j \in E$, can be categorized into the following three types:

1) *Free capacity* (denoted as f_j), which is the link capacity that can be reserved as either working or spare capacity.
2) *Spare capacity* (denoted as v_j), which is the link capacity reserved by some backup segment(s).
3) *Working capacity*, which is the link capacity already taken by some working path, and cannot be taken for any use until the corresponding working path is torn down.

The cost function for finding working path segment in the

k-th protection domain (denoted as W_k) with bandwidth $b(W)$ is as follows:

$$cw_j = \begin{cases} \infty & \text{if link } j \text{ is not reservable} \\ b(W)\cdot c_j + \varepsilon & \text{otherwise} \end{cases} \quad (1)$$

where c_j is the cost for each unit of bandwidth taken by a working path along link j, and ε is a small number defined as $\varepsilon = (\min_{(a,b)\in E} c_j)/|E|$ in this study. The link cost c_j is custom-designed, and can either be a constant (e.g., simply the bandwidth demand of the connection request for each hop), or take dynamic network traffic into consideration (e.g., the maximum reservable bandwidth along link j). In this study, the fact that link j is not reservable by W_k can only be due to $b(W) > f_j$, where f_j is the *free capacity* along link j (as illustrated in Fig. 4). The total cost of the working path is $\sum_{j\in W}\left(b(W)\cdot c_j + \varepsilon\right)$. The purpose of additionally imposing the small number ε in the cost function is to match the cost of backup segments, which will be defined later.

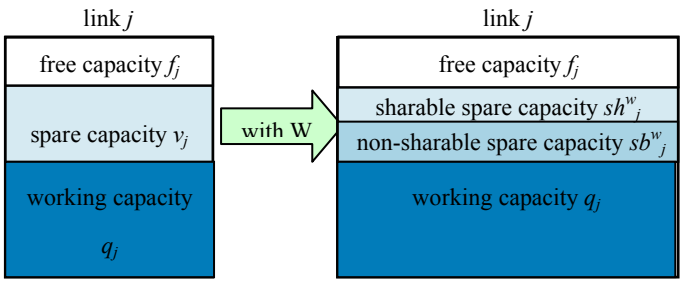


Fig. 4. An illustration of categories of capacity along link j.

For solving the backup segment of W_k, we need first to define the corresponding spare link-state and cost function. With the presence of W_k, the spare capacity along link j can be further categorized into the following two types:

1) *Sharable spare capacity* (denoted as $sh_j^{W_k}$), which is the link capacity that has been reserved by some other backup segment(s), and is sharable to the backup segment of W_k.
2) *Non-sharable spare capacity* (denoted as $sb_j^{W_k}$), which is the link capacity that has been reserved by some other protection paths, and is not sharable to the protection path of W due to the SRLG constraint. Note that $v_j = sb_j^{W_k} + sh_j^{W_k}$, which is the total spare capacity along link j.

The protection path may traverse through link j in any one of the following three states: (1) the case where the link has sufficient sharable spare capacity (i.e., $sh_j^{W_k} \ge b(W)$), in which the backup segment can take this link with the smallest cost (denoted as ε in this study); (2) the case where $f_j + sh_j^{W_k} \ge b(W) > sh_j^{W_k}$, and the backup segment must partly (or totally) take free capacity along this link with an extra cost. In this case the spare link-state is $b(W)\cdot sr_j^{W_k}\cdot c_j + \varepsilon$, where $sr_j^{W_k}$ is a [0,1] scaling parameter determined by the location of W_k and will be defined later. (3) The link does not have sufficient sharable spare capacity and free capacity (i.e., $f_j + sh_j^{W_k} < b(W)$), in which the backup segment cannot traverse through this link by any means. In this case the cost is ∞. Due to the dependency between the working and spare capacity in the network, the parameters $sr_j^{W_k}$, $sh_j^{W_k}$, and $sb_j^{W_k}$ cannot be defined until the presence of W.

In this study, $sr_j^{W_k}$ is defined as $1 - sh_j^{W_k}/b(W)$ for any link $j \in E$. It is clear that $sr_j^{W_k}$ is 1 if there is not any sharable spare capacity available along link j and is approaching to 0 if $sh_j^{W_k}$ is close to $b(W)$. In the former case (i.e., the case of $sr_j^{W_k} = 1$), the cost for the backup segment to take this link is $b(W)\cdot c_j + \varepsilon$, which is the same as that for the working path since all the reserved bandwidth has to be from the free capacity region as shown in Fig. 4. The spare link-state for the backup segment of W_k can be expressed as:

$$cp_j^{W_k} = \begin{cases} b(W)\cdot c_j \cdot sr_j^{W_k} + \varepsilon & \text{if } sh_j^{W_k} + f_j \ge b(W) > sh_j^{W_k} \\ \varepsilon & \text{if } sh_j^{W_k} \ge b(W) \\ \infty & \text{if } sh_j^{W_k} + f_j < b(W) \end{cases} \quad \text{for } j \in E,\ j \notin W_k \quad (2)$$

Fig. 5 shows the three situations defined in Eq. (2). In Fig. 5(b) and Fig. 5(c), the backup segment of W_k may partly take the free capacity region and the sharable spare capacity region; therefore, the link cost is $b(W)\cdot c_j \cdot sr_j^{W_k} + \varepsilon$, which is shown in the first condition in Eq. (2). In Fig. 5(a), the backup segment can have all $b(W)$ in the sharable spare capacity region, therefore, the cost is ε, as shown in the second condition in Eq. (2). In Fig. 5(d), the link cost is infinity because the backup segment of W_k cannot be supported by the residual capacity of the link, which is shown in the third condition in Eq. (2). Note that the protection path is assumed to take sharable spare capacity along a link whenever there is any sharable spare capacity available. If there is not enough sharable spare capacity along this link to cover the total bandwidth demand for protecting W_k (i.e., $b(W)$), the backup segment takes free capacity after considering all the sharable spare capacity.

Note that the adoption of the small constant ε is to keep the continuity between the first and second condition in Eq. (2). In this case, the cost of link j is set to ε as $sh_j^{W_k} = b(W)$ for both of the conditions. This is also the reason we impose ε in the cost function for the working path shown in Eq. (1), in which the cost for the working and backup path segments to take free capacity can match each other.

Our objective is to determine $cp_j^{W_k}$ in Eq. (2) – the spare link-state that defines the cost of the backup segment of W_k passing through link j, in which $sh_j^{W_k}$ is the only variable that must be figured out (or equivalently, $sb_j^{W_k}$ since

$v_j = sb_j^{W_k} + sh_j^{W_k}$). Note that $sh_j^{W_k}$ and $sb_j^{W_k}$ are network-wide link-state specific to the presence of W_k. Any link $l \in W_k$ is, in turn, traversed by a set of working path segments denoted as D_l, which are also the working path segments currently involved in a common SRLG of link l with W_k.

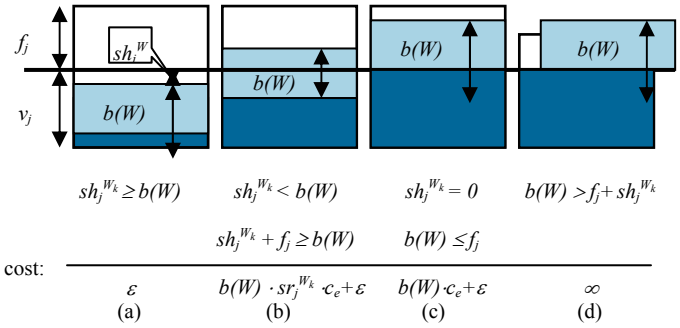


Fig. 5. The possible situations of a different cost function defined in Eq. (2). f_j denotes the amount of "free spare capacity while $sh_j^{W_k}$ for "sharable spare capacity"

The derivation of $sb_j^{W_k}$ and $sh_j^{W_k}$ can also be expressed in a matrix form, which is a graceful expression for determining spare capacity along each link by Y. Liu [15]. In this case, we define the working and protection path-link incidence matrices as A^{W_k} and B^{W_k}, in which A^{W_k} is a $|D_{W_k}| \times |E|$ array containing all the working path segments that are involved in a common SRLG (i.e., take any common physical link) with W_k, while B^{W_k} is an $|D_{W_k}| \times |E|$ array containing all backup segments corresponding to the working paths segments in D_{W_k}. Here we define $D_{W_k} = \bigcup_{l \in LW_k} D_l$, where $\bigcup$ is a union operation.

