On Achieving Optimal Survivable Routing for Shared Protection in Survivable Next-Generation Internet

Pin-Han Ho, Member IEEE
Department of Electrical and Computer Engineering, University of Waterloo, Ontario, Canada
pinhan@bbcr.uwaterloo.ca
TEL: (519) 888-4567-2452
Address: 200 University Ave. West, Waterloo, Ontario, Canada N2T 3G1

Janos Tapolcai, Member IEEE
Department of Computer Science and Information Theory, Budapest University of Technology and Economics,
Magyar tudósok Kőrútja 2., Hungary, 1117, janos@cs.bme.hu
TEL: (36) 1-4634391

Hussein T. Mouftah, Fellow IEEE
School of Information Technology and Engineering, University of Ottawa, Ontario, Canada
mouftah@site.uottawa.ca
TEL: (613) 562-5800-2173
Address: 800 King Edward Avenue, Ottawa, Ontario, Canada, K1N 6N5

Abstract –This paper proposes a suite of approaches to solving the survivable routing problem with shared protection. We first define in mathematics the maximum extent of resource sharing for a protection path given the corresponding working path according to the current network link-state. Then the problem of solving the least-cost working and protection path-pair (in terms of the sum of the cost) is formulated into an Integer Linear Programming process. Due to the dependency of the protection path on its working path, however, the formulation is nonetheless not scalable with the network size which takes an extra effort to solve. Therefore, two heuristic algorithms are introduced, called Iterative Two-Step-Approach (ITSA) and Maximum Likelihood Relaxation (MLR), which aim to explore the approximating optimal solutions with less computation time. We evaluate the performance of the proposed schemes and make a comparison with some reported counterparts. The simulation results show that the ITSA scheme with a properly defined tolerance to the optimality can achieve the best performance at the expense of taking longer computation time. On the other hand, MLR yield a faster path selection process, which initiates a compromise between computation efficiency and performance.

Keywords: WDM, Diverse Routing, Shared Protection, Lightpath, Single failure scenario, Iterative Two-Step-Approach (ITSA), Integer Programming, Shared Risk Link Group (SRLG).

I. INTRODUCTION

Survivability has emerged as the most important issue in the design of the control and management plane for the next-generation networks. To deal with any unexpected interruption caused by accident events (such as rodent bite on communication fibers), pre-planning a protection path with sufficient bandwidth for each working path has been widely accepted as the most effective solution. The strategy is also known as *survivable routing* in the path selection stage. Innumerable research has been

conducted since mid 1990's to equip the Internet and Metro-area networks with reliability and resilience to any unexpected interruption [1,2,4-15,17]. For the networks with connection requests arriving one after the other without any knowledge of future arrivals, it is important to develop a suite of interoperable strategies that can real-time solve for a link-disjoint working and protection path-pair upon the current link-state with high capacity-efficiency.

With survivable routing, working and protection path-pairs are link- or node-disjoint, in which two types of protection are defined – dedicated and shared protection. The difference between the two lies in whether or not spare capacity resource sharing is allowed between different protection paths. In dedicated protection, each working and protection path-pair is pre-configured, and is launched with the same copy of data between a source-destination (S-D) pair at the same time during a normal operation. Although dedicated protection (e.g., 1+1) provides a very fast restoration service, the ratio of redundancy (i.e., the ratio of capacity taken by protection and working paths in the network) usually reaches 100%. On the other hand, in the shared protection only working paths are launched with data flows. The spare capacity reserved by protection paths is *pre-planned* without the necessity of being pre-configured, and may be shared with the other protection paths in the shared protection mode. It has been observed that the spare capacity resource sharing between different protection paths can substantially reduce the ratio of redundancy required to achieve 100% restorability at the expense of a little longer restoration time [7,8].

For the survivable routing problem, Suurballe's algorithm, reported in early 70's, is famous for its polynomial computation complexity in solving optimal disjoint path-pairs in terms of the cost sum of the two paths on a directed graph [6,16]. It is notable that Suurballe's algorithm uses the same suite of link-state to derive the two paths, and can only be useful with dedicated protection. For shared protection, the constraint of spare capacity sharing must be investigated upon each network link before the best protection path can be derived for a working path. Whether or not a link has *sharable spare capacity* for a protection path depends on the physical location of the corresponding working path. This is also known as the *dependency* of the protection path on its working path.