The spare provision matrix for W_k is defined as $C^{W_k} = (B^{W_k})^T \cdot A^{W_k}$, which is a $|E| \times |E|$ matrix. Applying a MAX operation upon each row of C^{W_k} will yield a $1 \times |E|$ vector $\underline{SB}^{W_k}$, which keeps the amount of non-sharable spare capacity along each link provided with the working path segment W_k. The $1 \times |E|$ vector $\underline{SH}^{W_k}$, which keeps the amount of sharable spare capacity along each link provided with the working path segment W_k, can thus be derived by referring to the relationship $\underline{SH}^{W_k} = \underline{V} - \underline{SB}^{W_k}$, where $\underline{V}$ is a $1 \times |E|$ vector recording the amount of spare capacity along each link.

III LINEAR FORMULATION FOR SEGMENT SHARED PROTECTION

This section introduces a linear formulation for the segment shared protection problem. Our approach is to find a path Q, called *mass protection path*, which is composed of all the backup segments and some links along the working path. A simple example is shown in Fig. 6, where Q is (s-a-b-c-e-d). The first protection domain is formed by the working and protection segments (s-c-b) and (s-a-b), respectively; while the second is formed by (c-b-d) and (c-e-d), respectively. The allowance of overlapping between the working segments of two neighbour protection domains is to explore the largest design space so as to guarantee the optimality of the derived solution. Note that Q may contain loops to reflect the fact that spare capacity sharing can happen between two protection segments of different protection domains.

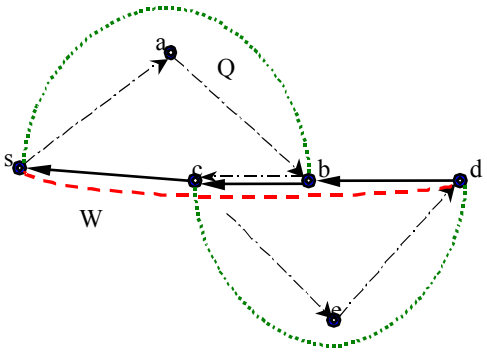


Fig. 6. Design of mass protection path Q.

Let the network be denoted as $G(N,E)$, where N and E is the set of nodes and directional links in the network, respectively. Let the source and the destination of the upcoming connection request, W, be denoted as s and d. Three residual graphs are defined to facilitate the solving of this problem, each of which carries one or a few variables for the identification of the working and protection segment-pairs. The graph for solving the working segments is denoted as $G_w(N,E_w)$ and is composed of links with $f_j \geq b(W)$ for $j \in E_w$ (x, lx and δx variables in the following formulas are assigned to this graph). The second residual graph is denoted as $G_p(N, E_p)$, which is to facilitate solving the protection segments. We need this graph to record the spare link-state because working and protection path take different suites of link-state with shared protection. This graph is composed of the links where the amount of free capacity f_j plus that of the spare capacity v_j is larger than or equal to $b(W)$. (i.e., $b(W) \leq f_j + v_j$ for $j \in E_p$) (δy variable in the following formulas is assigned to this graph).

The third residual graph $G'_p(N, E'_p)$ is composed of all the links in E_p along with the links of E_w in a reversed direction. The inclusion of the links of E_w in a reverse direction into E'_p is called *arc-reversal transformation* similar to the graph technique adopted in the Suurballe's algorithm [19]. With $G'_p(N, E'_p)$, we will be able to handle the reverse arcs caused by the overlapping between Q and W, such that the route of Q can be identified. The variables defined in this graph are y' and ly', which will be discussed in detailed later. It is clear that we have the following relationship: $\underline{E}_p = E_p \bigcup reversed(E_w)$. Therefore, E'_p contains *forward arcs* denoted as (a,b), which are due to the links of E_p, as well as the *reversed arcs* denoted as $(\overleftarrow{a,b})$, which are due to the reversal of link (a,b) in E_w. In

the latter case, the reversed links direct from node *b* to node *a* for $a,b \in E_w$. The transformation yields a fact that if there is a bi-directional link between node *a* and *b* with the amount of free capacity larger than or equal to *b*(*W*), then G'_p will have four links between node *a* and *b* (i.e., $\overrightarrow{a,b}$, $\overleftarrow{a,b}$; $\overrightarrow{b,a}$, $\overleftarrow{b,a}$), in which two of them direct to node *b* (i.e., $\overrightarrow{a,b}$; $\overleftarrow{b,a}$) and the other two direct to node *a* (i.e., $\overrightarrow{b,a}$; $\overleftarrow{a,b}$). If the amount of free capacity is less than *b*(*W*), and the amount of free plus spare capacity is larger than or equal to *b*(*W*) (i.e., $f_j < b(W) \le f_j + v_j$), then G'_p will have two directional links between *a* and *b* (i.e., $\overrightarrow{a,b}$; $\overrightarrow{b,a}$); otherwise, there is no link between node *a* and *b* in G'_p. Please refer to Fig. 7 for an illustration of the graph transformation.

In the formulation, all the three graphs (G_w, G_p and G'_p) are considered and indexed in an array during the implementation, and we have to keep track of the index of each directional link on different graphs even though they are in the same direction and at the same physical location. For example, we must distinguish between $(\overrightarrow{a,b}) \in E'_p$ and $a,b \in E_w$, and between $\overleftarrow{a,b} \in E'_p$ and $a,b \in E_p$, and between $a,b \in E_p$ and $a,b \in E_w$, in the array keeping the indexes of the links.

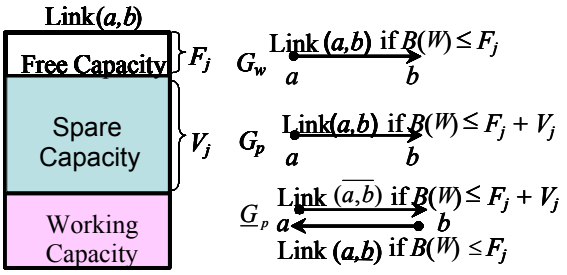


Fig. 7. Graph transformations for the links in *G,* which yields the graphs G_w, G_p and G'_p, respectively.

To introduce the target function of the proposed ILP formulation, the following two flow indicators are defined in each graph of G_w and G'_p: $x_{a,b}$ is a binary variable with a size of $|E_w|$ defined in graph G_w, while $y'_{u,v}$ is an integer variable ranged [0..k_{max}] with a size of $|E'_p|$ defined in graph G'_p, where k_{max} is the maximum number of protection domains that can be possibly handled in the problem. The target function is as follows:

$$\text{Minimize} \sum_{(a,b)\in E_w} b(W)\cdot c_{a,b}\cdot x_{a,b} + \sum_{(u,v)\in E_p} \left(b(W)\cdot c_{u,v}\cdot r_{u,v} + \varepsilon\right)\cdot y_{u,v} \qquad (3)$$

where $c_{a,b}$ is the cost per unit of working bandwidth to reserve link (*a*,*b*), which is equivalent to c_j in Eq. (1), ε is a small constant that can be set such that $\varepsilon \cdot |E_p| < \min_{(a,b)\in E_p} c_{a,b}$, which is same as that defined in Eq. (2). $y_{u,v}$ is a mirror binary variable of $y'_{u,v}$ to facilitate the calculation of the total cost caused by the protection segments, and will be further discussed later.

Each of $x_{a,b}$ and $y_{u,v}$ indicates the number of times the working and mass protection paths traverses $(a,b) \in E_w$ and $(u,v) \in E'_p$, respectively. The reason of setting $y'_{u,v}$ an integer instead of a binary variable is that path *Q*, which is composed of the protection segments of all the protection domains, may have loops in case the protection segments of two different protection domains traverse through the same link. Therefore, $y'_{u,v}$ may larger than 1. Therefore, the concatenation of all links with $x_{a,b}$ = 1 yields *W*, while the concatenation all the links with $y'_{u,v} \ge 1$ forms *Q*. It is clear that the overlapped links between *W* and *Q* should not be considered when calculating the cost for the corresponding protection segments in the target function. Besides, we should count once for each link traversed by *Q* even if *Q* traverses any link by multiple times due to the possible spare capacity sharing between protection segments of any two protection domains for the connection. Therefore, a transformation is required from the integer variable $y'_{u,v}$ (defined in G'_p) into a new binary variable, denoted as $y_{u,v}$ (defined in G_p). This transformation can be simply done by filtering out those links taken by *Q* which are defined in G'_p but not in G_p. This filtering can be done with the following linear formula:

$$k_{\max}\cdot y_{u,v} - y'_{\overrightarrow{u,v}} \ge 0 \quad \text{for } \forall (u,v)\in E_p, \forall (\overrightarrow{u,v}) \in E'_p$$

In the transformation, $y_{u,v}$ is zero if $y'_{\overrightarrow{u,v}}$ is zero, and is 1 if $y'_{\overrightarrow{u,v}}$ >0. As for $r_{u,v}$, it is for the protection segment taking per-unit of spare capacity along link (*a*,*b*), which depends on the value of $y_{u,v}$ and has the same physical meaning as that of the parameter sr_j^W defined in Eq. (2) if link (*u*,*v*) is equivalent to link *j*. This statement will be discussed and verified later in this section.

The target function is subject to the following constraints:

$$\sum_{(a,b)\in E_w} x_{a,b} - \sum_{(b,a)\in E_w} x_{b,a} = \begin{cases} 1 & s == a \\ -1 & d == a \\ 0 & \text{otherwise} \end{cases} \quad \text{for } a \in N \qquad (4)$$

$$\sum_{(a,b)\in E'_p} y'_{a,b} - \sum_{(b,a)\in E'_p} y'_{b,a} = \begin{cases} 1 & s == a \\ -1 & d == a \\ 0 & \text{otherwise} \end{cases} \quad \text{for } a \in N \qquad (5)$$

Each Eq. (4) and (5) is the flow conservatin constraint for the working and mass protection paths, respectively.

It is important to note that $x_{a,b}$ and $y'_{\overrightarrow{a,b}}$ will be exclusive in terms of the physical links they take. However, a link can be taken by $y'_{\overrightarrow{a,b}}$ in a reversed direction only if $x_{a,b}$ pass through it. Besides, each reversed arc can be used only once since the algorithm only allows two working segments overlapped. The above statements can be formulated into the following two constraints:

$$k_{\max} \cdot x_{a,b} + y'_{\overline{a,b}} \le k_{\max} \quad \text{for } \forall (a,b) \in E_w, \forall (\overrightarrow{a,b}) \in E'_p$$
$$x_{a,b} \ge y'_{\overline{a,b}} \quad \text{for } \forall (a,b) \in E_w, \forall (\overleftarrow{a,b}) \in E'_p$$

where $y'_{\overline{a,b}}$ represents the reversed link of E'_p. Under the above two constraints, Q has to be disjoint from W except for those arcs of W being reversed (see Fig. 6). These two constraints not only assert the disjointedness of the working and the corresponding protection segment, but also facilitate the indication of the switching/merging nodes for each protection domain along W.

A pair of variables, $lx_{a,b}$ (with a size of $|E_w|$) and $ly'_{a,b}$ (with a size of $|E'_p|$), is assigned to each link along W and Q, respectively, such that the first link from the source has a label of 1; and if a protection domain ends or starts at a node, the labels of the following arcs will be increased by 1. This labelling method is similar to that proposed in [7]. Let k_{max} be the number of protection domains with an upper bound $|N|-1$. In solving the formulation, the value of K should be set to the upper bound: $|N|-1$ to guarantee the derivation of the optimal solution. If the value of k_{max} is set less than k_{opt} (i.e., the number of protection domain in the optimal solution), the LP solver cannot return an optimal solution although we have a smaller problem size.

For $lx_{a,b}$, we have the following constraints:

$$(2 \cdot k_{\max} - 1) \cdot x_{a,b} \ge lx_{a,b} \ge 0 \quad \text{for } \forall (a,b) \in E_w \tag{6}$$

$$\sum_{\substack{(a,b)\in E_w \\ a\neq d}} lx_{a,b} - \sum_{\substack{(b,a)\in E_w \\ a\neq s}} lx_{b,a} \ge \sum_{\substack{(\overrightarrow{a,b})\in E'_p \\ a\neq d}} y'_{\overline{a,b}} + \sum_{\substack{(\overleftarrow{b,a})\in E'_p \\ a\neq s}} y'_{\overline{b,a}} + \sum_{\substack{(a,b)\in E_w \\ a\neq d}} k_{\max} \cdot x_{a,b} + \sum_{\substack{(b,a)\in E_w \\ a\neq s}} k_{\max} \cdot x_{b,a} - 2 \cdot k_{\max} \quad \forall a \in N \tag{7}$$

$$\sum_{\substack{(a,b)\in E_w \\ a\neq d}} lx_{a,b} - \sum_{\substack{(b,a)\in E_w \\ a\neq s}} lx_{a,b} \le \sum_{\substack{(\overrightarrow{a,b})\in E'_p \\ a\neq d}} y'_{\overline{a,b}} + \sum_{\substack{(\overleftarrow{b,a})\in E'_p \\ a\neq s}} y'_{\overline{b,a}} \quad \forall a \in N \tag{8}$$

$$\sum_{(s,b)\in E_w} lx_{s,b} = 1 \quad \forall b \in N \tag{9}$$

The constraint in Eq. (6) ensures that $lx_{a,b}$ is upper-bounded by $(2k_{max}-1)$, and is nonzero only if W passes through (a,b). To verify Eq. (7) and Eq. (8), the following four situations are defined for a node (not the source or destination) taken by W: (a) Q merges back to W at the node; (b) Q switches out of W at the node; (c) Q merges back and switches out of W at the node; (d) otherwise. Eq. (7) and (8) behave as a special type of flow conservation constraint upon the net change of $lx_{a,b}$ for node a in the network, which is denoted at the left-hand side of the equations. In Eq. (7), the value of $lx_{a,b}$ of node a along W increases by 1 in the case of situations (a) and (b), and increases by 2 in the case of situation (c), and is unchanged otherwise. The constraint on the net change in terms of $lx_{a,b}$ has a lower bound specified at the right-hand side of the equation, where the term $\sum_{(a,b)\in E_w, a\neq d} k_{\max} \cdot x_{a,b} + \sum_{(a,b)\in E_w, a\neq s} k_{\max} \cdot x_{a,b} - 2 \cdot k_{\max}$ checks if node a is taken by W. It is clear that the term is 0 if node a is traversed by W, and is $-2k_{max}$ otherwise. Therefore, if node a is not taken by W, Eq. (7) automatically holds.