The dependency has imposed different design criteria such that the existing linear formulations and most of the reported schemes cannot address the problem. In order to explore the dependency, the most straightforward way is to use Two-Step-Approach [6] to derive the link- or node-disjoint path-pairs [2,5,7,9,10,11,12,13,14,17]. However, the Two-Step-Approach is neither general to different network topologies nor systematic in deriving the best working and protection path-pairs upon the current link-state. In [5], a method is provided to find an optimal protection path given a working path for shared protection, called Pool-Reserved Backup Sharing. However, the study did touch the problem how to select the working paths. To address the allocation of working paths, the studies in [2,9,10,11] inspect k-shortest paths between each S-D pair one after the other until the least-cost working and shared protection path-pair is derived. In [2], a similar method to that of [5] is adopted along with the inspection of the k-shortest paths. The study in [9] creates special data structure to fast derive the k-shortest paths for solving the optimal diverse routing problem. However, it fails to specifically define

the sharing constraint to achieve 100% restorability. The study in [10] proposes an approach that further assigns cost to those links with sharable spare capacity, in which the probability for a link to be used by some other affected working paths due to failure is considered. However, this method does not guarantee the availability for the spare capacity along the derived protection paths at the occurrence of failure, and is not effective in some cases where the restorability of traffic flows is strictly required. In [11], a novel link cost metric for solving working paths is provided. The cost for the working path to take link j is determined by the maximum spare capacity among all the other links in the network which protects link j. The proposed link metric can encourage the working path to take links yielding smaller maximum non-sharable spare capacity along some other links for the protection path, and is reported to have a better performance behavior (in terms of blocking probability) than simply using hop count as the cost function. All the above schemes take the approach of enumerating the k-shortest paths n each S-D pair, which wears out the novelty and leaves a large space to improve.

In addition to the end-to-end protection, the studies in [7,12,13,14,17] suggest to segment each working path and find a protection path segment for each working path segment. In [12], a heuristic algorithm is developed, which, however, did not consider the backup bandwidth sharing until the physical routes of the backup segments are defined. In [13] and [17], two related dynamic algorithms are proposed to switchover for each link from its immediate upstream node and merge back to the original path at the immediate downstream node and any of the downstream nodes, respectively. However, both of the studies do not impose any limitation on the length of the backup paths, and may impair the overall performance. It is notable that all the above schemes are Active-Path-First-based [11] (abbreviated as APF, which means that the working path is derived first), in which little attempt has been made to jointly solve the working and shared protection path-pair. In [7,14], a framework called Short Leap Shared Protection (SLSP) along with a dynamic algorithm called CDR (Cascaded Diverse Routing) is proposed to perform segmented shared protection, in which the enumeration of k-shortest paths in each segment of the working path is performed. The performance is reported to outperform its path-based and path-based shared protection counterparts while at the expense of much larger computation complexity. In addition, the link-based or segment-based shared protection takes extra signaling efforts and is imposed of high requirements on the hardware responsiveness.

This paper turns back to the path shared protection aiming at achieving high computation efficiency without losing much performance. In this paper, we first define the optimality for a working and protection path-pair in shared protection. Then the optimal survivable routing problem is formulated into an Integer Linear Programming process, in which the optimal working and protection path-pair is jointly solved. To our best survey, this is the first study that formulates the optimal shared diverse routing problem into an Integer Linear Programming process. To avoid the high computation complexity caused by the NP-completeness in solving the ILP formulation, we develop two heuristic schemes to explore the best efficiency in solving the problem. The first is called Iterative Two-Step-Approach (ITSA), which enumerates and inspects k-shortest paths as the working path, and is also an extension of the schemes in [2], [9] and [10]. With ITSA, the computation complexity for on-line searching the k-shortest paths becomes a bottleneck for the overall performance. Therefore, the

second scheme, called Maximum Likelihood Relaxation (MLR), is a modified Dijkstra's algorithm which yields polynomial time complexity. The MLR scheme considers the allocation of the protection path while the working path is being calculated, which aims to select a working path in such a way that the number of links containing enough sharable spare capacity weighted by the reciprocal of the link cost is maximized. We evaluate the performance of the proposed schemes by making a comparison with their counterparts using four network topologies with dynamically arrived connection requests.

This paper is organized as follows. In Section II, a formal definition is given to the Two-Step-Approach for the shared protection purpose, including a short introduction to the sharing constraint unique to shared protection. In Section III, the optimal diverse routing problem is formulated into an Integer Linear Programming process followed by the introduction of the two heuristic approaches: ITSA and MLR. Section IV evaluates the performance of the proposed schemes and makes a comparison with some other reported schemes. Section V concludes.