The four cases specified in Eq. (7) are presented as follows. In the case of (a), the increase of $lx_{a,b}$ is 1 since $\sum_{(\overrightarrow{a,b})\in E'_p, a\neq d} y'_{\overline{a,b}} = 0$ and $\sum_{(\overleftarrow{b,a})\in E'_p, a\neq s} y'_{\overline{b,a}} = 1$ (since Q merges back to W at node a). In case (b), the increase of $lx_{a,b}$ is still 1 because $\sum_{(\overrightarrow{a,b})\in E'_p, a\neq s} y'_{\overline{a,b}} = 0$ and $\sum_{(\overleftarrow{b,a})\in E'_p, a\neq d} y'_{\overline{b,a}} = 1$. In situation (c), increase of $lx_{a,b}$ is 2 since $\sum_{(\overrightarrow{a,b})\in E'_p, a\neq d} y'_{\overline{a,b}} = \sum_{(\overleftarrow{b,a})\in E'_p, a\neq s} y'_{\overline{b,a}} = 1$. If node a is neither a switching nor a merging node (but it is taken by W), the right-hand-side of Eq. (7) becomes 0, in which no change upon $lx_{a,b}$ is required.

Eq. (8) basically has the same working principles as Eq. (7) except that when node a is not taken by W, the right-hand side becomes 0 instead of $-2k_{max}$. Both Eq. (7) and (8) constrain the net change of the value of $lx_{a,b}$ to be either 0, 1, or 2, for any node a taken by W, depending on how path Q switches out and merges back to W. Eq. (9) sets $lx_{a,b}$ to 1 if node a is the source node.

For $ly'_{a,b}$, we have the following constraints:

$$(2 \cdot k_{\max} - 1) \cdot y'_{a,b} \ge ly'_{a,b} \ge 0 \quad \text{for } \forall (a,b) \in E'_p \tag{10}$$

$$\sum_{\substack{(a,b)\in E'_p \\ a\neq d}} ly'_{a,b} - \sum_{\substack{(b,a)\in E'_p \\ a\neq s}} ly'_{b,a} \ge 0 \quad \forall a \in N \tag{11}$$

$$\sum_{\substack{(a,b)\in E'_p \\ a\neq d}} ly'_{a,b} - \sum_{\substack{(b,a)\in E'_p \\ a\neq s}} ly'_{b,a} \ge \sum_{\substack{(a,b)\in E_w \\ a\neq d}} \left(x_{a,b} - y'_{\overline{a,b}}\right) + \sum_{\substack{(b,a)\in E_w \\ a\neq s}} \left(x_{b,a} - y'_{\overline{b,a}}\right) + \sum_{\substack{(a,b)\in E'_p \\ a\neq d}} k_{\max} \cdot y'_{a,b} + \sum_{\substack{(b,a)\in E'_p \\ a\neq s}} k_{\max} \cdot y'_{b,a} - 2 \cdot k_{\max} + \sum_{\substack{(b,a)\in E_w \\ a\neq s}} {k_{\max}}^2 \cdot x_{b,a} + \sum_{\substack{(a,b)\in E_w \\ a\neq d}} {k_{\max}}^2 \cdot x_{a,b} - 2 \cdot {k_{\max}}^2 \quad \forall a \in N \tag{12}$$

$$\sum_{\substack{(a,b)\in E'_p \\ a\neq d}} ly'_{a,b} - \sum_{\substack{(b,a)\in E'_p \\ a\neq s}} ly'_{b,a} \le \sum_{\substack{(a,b)\in E_w \\ a\neq d}} (x_{a,b} - y'_{\overline{a,b}}) + \sum_{\substack{(b,a)\in E_w \\ a\neq s}} (x_{b,a} - y'_{\overline{b,a}}) \quad \forall a \in N \tag{13}$$

$$\sum_{(s,b)\in E'_p} ly'_{a,b} = 1 \quad \forall b \in N \tag{14}$$

The constraint in Eq. (10) ensures that $ly'_{a,b}$ is nonzero only if q passes through arc (a,b). The constraint in Eq. (11), (12) and (13) ensures that the value of $ly'_{a,b}$ on path Q increases by 1 only if Q merges back to W or Q switches out of W at node a. The idea behind Eq. (11), (12) and Eq. (13) is similar to that of Eq. (7) and Eq. (8). The only difference is that instead of the

forward arcs of Q (denoted as $y'_{\overrightarrow{a,b}}$), the term ($x_{a,b} - y'_{\overrightarrow{a,b}}$) is used, which is non-zero for link (a,b) along W not taken by Q. The constraint in Eq. (14) is to set $ly'_{a,b}$ as 1 if node a is the source node, and that there would be only a single protection link stretching out of the source node. Please refer to Fig. 8 for an explicit illustration for the variables formulated above. It can be easily observed that the maximum of $lx_{a,b}$ is $2k_{max} - 1$; and the maximum of $ly'_{a,b}$ is less than $(2k_{\max}-1)\cdot y'_{a,b}$ even if Q have loops.

With lx and ly' link labels, path W is divided into segments such that each link along it is covered by at least one protection segment. This effort introduces $k_{\max}\cdot|E_w|$ and $k_{\max}\cdot|E_p|$ link-domain incidence binary variables denoted as $\delta x^k_{a,b}$ and $\delta y^k_{a,b}$, which is 1 if link (a,b) is traversed by the working and protection segment of the k-th protection domain, respectively. Note that the only variable defined in graph E_p is the variable $\delta y^k_{a,b}$ in the formulation. We can alternatively define the variable $\delta y^k_{a,b}$ upon E'_p instead of having a new graph E_p, in which the formulation turns out to take only two residual graphs. Although it is a way more tractable to implement, there would be at most $k_{\max}\cdot|E_w|$ variables unnecessarily induced due to the fact that we do not need to define $\delta y^k_{a,b}$ on the reversed links of E'_p.

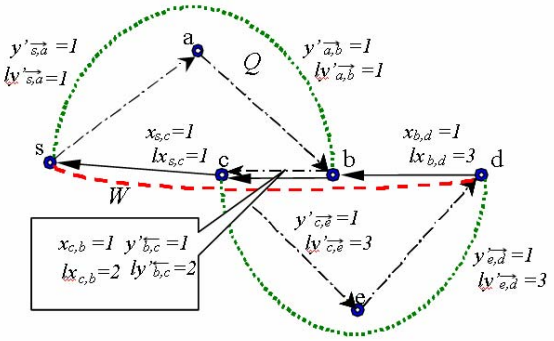


Fig. 8. An example showing the variables $x_{a,b}$, $y'_{a,b}$, $lx_{a,b}$, and $ly'_{a,b}$. It is zero for the parameters of a link not shown on the figure.

For $\delta y^k_{a,b}$, the following constraints are introduced:

$$\sum_{k=1}^{k_{\max}}(2k-1)\cdot\delta y^k_{a,b} = ly'_{\overrightarrow{a,b}} \quad \forall\,(a,b)\in E_p \tag{15}$$

$$\sum_{k=1}^{k_{\max}}\delta y^k_{a,b} = y'_{\overrightarrow{a,b}} \quad \forall\,(a,b)\in E_p \tag{16}$$

$$0\le\delta y^k_{a,b} \quad \forall\,k=1\sim k_{\max} \quad \forall(a,b)\in E_p \tag{17}$$

$$-\sum_{(a,b)\in E_w}\left(lx_{a,b}+x_{a,b}\right)\le\sum_{(a,b)\in E_p}(2k_{\max}-1)\cdot\delta y^k_{a,b}-\sum_{(b,a)\in E_p}(2k_{\max}-1)\cdot\delta y^k_{b,a} \tag{18}$$

$$\forall a\in N$$

$$\sum_{(a,b)\in E_p}(2k_{\max}-1)\cdot\delta y^k_{a,b}-\sum_{(b,a)\in E_p}(2k_{\max}-1)\cdot\delta y^k_{b,a}\le\sum_{(b,a)\in E_w}\left(lx_{b,a}+x_{b,a}\right) \tag{19}$$

$$\forall a\in N$$

Eq. (15) can be easily verified by observing Fig. 8, where the value of $ly'_{a,b}$ on Q of the first protection domain is 1; and in the second protection domain $ly'_{a,b}$ is 3; and in the k-th protection domain $ly'_{a,b}$ is $2k-1$. Eq. (15) ensures that the number of traversals of path Q upon each link is correctly counted. It is clear that Eq. (15), Eq. (16) and Eq. (17) set $\delta y^k_{a,b}=1$ only when $ly'_{a,b}=2k-1$. Eq. (18) and Eq. (19) are flow conservation constraints for $\delta y^k_{a,b}$ since $\sum_{(b,a)\in E_w}lx_{b,a}$ is 0 for all nodes accept for the ones along W. This ensures, that $\delta y^k_{a,b}$ to be a flows starting from a node along W with label $lx=2k-1-1$ (an incomming arc has the label) and terminate at a node along W with label $lx=2k-1+1$ (an outgoing arc has the label).

For $\delta x^k_{a,b}$, the following constraints are introduced:

$$-1\le\sum_{k=1}^{k_{\max}}(2\cdot k-1)\cdot\delta x^k_{a,b}-lx_{a,b}-ly'_{\overrightarrow{a,b}}\le 1 \quad \forall\,(a,b)\in E_w \tag{20}$$

$$\sum_{k=1}^{k_{\max}}\delta x^k_{a,b} = x_{a,b}+y'_{\overrightarrow{a,b}} \quad \forall\,(a,b)\in E_w \tag{21}$$

$$(2\cdot k-1)\cdot\delta x^k_{a,b}\le lx_{a,b}+1 \quad \forall\,k=1\sim k_{\max} \quad \forall\,(a,b)\in E_w \tag{22}$$

$$0\le\delta x^k_{a,b}\le 1 \quad \forall\,k=1\sim k_{\max} \quad \forall\,(a,b)\in E_w \tag{23}$$

Eq. (20) can be easily verified by the following argument. The value of $lx_{a,b}$ of link (a,b) taken by W in the first protection domain is either 1 or 2, depending on whether or not there is overlapped link(s) between the working segments of the first and the second protection domain; while on the links of the k-th protection domain, we have $2k-2=(2k-1)-1\le lx_{a,b}\le 2k=(2k-1)+1$. Since the overlapped link(s) of two neighbour working segments is counted twice in the term $\delta x^k_{a,b}$, while the corresponding reversed arcs of G'_p are taken by Q. This rule is formulated as Eq. (21). If the reversed arcs of W in G'_p is used by Q, then $lx_{a,b}=ly'_{\overrightarrow{a,b}}$; otherwise $ly'_{\overrightarrow{a,b}}=0$. With this, $\sum_k(2k-1)\cdot\delta x^k_{a,b}$ is less than $lx_{a,b}+ly'_{\overrightarrow{a,b}}+1$, and is larger than $lx_{a,b}+ly'_{\overrightarrow{a,b}}-1$ as shown in Eq. (20). To set the realtionsship of three variables ($\delta x^k_{a,b}$, $lx_{a,b}$ and $ly'_{\overrightarrow{a,b}}$) with linear equations a third linearly independent equation is needed. Thus Eq. (22) is formulated, which trivially holds and independent form Eq. (20), (21).

The constraint upon the variable $r_{u,v}$ defined in the target function is as follows:

$$\delta x^k_{a,b}+\delta y^k_{u,v}-1-\frac{sh^{a,b}_{u,v}}{b(W)}\le r_{u,v} \quad \forall\,k=1\sim k_{\max}, \tag{24}$$

$$\forall(a,b)\in E_w,\ \forall(u,v)\in E_p, sh^{a,b}_{u,v}+f_{u,v}\ge b(W)$$

$$r_{u,v}\ge 0 \quad \forall(u,v)\in E_p \tag{25}$$

Here the SRLG constraint is considered using a pre-defined

$|E_w| \times |E_p|$ matrix recording $sh^{a,b}_{u,v}$, where $(u,v) \in E_p$ and $(a,b) \in E_w$, which can be prepared off-line and behaves as an upper bound of spare capacity along link (u,v) sharable by the backup segment if the corresponding working segment passes through link (a,b). Eq. (24) ensures that when link (a,b) and (u,v) is taken by the working and protection segments in the *k-th* protection domain, respectively, the resultant amount of scaling (i.e., $r_{u,v}$) is at least $1 - sh^{u,v}_{a,b} / b(W)$ (since $\delta x^k_{a,b} + \delta y^k_{u,v} = 2$). If $sh^{a,b}_{u,v} \geq b(W)$, it means that there is sufficient sharable spare capacity along link (u,v) that can be taken to protect any additional $b(W)$ units of working capacity along link (a,b). In this case, $r_{u,v} = 0$, and the only cost imposed upon the consumption of the sharable spare capacity along link (u,v) is ε, as shown in the target function.

The following constraint imposes a bandwidth limitation upon the consumption of spare capacity.

$$\delta x^k_{a,b} + \delta y^k_{u,v} \leq 1 \quad \forall\, k = 1 \sim k_{\max}, \forall (a,b) \in E_w, \\ \forall (u,v) \in \underline{E}_p, \text{ and } sh^{a,b}_{u,v} + f_{u,v} < b(W) \tag{26}$$

Eq. (26) ensures that if $sh^{a,b}_{u,v} + f_{u,v} < b(W)$, link (a,b) and (u,v) cannot be used at the same time for a working and protection segment in the same protection domain. Note, $r_{u,v}$ is automatically transformed from E'_p to E_p such that values of $r_{u,v}$ at those reverse arcs in E'_p are set to zero.

It is clear that the adoption of the second graph has successfully defined all the three states for the protection path to take spare capacity, which are the case of $sh^{a,b}_{u,v} \geq b(W)$, the case of $sh^{a,b}_{u,v} + f_{u,v} \geq b(W) > sh^{a,b}_{u,v}$, and the case of $sh^{a,b}_{u,v} + f_{u,v} < b(W)$. The former two cases are jointly defined by Eq. (24) and Eq. (25), where $r_{u,v}$ is constrained no smaller than $1 - sh^{a,b}_{u,v} / b(W)$ and 0 in the two cases, respectively; while the latter case is defined by Eq. (26), which prohibits the traversal of any protection segment through (u,v) if there is no sufficient capacity along the link.

To sum up the above, the shared protection problem has been formulated with the same spare link-state defined in Eq. (2), in which $r_{u,v}$ is equivalent to r^W_j provided that link (u,v) is equivalent to link j. With the ILP formulation, we claim that all the three states for the protection path to take spare capacity can be well defined. Readers are encouraged to compare the resultant cost function adopted in the ILP formulation with that defined in Section II. It can also be observed that the use of the residual graphs E_w and E_p along with the constraints of Eq. (24), Eq. (25) and Eq. (26) has imposed a bandwidth limitation constraint along each link upon the selection of working and protection segment of each protection domain, respectively. Without such a design, the extra constraint on the feasibility of spare capacity resource sharing and the link bandwidth limitation for protection paths can never be defined at the same time by using a single graph.