II. TWO-STEP-APPROACH IN SHARED PROTECTION

In this section a specific cost function and dynamic link metric are defined to facilitate solving the best protection path for a given working path by using the Two-Step-Approach. The following tables show a list of notations and acronym in this study, respectively.

Table I. List of symbols.

cw_j	Cost of taking link j by working path W
cp_j^W	Spare link-state, or the cost of taking link j for spare capacity by the protection path of W
LW	The set of links taken by W
WD	A set of working paths taken by any working path other than W
$B(W)$	The bandwidth of W
V_j	The amount of spare capacity along link j
S_j^W	The amount of non-sharable spare capacity in terms of the protection path of W
sh_j^W	The amount of sharable spare capacity in terms of the protection path of W; $sh_j^W = V_j - S_j^W$
SB_l^j	The amount of non-sharable spare capacity on link j due to the working capacity on link l

Table II. List of acronym.

SRLG	Shared Risk Link Group
ITSA	Iterative Two-Step-Approach
MLR	Maximum Likelihood Relaxation
APF	Active Path First
CDR	Cascaded Diverse Routing
ILP	Integer Linear Programming

In shared protection, the feasibility of resource sharing between different protection paths is

determined by the relationship of their working paths. With the *single failure scenario* assumption in this study, we take working paths traversing through the same fiber conduit to be in a Shared Risk Link Group (SRLG) [7] subject to the same risk of single failure. All protection paths for the working paths in an SRLG are subject to a spare capacity resource sharing constraint, or called the *SRLG constraint* defined in [7]. With the SRLG constraint, protection paths cannot share spare capacity if their working paths are in an SRLG. As an example shown in Fig. 1, a working and protection path-pair W1-P1 connects nodes S and D. If another working path W2 is allocated such that a common physical link (i.e., link A-B) with W1 is taken, we call that W1 and W2 are in an SRLG. The SRLG constraint stipulates that the protection path of W2 has to reserve extra spare capacity if it attempts to traverse any common link with P1.

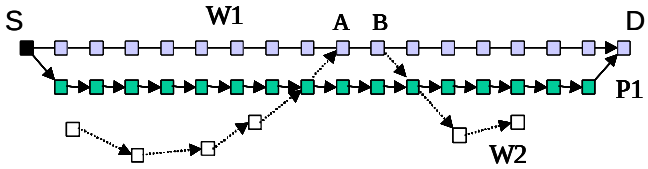


Fig. 1. An example showing the SRLG and the SRLG constraint.

The dependency between the working and protection paths caused by the SRLG constraint is the reason why Suurballe's algorithm cannot be applied in this case. Therefore, we can only use the Two-Step-Approach to solve the two paths one after the other, where the working path is solved first, followed by its protection path on the residual network topology with the links traversed by the working path excluded. This residual network topology along with the associated link-state is also called *spare link-state*, which is specific to the working path, and can only be derived by correlating all the other working paths on the same SRLG with the working path.

The cost function and link-state metric for finding working path *W* are as follows:

$$cw_j = \begin{cases} \infty & \text{if the channel } j \text{ is not reservable} \\ c_j & \text{otherwise} \end{cases} \quad (1)$$

For the reservation of a protection path of *W* we need to define the corresponding spare link-state, which is:

$$cp_j^W = \begin{cases} \infty & \text{if link } j \text{ is not reservable} \\ r_j \cdot c_j & \text{if resource sharing is allowed or partly allowed along } j \end{cases} \quad (2)$$

where r_j is a scaling parameter with a value between 0 and 1, and will be defined later in Eq. (3). It is notable that this study is different from most of the other studies in that the cost for a protection path to take sharable spare capacity along link j is proportional to the link cost for working paths instead of assigning zero or a very small value. The reason for doing so is as follows. Although a protection path traversing a link with sharable spare capacity will not consume any network resource along the link, *the registration of the spare capacity may reduce the chance for the spare capacity to be taken by the other protection paths arriving in the future*. The link cost c_j is custom-designed, and can either be a constant (e.g., simply 1 for each hop), or can take dynamic network traffic into consideration (e.g., the maximum

reservable bandwidth along link *j*). The fact that a channel is not reservable by a working path can be because the link is too full of bandwidth to accommodate a single lightpath, or because the link is under some administrative constraints. However, for a protection path, a link cost can be zero because the link has sufficient ***sharable*** spare capacity; or infinity because it has insufficient free capacity or sharable spare capacity; or c_j if the protection path has to reserve free capacity, which results in the same amount of resource consumption as the working path.