The number of variables in an ILP formulation directly influences the computation time required to solve the formulation. In this formulation, the number of variables is $(k_{\max} + 4) \cdot |E_w| + (k_{\max} + 3) \cdot |E_p|$, and the number of rows in the constraint matrix (where the linear formulation can be expressed in a general form as $\underline{A} \cdot \underline{x} = \underline{b}$ with a target to minimize $\underline{x} \cdot \underline{c}$) is: $8 \cdot |E_w| + 9 \cdot |E_p| + 11 \cdot |N|$ plus the SRLG constraints shown in Eq. (24) and (26). Therefore, the number of rows in the matrix $\underline{A}$ has an upper bound $k_{\max} \cdot |E_w| \cdot |E_p| + 8 \cdot |E_w| + 9 \cdot |E_p| + 11 \cdot |N|$. The computation time and memory occupation for each network topology adopted in this study will be shown in Section V.

IV A HEURISTIC APPROACH – CASCADED DIVERSE ROUTING (CDR)

A novel heuristic algorithm called Cascaded Diverse Routing (CDR) is introduced in this section to perform survivable routing for segment shared protection. In general, the efforts of pre-defining a set of candidate switching/merging node-pairs and the adoption of the Iterative Two-Step-Approach (ITSA) algorithm [12] distinguish this heuristic from its counterparts.

With CDR, the switching and merging nodes of protection domain i is denoted as PSL(i) (which is the abbreviation of Path Switch LSR, where LSR is Label Switched Router in the context of Multi-Protocol Label Switching) and PML(i) (which is the abbreviation of Path Merged LSR), respectively. As an example shown in Fig. 9, the second protection domain has node E (or J) as PSL (or PML), which switches over (or merges back) the affected traffic flows originally along the working segment E-F-G-H-I as any failure occurs upon the working segment. The heuristic is characterized by the fact that the PSL-PML pairs are pre-defined to make the design space smaller. The algorithm contains the following three steps:

Step-1: Select the shortest M alternate paths from the k-shortest paths in terms of hop count for each node-pair.

Step-2: Define a series of PSL-PML pairs along each alternate path with a fixed distance D. Step (1) and (2) can be performed before the connection request arrives (or off-line). The distance D between PSL(i) and PML(i) in terms of hop count is called *diameter* of protection domain i.

Step-3: As a connection request arrives, the ITSA algorithm is invoked upon a set of PSL-PML pairs along an alternate path. This is then iteratively performed along each alternate path until a feasible solution is derived.

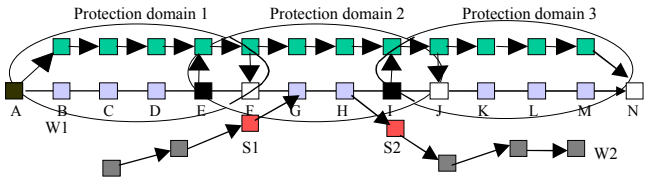


Fig. 9. An illustration of protection domains and PSL-PML pairs for W1.

In this study, the PSL-PML pairs are assigned to an alternate path according to the following pseudo-code. First, the nodes with a nodal degree larger than 2 is labelled from 1 to *N'*, where *N'* is the number of nodes with a nodal degree larger than 2 along the alternate path.

```
PSL(0) = PML(0) := 0;
overlap := 0; /* a flag which is 1 if the working segment of the
current protection domain overlaps with that of its immediate
upstream protection domain, and 0 otherwise; */
PSL(1) := 1; /* the source node is the PSL of the first protection
domain; */
i := 1; /* index for protection domains; */
For n := 2 to N'
  If (n == PML(i – 1) – overlap + D),
     PML(i) := n; i++;
     If (n = N') Break; /* the whole job is complete; */ End If;
       Else If (ND(n) >= ND(n – 1))
              PSL(i) := n; overlap := 0;
       /* ND(n) returns the nodal degree of node n, and both PSL(i)
       and PML(i-1) are at node n; */
       Else
            PSL(i) := n-1; overlap := 1;
       Else If (PML(i-1) – overlap +D > N')
     /* the diameter of the last protection domain is smaller
     than D; */
          PML(i) := N'; Break;
  End If;
End For;
```

In the above PSL-PML assignment algorithm, the working segments of two neighbour protection domains can overlap with each other either by a single node (the case with *overlap* = 0) or by two nodes and a link (the case with *overlap* = 1). The node with a higher nodal degree will be prioritized to serve as a PSL. This algorithm is simple but can efficiently determine the PSL-PML pairs framed by an alternate path.

In CDR, the ITSA algorithm is adopted to find the working and backup segments for each PSL-PML pair each alternate path, in which the cost function defined in Eq. (1) and Eq. (2) are adopted. It is clear that each alternate path is to "frame" the assignment of a set of PSL-PML pairs, which yields a fact that the working segment in protection domain *i* does not necessarily coincide with the segment between PSL(*i*) and PML(*i*) of the alternate path.

CDR yields merits in terms of network practical implementation. Due to the independency of the computation effort possibly held in the PSL of each protection domain, CDR incorporate with a distributed-control environment in such a way that each pre-assigned PSL instead of the source node of the connection takes the responsibility of calculating the working and protection segments in the corresponding protection domain. This design can release workload of the edge routers by possibly addressing intelligence upon the glass nodes in the networks. In addition, the segmentation of the whole diverse routing process can help reducing the total computation complexity. Since each PSL-PML pair along an alternate path is pre-defined, some of the computation tasks can go through a look-up-table process, such as the calculation of the **SH** matrix defined in Section II.C. Due to the fixed and relatively short distance between a PSL-PML pair, the number of iterations allowed for the ITSA algorithm can be much smaller than the case where an end-to-end working and protection path-pair is calculated, which further reduces the computation complexity.

V PERFORMANCE EVALUATION

We conduct experiments to verify the proposed ILP model and the CDR algorithm on the networks topologies as listed in Table I. The experiment is divided into two parts in terms of their objectives. The first part investigates the diameter for CDR that can achieve the best performance in terms of blocking probability in each network topology. The second is aimed at comparing the CDR algorithm with the other reported schemes. In the second part, we will also examine the optimality that is achievable by each scheme relative to the results derived from solving the ILP formulation.

Table I

Topology of the networks adopted in the simulation. All the topologies are two-connected mesh networks.

Network	N0	N1	N2	N3	N4
No. of nodes	10	22	30	79	100
No. of links	28	88	126	216	358
Nodal degree	2.8	4.0	4.2	2.73	3.58

The experiment is arranged as follows. Each directional link in the networks contains 32 units of bandwidth. We consider the capacity efficiency in terms of blocking probability for the dynamically arrived connection requests (of a single bandwidth unit) following the Poisson model and with a holding time defined in an exponential distribution function. In the experiment arrangement, node-pair (*i,j*) has a traffic load $\rho_{i,j} = \eta \cdot (\lambda_{i,j} / \mu_{i,j})$, where $\lambda_{i,j}, \mu_{i,j}$ are arrival and departure rate upon the node-pair (*i,j*), respectively. Without loss of generality, $\mu_{i,j}$ is set to 1, while $\lambda_{i,j}$ is a random number between 0.5~1.5 for every node-pair (*i,j*) such that each node-pair may be subject to different amount of path connection setup demand. The scaling parameter η represents the level of traffic load in the network with the unit of *Erlang*. Each data point in the figures is the blocking probability for 100,000 connection requests using a specific survivable routing algorithm. The confidence interval is within 0.1% if we take the result of 100 connection requests as a trial.