To figure out the spare link-state cp_j^W in Eq. (2), we first need to define ***sharable*** and ***non-sharable*** spare capacity along every network link. Fig. 2 illustrates the capacity distribution along link *j*. Note that due to the dependency between working and protection path-pairs, the amount of sharable spare capacity (i.e., $V_j - S_j^W$), and non-sharable spare capacity (i.e., S_j^W), cannot be defined until the corresponding working path *W* is presented.

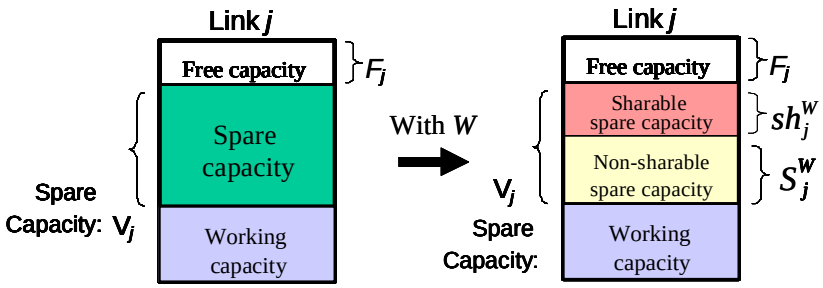


Fig. 2. An illustration of capacity distribution along link *j*.

With the categorization of capacity along link *j* shown in Fig. 2, Eq. (2) can be generalized as:

$$cp_j^W = \begin{cases} c_j \cdot [1 - sh_j^W / B(W) + sh_j^W / (V_j + sh_j^W)] & \text{if } sh_j^W + F_j \geq B(W) \\ c_j \cdot B(W) / (V_j + B(\mathrm{W})) & \text{if } sh_j^W > B(W), \\ \infty & \text{if } sh_j^W + F_j < B(W), \text{ or } j \in LW \end{cases} \quad (3)$$

for all $j \in L$, where $B(W)$ is the bandwidth of working path *W*, and *LW* is all the links traversed by *W*. Fig. 3 shows the four situations defined in Eq. (3). In Fig. 3(a) and Fig. 3(b), $B(W)$ may take partly from the free-capacity region and the sharable spare capacity region; therefore, the link cost is $c_j \cdot [1 - (V_j - S_j^W) / B(W) + (V_j - S_j^W) / (2 \cdot V_j - S_j^W)]$, which is shown in the first condition in Eq. (3). The two terms $c_j \cdot [1 - sh_j^W / B(W)]$ and $c_j \cdot [sh_j^W / (V_j + sh_j^W)]$ represent the cost for taking free and sharable spare capacity, respectively. In Fig. 3(c), all $B(W)$ can take sharable spare capacity, therefore, the cost is $c_j \cdot B(W) / (V_j + B(W))$, as shown in the second condition in Eq. (3). In Fig. 3(d), the link cost is infinity because $B(W)$ cannot be supported by the residual capacity of the link, as shown in the third condition in Eq. (3). Note that a protection path is assumed to take sharable spare capacity along a link whenever it is available. If there is not enough sharable spare capacity, the protection path

takes free capacity after considering all the sharable spare capacity.

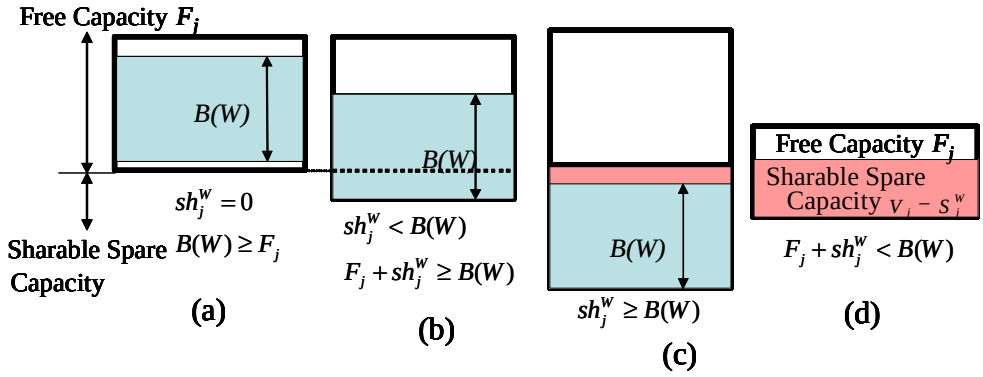


Fig. 3. The four situations defined in Eq. (3). This figure is an extension of Fig. 2.