A Derivation of the Best Diameter for CDR

With CDR, since the diameter of protection domains may

influence the total throughput, we first derive the best diameter for achieving the best performance in each network topology through simulation. Fig. 11 shows the simulation results, where "CDR(D)" is the CDR scheme taking the diameter of each protection domain as D; and the "Path" is the scheme with the number of protection domains as 1 for each connection. With CDR, the ITSA algorithm is invoked for solving the working and protection segment pair in the corresponding protection domain. In this simulation, the number of alternate paths pre-defined for each node-pair is 10 (i.e., M = 10), and the number of iterations allowed for the ITSA algorithm is 10, 10, 20, and 25 for the network N1, N2, N3, and N4. To achieve better computation efficiency, the algorithm returns either when 6 iterations are performed or when the least-cost working-protection segment-pair is derived. If the algorithm fails to find a feasible working and protection segment pair within 6 iterations in any protection domain along a pre-defined alternate path, it will drop the results/calculation processes of all the other protection domains framed by the alternate path, and proceed to inspect the next alternate path. If the algorithm fails to find a solution after inspecting all the alternate paths, a blocking is announced for the connection request. The parameter c_j in Eq. (1) is defined as: $c_j = 1/rc_j$, where rc_j is the residual bandwidth along link j.

The experiment results are shown in Fig. 10. It is clear that the path shared protection scheme is outperformed by CDR with any size of protection domain. With the small 22-node and 30-node networks in which the average distance between each S-D pair is 2.49 and 2.71, respectively, the best performance in capacity efficiency is achieved in the case CDR(2). With the 79-node and 100-node networks in which the average distance between each S-D pair is 6.57 and 9.5, respectively, the best capacity efficiency is achieved in the case CDR(3) and CDR(4), respectively. It is observed that the best diameter of protection domains varies as the network size is different.

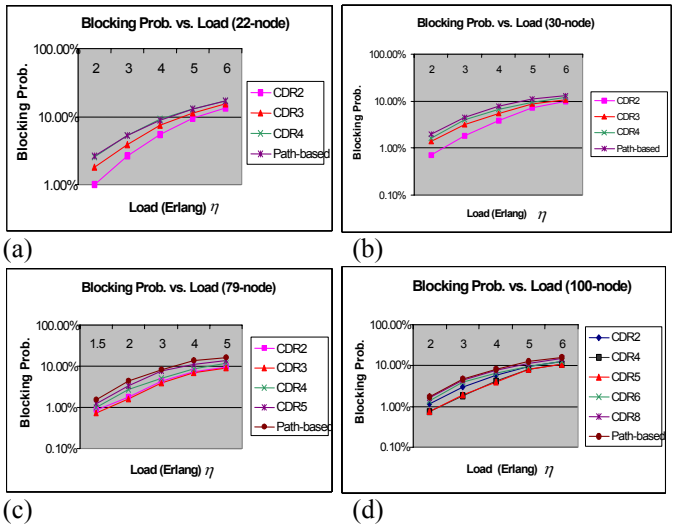


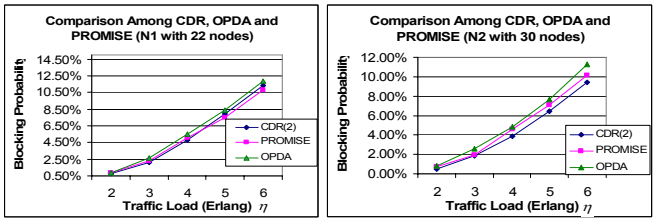
Fig. 10. Simulation for CDR with different traffic load and diameters of protection domains.

B The Experiment for Comparison

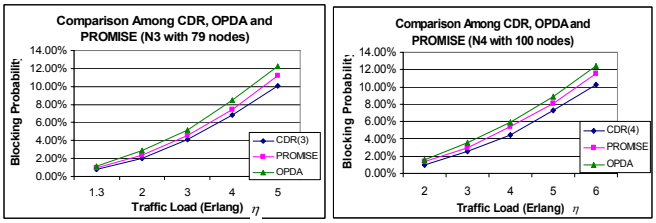
In addition to the CDR algorithm with the best diameter in each network topology (i.e., 2, 2, 3, and 4 hops for the network N1, N2, N3 and N4, respectively), we also adopt the segment shared protection algorithms in [6] and [7] for making a comparison. Note that in this study we focus on capacity efficiency of each algorithm, and will not pay attention to the resultant restoration time.

The implementation of the other two schemes is briefly described as follows. The first scheme taken for comparison in this simulation is the dynamic-programming-based solution provided in [7], which is denoted as PROMISE. With PROMISE, a limited number of combinations of segmentation are inspected along a single working path, which is derived in advance by invoking Dijkstra's shortest path first algorithm with the cost function shown in Eq. (1) and Eq. (3). The second algorithm is provided in [9] (denoted as OPDA), where the length limitation for each candidate cycle is 10 hops. Since OPDA is originally designed for optical networks with partial wavelength conversion capability in each node, we simplify the algorithm by taking the whole networks as having a single wavelength plane.

Fig. 11 shows the simulation results of the comparison made among the three cases under different network topologies and traffic load. It is clear that CDR outperforms the other two while OPDA is the worst. An observation is made upon CDR and PROMISE described as follows. The two schemes place the scarce computation resources on different searching dimensions for allocating the protection domains. The CDR algorithm pre-defines several sets of PSL-PML pairs with fixed distance (back-tract of one hop is allowed) for each S-D pair, and tries to find a better solution by manipulating the location of working path segments. The PROMISE scheme, on the other hand, tries to maximize the spare capacity resource sharing by finding better locations of the switching and merging node (i.e., PSL-PML pairs) for each protection domain along a working path given in advance. With different approximate terms, the two approaches yield different blocking performance, optimality (see Table III), and computation time (see Fig. 12), in which a compromise upon different design dimensions is initiated in each scheme.



(a) (b)



(c) (d)

Fig. 11. Comparison among the three schemes: CDR, OPDA and PROMISE using the topologies N1, N2, N3, and N4.

As for OPDA, it is seen outperformed by both of the other two schemes. Since OPDA is designed for WDM networks with partial wavelength conversion capability in each node, the migration of OPDA to this study may invalidate the merits of the algorithm that are originally possessed. For example, OPDA provides a strong functionality to deal with the non-linearity inherent in the task of routing and wavelength assignment for partial wavelength convertible optical networks, which, however, is not appreciated in this case.

Fig. 12 shows the average computation time for allocating a connection request in each case, which shows that the CDR scheme takes a little bit more computation time than PROMISE with the current experiment settings, while OPDA yields the least average computation time in the simulation. Note that this computation time is achieved in each case by creating and maintaining a $|E|\times|E|$ matrix (called spare provision matrix [15]) while the simulation is running, which records the dependency between spare and working capacity along each pair of links in the network.

We solve the ILP formulation in this paper using CPLEX 7.5 and a Sun Ultra 80 workstation. The average computation time (including time taken by the pre-solver of CPLEX), memory occupation, the number of rows, columns and the number of non-zero elements of the constraint matrix (after the CPLEX pre-solver reducing the size of the problem) are shown for both cases in Table II.

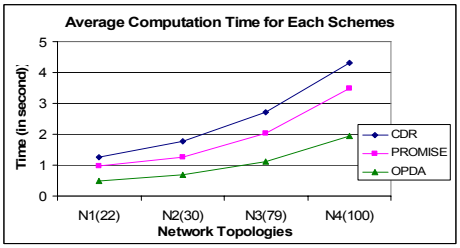


Fig. 12. Comparison of average computation time for each case in the network topologies N1, N2, N3, and N4.

It is clear that the complexity of solving the ILP grows very fast as the network size increases, which leads to a fact that the approach can hardly be applied for any on-line purpose. Therefore, it is positioned as a benchmark to evaluate the optimality achievable by the other heuristic counterparts. Table III is a demonstration on the optimality achieved by each scheme in the simulation. The offset from the optimality is defined as $Q = C_a/C_{opt} - 1$, where C_a is the total cost of the working and protection segments for the connection request by using a specific scheme, while C_{opt} is the cost achieved by solving the ILP formulation. The index Q is called *Offset of Optimality*; in this particular case, Q is nothing but an evaluation on the extent of resource sharing plus the extra (or unnecessary) cost taken by the working path. The smaller the value of Q is the better optimality has the heuristic algorithm achieved. Due to the very lengthy computation process in the 79- and 100-node networks, we only conduct this experiment in the 10-, 22- and 30-node networks, in which the ILP formulation is solved for every one of 1000 connection requests while the simulation is running in each case. Note that since the optimality is focused in this experiment, we do not consider any connection request that is blocked. Table III shows the average Q value and the average cost taken by each connection for in the experiment using N1 and N2.