The remaining problem is how to derive S_j^W, which is a network-wide link-state specific to the presence of *W*. Let *W* traverse through a set of links, *LW*. Every link $l \in LW$ is traversed by a set of working paths, WD_l^W, before *W* is present. A simple illustration of the network apparatus is shown in Fig. 4. In this example, *LW* is a link set traversed by *W*, which are A-B, B-C and C-D. Our objective is to determine cp_j^W – the link cost as the protection path of *W* attempts to reserve link *j*. To determine cp_j^W we must figure out the amount of spare capacity along link *j* subject to the SRLG constraint with the protection path of *W*. This amount of non-sharable spare capacity originates from the protection paths whose working paths are in the same SRLG with *W*. Therefore, the amount of non-sharable spare capacity along link *j* strongly depends on whether or not the working paths in *LW* have protection paths traversing through link *j*.

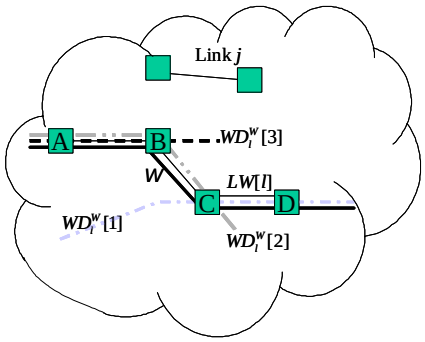


Fig. 4. An illustration of the network apparatus and symbols.

Now we are interested in the spare capacity on link *j* demanded by the working paths along a link $l \in LW$, which is denoted as SB_l^j. Note that the protection paths of the working paths in WD_l^W, where $l \in LW$, can possibly yield non-sharable spare capacity along some links in the network. Thus, SB_l^j can be expressed as:

$$SB_l^j = \sum_p B(W) \cdot d_p^j \cdot s_p^l \quad \text{for } l \in LW \text{ and all path } p \tag{4}$$

where $B(p)$ is the bandwidth of working path p, and d_p^j and s_p^l are two binary indicators defined as follows:

$$d_p^j = \begin{cases} 1 & \text{if } p \text{ has its protection path traverse link } j \\ 0 & \text{otherwise} \end{cases}$$

$$s_p^l = \begin{cases} 1 & \text{if } p \text{ passes } l \\ 0 & \text{otherwise} \end{cases}$$

In other words, the demand for spare capacity upon link j from the working capacity along link l, which is subject to the SRLG constraint with the protection path of W, is the summation of all working bandwidth on link l that has the corresponding spare capacity along link j. We need to have summation of working capacity because the working paths traversing through link l are in the same SRLG, as a result, their protection paths cannot share any spare capacity along link j.

Given a single failure scenario, only single link belonging to LW can possibly be subject to any interruption at a moment. Therefore, we can derive S_j^W by finding the maximum demand of spare capacity along all the links $\{l \in LW\}$, i.e.,

$$S_j^W = \max_{l \in LW}(SB_l^j) \tag{5}$$

With the knowledge of S_j^W, we can determine the link-state cp_j^W defined in Eq. (3) to solve the optimal protection path given the working path W.

A pseudo code for deriving the *spare link-state* corresponding to W is as follows. A network is modeled as a graph topology **G**(N, E, WD), where N, E and WD are the set of nodes, directional links, and working paths existing in the network, respectively. Let the given working path, W, traverse link set LW, where $LW \subset E$. The amount of original spare capacity along each link is V_j where $j \in E$. WD_l^W is denoted as a set of working paths traversing through link $l \in LW$. The pseudo code first derive the non-sharable spare capacity S_j^W, where $j \in E$. Without loss of generality, every connection request is for a single lightpath with the same bandwidth, and therefore both S_j^W and V_j are integer. It is clear that $S_j^W \geq V_j$, and the difference between S_j^W and V_j is the sharable spare capacity along each link as shown in Fig. 2.

Input: Network topology; original spare capacity V_j and the free capacity for each link F_j for $j \in E$;

working path to be inspected, *W*. It is defined that the bandwidth of a path *p* is denoted as *B*(*p*).

Output: Spare link-state, LS_j^W for $j \in E$.

Initiate: $S_j^W \leftarrow 0$ and $cp_j^W \leftarrow \infty$ for $j \in E$;
For ($\forall$ link $j \in E$ and $j \notin LW$) Loop (1)
For ($l \leftarrow 1$ to $|LW|$) Loop (2)
$SB_j^l \leftarrow 0$;
/ SB_l^j is the amount of spare capacity on link j used to protect working capacity in WD_l^W */*
$WD_l^W \leftarrow$ all the working paths passing through LW_l;
/ i.e., all working paths sharing the same risk of a single failure passing through LW_l */*
For ($\forall$ working path $p \in WD_l^W$ and its protection path passing link j) Loop (3)
$SB_j^l \leftarrow SB_j^l + B(p)$;
/ SB_j^l is the minimum spare capacity required along link j to protect the working capacity on link l where $l \in LW$, which is the bandwidth sum of all the working paths traversing through link l; we need to sum up because the working capacity are subject to the SRLG constraint, where no resource sharing between the protection paths of the working paths is possible; */*
End For; Loop (3)
If ($SB_j^l > S_j^W$)
$S_j^W \leftarrow SB_j^l$;
End If
/ with the single failure scenario, we need to find the maximum of SB_j^l for all $l \in LW$; */*
End For Loop (2)
/ the non-sharable spare capacity S_j^W caused by the SRLG constraint is determined; */*
/ after the derivation of S_j^W , the following pseudo code determines the spare link-state of link l, which is an implementation of* Eq. (3)*; */*

If ($sh_j^W + F_j \geq B(W)$)
/ enough capacity for the protection path exists on link j */*
If ($sh_j^W \leq B(W)$) then $cp_j^W \leftarrow c_j \cdot [1 - sh_j^W / B(W) + (V_j + sh_j^W)/V_j]$;
/ partly sharable and partly free capacity; the cost is determined by the percentage of free and sharable capacity; in case $V_j == S_j^W$, $cp_j^W \leftarrow c_j$ */*
If ($sh_j^W > B(W)$) then $cp_j^W \leftarrow c_j \cdot B(W)/(V_j + B(W))$;
/ In the case that enough sharable spare capacity available the cost is determined by the ratio between the B(W) and V_j ; */*
If ($V_j == S_j^W$) then $cp_j^W \leftarrow c_j$;
Else $cp_j^W \leftarrow \infty$;
/ the jth link does not have enough capacity; */*
End If
End For; */* the cost for all link l is defined */* Loop (1)
Return cp_j^W for $j \in E$;

With the network topology cost by the spare link-state cp_j^W for $j \in E$, the best protection path for *W* can be solved by using Dijkstra's shortest path first algorithm. The above pseudo-code and the definition of cost function can only work for the case with unity and discrete-bandwidth provisioning (e.g., lightpaths in the optical domain), but also the connections with continuous bandwidth. However, in the following

discussion, we will only focus on the former case where the setup of a working and protection lightpath-pair is requested. In this case, each connection request is for a single lightpath with uniform bandwidth.

III FINDING OPTIMAL WORKING AND PROTECTION PATH-PAIR

A Integer Linear Programming Formulation

In this subsection, an Integer Linear Programming formulation is presented for solving the optimal diverse routing problem, where a commercially available ILP solver (i.e., CPLEX 7.5 in this case) is adopted to jointly determine the optimal link-disjoint working and protection path-pair upon a connection request according to the current link-state.

The following symbols are adopted to develop the ILP formulation. Let the set of directional links and nodes in the network be denoted as E and N, respectively, the source and destination be denoted as s and d, and $x_{a,b}$ (or $y_{a,b}$) be the binary variable which is 1 if the flow for the working (or protection) path passes the directional link (a,b). The symbol $c_{a,b}$ is the cost of taking a link for the working path. Note that $c_{a,b}$ is a parameter determined by the current link-state and network topology (and can be treated as a constant in the formulation), while $z_{a,b}$ is the cost for the protection path to take link (a,b). Note that $z_{a,b}$ is dependent on the location of working flows (i.e., $x_{a,b}$). A small non-zero additional e cost is imposed when the protection path consumes sharable spare capacity to minimize its length, which is defined in such a way that $e \cdot |E| < \min_{(a,b)\in E} c_{a,b}$. In other words, e is a parameter only significant to the selection of protection paths. The symbol $sh_{a,b}^{u,v}$ for all edges $(a,b)\in E$ and $(u,v)\in E$ is a pre-calculated $|E|\times|E|$ array, which is the number of sharable wavelength channel along link (u,v) in case the corresponding working lightpath passes on link a to b. $sh_{a,b}^{u,v}$ can be determined with the following formula: $sh_{a,b}^{u,v} = V_{a,b} - SB_{a,b}^{u,v}$, where $SB_{a,b}^{u,v}$ is defined in the previous section and is the number of spare channels at link u,v, which protect the working capacity on link a,b and $V_{a,b}$ is the spare capacity on link a,b. Note that the preparation of $sh_{a,b}^{u,v}$ can be determined by the current link-state before the connection request arrives (or in the *Inter-arrival planning* [15]).