Table II

The computation time and the amount of memory occupied for CPLEX solving the ILP formulation.

	Time (sec)	Memory (MB)	Row	Columns	Non-zero
N0 (10)	5.6	5	1772	281	6012
N1 (22)	255	23	16996	993	53534
N2 (30)	1185	42	31840	1293	99012

Table III

Offset of optimality (Q value) and the average cost taken by each connection request for all the schemes.

	CDR	**PROMISE**	**OPDA**	**ILP**
N1	9.3%	10.1%	13.8%	0%
(22-node)	2.210	2.226	2.301	2.022
N2	12.1%	13.9%	16.4%	0%
(30-node)	2.439	2.478	2.533	2.176

V CONCLUSIONS

In this paper we have studied dynamic survivable routing for a special type of protection, called segment shared protection, in which a novel Integer Linear Programming (ILP) formulation and a heuristic algorithm, called Cascaded Diverse Routing (CDR), are proposed. This paper first defines segment shared protection, and qualitatively demonstrates the advantages in using segment shared protection, which includes the facts that the restoration time can be shortened/guaranteed and that a higher possibility of resource sharing can happen between different protection segments. We also define the spare link-state taking the SRLG constraint into consideration. The ILP formulation is thus presented, where the switching/merging nodes and the corresponding least-cost

working and protection segment pair for a connection request are jointly determined in the programming process. A novel approach of arc reversal along with a graph transformation method is devised to keep the formulation linear and to deal with the situation that the working segments of two neighbouring protection domains may overlap with each other by more than a single node. Due to the very high computation complexity in solving the ILP, the heuristic algorithm, CDR, is introduced. CDR is characterized by the fact that each working and protection segment pair in a protection domain can be solved independently from solving that of the other protection domains for a connection request, which is designed to achieve a better performance/computation-complexity gain. To verify the proposed algorithms, a series of experiments/simulation are conducted. By the simulation results, we first find that CDR with a well designed diameter for each protection domain in a specific network topology can yield the best performance. With the best diameter in each network topology, CDR is compared with two reported counterparts, namely PROMISE and OPDA. The simulation results show that CDR can achieve the best performance at the expense of longer computation time while OPDA yields the worst. We also examine the optimality achievable in each algorithm referring to the results of solving the ILP formulation, and find that the better performance of CDR is due to the smaller offset of optimality, which is in turn caused by the flexibility of selecting a working segment in each protection domain.

REFERENCES

[1] D. Zhou and S. Subramaniam, "Survivability in Optical Networks", *IEEE Networks,* November/December 2000, pp. 16-23.

[2] P. –H. Ho and H. T. Mouftah, "A Framework of Service Guaranteed Shared Protection for Optical Networks, "*IEEE Communications Magazine*, Feb. 2002, pp. 97-103.

[3] C. V. Saradhi and C. Siva Ram Murthy, "Dynamic Establishment of Segmented Protection Paths in Single and Multi-fiber WDM Mesh Networks", *Proceedings SPIE OPTICOMM* Aug. 2002, Boston, MA, pp. 211-222.

[4] M. Kodialam and T. V. Lakeshman, "Dynamic Routing of Locally Restorable Bandwidth Guaranteed Tunnels Using Aggregated Link Usage Information", *Proceedings IEEE INFOCOM 2001*, Anchorage, Alaska, pp. 376-385.

[5] C. –F. Su and X. Su, "An On-line Distributed Protection Algorithm in WDM Networks", *in ICC 2001.*

[6] Y. Bejerano, Y. Breitbart, A. Orda, R. Rastogi, and A. Srintson, "Algorithms for Computing QoS Paths with Restoration", *IEEE INFOCOM,* San Francisco, April 2003.

[7] D. Xu, Y. Xiong, and C. Qiao, "Protection with Multi-Segments (PROMISE) in Networks with Shared Risk Link Groups (SRLG)", *in the 40th Annual Allerton Conference on Communication, Control, and Computing*, 2002.

[8] L. Li, M. Buddhikot, C. Chekuri, and K. Guo, "Routing Bandwidth Guaranteed Paths with Local Restoration in Label Switched Networks", *The 10th IEEE International Conference on Network Protocols (ICNP)*, Nov. 2002, Paris, France.

[9] P. -H. Ho and H. T. Mouftah, Spare Capacity Allocation for WDM Mesh Networks with Partial Wavelength Conversion Capacity", *IEEE High Performance Switching and Routing, HPSR 2003*.

[10] D. Xu, C. Chunming, and Y. Xiong, "An Ultra-Fast Shared Path Protection Scheme – Distributed Partial Information Management, Part II*", The 10th IEEE International Conference on Network Protocols (ICNP)*, Nov. 2002, Paris, France.

[11] Y. Xiong, D. Xu and C. Qiao, "Achieving Fast and Bandwidth Efficient Shared-path Protection", *IEEE Journal of Lightwave Technology,* Feb. 2003.

[12] P. –H. Ho and H. T. Mouftah, "On Optimal Diverse Routing for Shared Protection in Mesh WDM Networks", *IEEE Transaction on Reliability*. (*to appear in March 2004*)

[13] J. Tapolcai and T. Cinkler, "On-line Routing Algorithm with Shared Protection in WDM Networks", *ONDM,* Budapest, Hungary, Feb. 2003.

[14] G. Li, D. Wang, C. Kalmanek, and R. Doverspike, "Efficient Distributed Path Selection for Shared Restoration Connections", *IEEE INFOCOM 2002*, NY, NY, June 2002.

[15] Y. Liu, D. Tipper, and P. Siripongwutikorn, "Approximating Optimal Spare Capacity Allocation by Successive Survivable Routing", *Proceedings IEEE INFOCOM 2001*, vol. 2, pp. 699-708, April 2001.

[16] R. Ramamurthy, S. Sengupta, S. Chaudhuri, "Comparison of Centralized and Distributed Provisioning of Lightpaths in Mesh Optical networks", *OFC 2001*, Anaheim, CA, March 2001.

[17] E. Bouillet, J. –F. Labourdette, G. Ellina, R. Ramamurthy, and S. Chaudhuri, "Stochastic Approaches to Compute Shared Mesh Restored Lightpaths in Optical Network Architectures", *IEEE INFOCOM 2002*, NY, NY, June 2002.

[18] C. Qiao and D. Xu, "Distributed Partial Information Management (DPIM) Schemes for Survivable Networks - Part I", *IEEE INFOCOM 2002*, NY, NY, June 2002.

[19] J. W. Surrballe and R. E. Tarjan, "A Quick Method for Finding Shortest Pairs of Disjoint Paths," *Networks*, 14(2):325-336, 1984.

[20] D. Papadimitriou, F. Poppe, S. Dharanikota, R. Hartani, R. Jain, J. Jones, S. Venkatachalam, and Y. Xue, "Shared Risk Link Groups Inference and Processing", *Internet Draft, <draft-papadimitriou-ccamp-srlg-processing-02.txt>, work in progress,* June 2003.

[21] S. Yuan and J. P. Jue, "A Heuristic Routing Algorithm for Shared Protection in Connection- Oriented Networks," *Proceedings SPIE OPTCOMM 2001*, Denver, CO, vol. 4599, pp. 142-152, August 2001.