The Integer programming formulation is as follows:

Target function: minimize $\sum_{(a,b)\in E} c_{a,b} \cdot (x_{a,b} + z_{a,b}) + e \cdot y_{a,b}$, (6)

which is subject to the following constraints

1. The flow conservation constraint:

$$\sum_{b\ s.t.(a,b)\in E} x_{a,b} - \sum_{a\ s.t.(b,a)\in E} x_{b,a} = \begin{cases} 1 & \text{s} == a \\ -1 & \text{d} == a \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } a \in N \tag{7}$$

$$\sum_{v\ s.t.(u,v)\in E} y_{u,v} - \sum_{v\ s.t.(v,u)\in E} y_{v,u} = \begin{cases} 1 & \text{s} == v \\ -1 & \text{d} == v \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } v \in N \tag{8}$$

2. The constraint that working and protection paths should be link-disjoint:

$$x_{a,b} + y_{a,b} \le 1 \quad \text{for } \forall (a,b) \in E \tag{9}$$

3. The constraint that both flows for working and protection paths should be integer (or the flow integrity constraint):

$$x_{a,b},\ y_{a,b} \in \{0,1\}, \quad z_{a,b} \ge 0 \quad \text{for } \forall (a,b) \in E \tag{10}$$

4. The SRLG constraint 9

$$x_{a,b} + y_{u,v} - z_{u,v} \le 1 + sh_{a,b}^{u,v} \quad \text{for all edges } (a,b) \in E \text{ and } (u,v) \in E \tag{11}$$

In the above formulation, Eq. (7) and (8) are for the flow conservation constraint, which ensure the path requirement for both working and protection paths. Eq. (9) and (10) are to ensure the disjointness of the working and protection paths, and the integrity of the traffic flows, respectively. Eq. (11) states that the cost for the protection path taking link (u,v) has a lower bound of $x_{a,b} + y_{u,v} - 1 - sh_{a,b}^{u,v}$ for all edges (a,b) and (u,v).

Since the solving of the ILP formulation is time-consuming and can hardly be used as an on-line algorithm for large-sized networks (please see the discussions in Section IV), in the following sections we will provide two heuristic algorithms for this problem.

B Iterative Two-Step-Approach (ITSA)

We have defined the Two-Step-Approach for shared protection in Section II, where the ***spare link-state*** is determined for solving the best protection path with a given working path. However, the derived working and protection path-pair may be far from the optimal in this way. Even worse, in some cases the algorithm cannot find a disjoint path-pair even if one is available. The following figures show an example. In Fig. 5(a), the shortest path W1 between the source node A and the destination node B is found for the working path, which, however, isolates A and B. In this case, no disjointed path-pair can be found. Another example in Fig. 5(b) shows that the derived working-protection path-pair may be far from the optimal if there is an alternate path connecting node A and B. Both of the cases impair the overall performance.

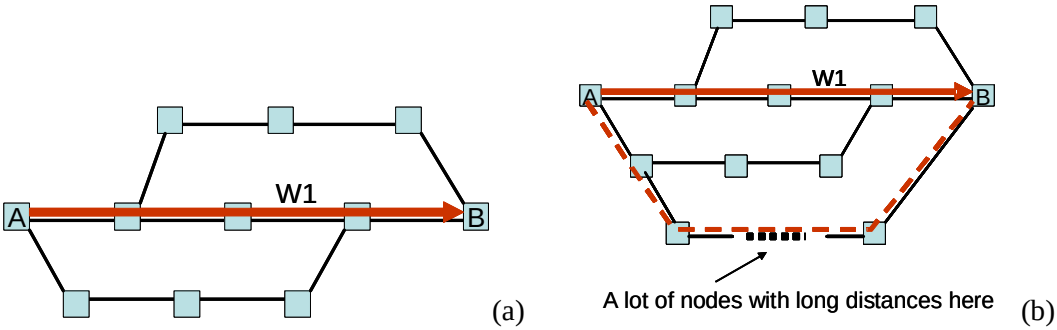


Fig. 5. Examples to show the shortages of the Two-Step-Approach

These shortages motivate us to enhance the conventional Two-Step-Approach. The algorithm, ITSA, can guarantee the derivation of the best working and protection path-pair under the current link-state by iteratively inspecting k-shortest paths as working paths in an ascending order of cost between the source and destination, where the Two-Step-Approach is invoked in every iteration. The iterative process does not end until certain pre-defined requirements are met, such as when the optimality of the derived path-pair has been achieved or a certain number of working paths have been inspected. A flowchart for the process of the ITSA algorithm is shown in Fig. 6.

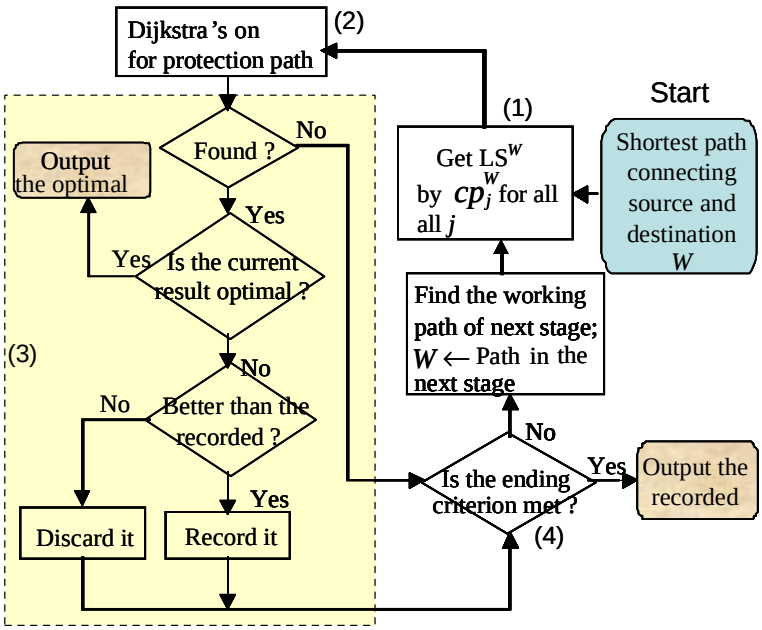


Fig. 6. A flowchart of the Iterative Two-Step-Approach

In the flowchart, the algorithm takes the shortest path as the working path corresponding to the connection request in the first iteration, which is denoted as W. The spare link-state cp_j^W for $j \in E$ is derived according to W (which is shown in the rectangle (1)). The rectangle (2) is to find the protection path using Dijkstra's algorithm on the residual graph after excluding the physical edges taken by W and remarking each residual edge with the spare link-state. At this moment the working and protection path-pair has been derived for this iteration. The big rectangle (3) contains a series of functions to

perform the optimality check for the newly derived path-pair, compare the derived path-pair at this iteration with the stored one in terms of their cost (in the case of the first iteration, the cost of stored one is infinity), and replace the stored one with the newly derived path-pair if the newly derived path-pair has less cost. If the algorithm proceeds (i.e., the newly derived one is not the optimal), then the algorithm checks the pre-defined ending criterion to possibly terminate the iterative process in triangle (4). If not, the algorithm solves for the second shortest path denoted as *W*, and starts the next iteration.

The functions of the optimality check and the ending criterion are presented. Without loss of generality, an example showing the approach of the optimality check is demonstrated in Fig. 7. Let the best solution recorded at that iteration be $C_{recorded} + CP_{recorded} = 5$, where $C_{recorded}$ and $CP_{recorded}$ are the cost for the recorded working and protection paths, respectively. In the event that the working path under inspection in the current iteration has a cost larger than or equal to 5, it is impossible to derive a better disjointed path-pair than the recorded one even if we can find a totally free protection path in the current or subsequent iterations. Therefore, the iterative process can stop since the optimal solution is the recorded one. Note that in order to guarantee the optimality for the derived path-pair, we may need to perform much more iterations to verify the optimality rather than for just deriving it. As shown in this example, the algorithm could have stopped and returned the derived solution when it finds $CP_{recorded}$ and $C_{recorded}$ (in which the cost for the working path is 3). However, to ensure the optimality, it cannot terminate the iterative process until the iteration yielding the working path with a cost 5 (i.c., $C_{current}$). To improve this situation, the ending criterion is devised to tolerate the sub-optimality of the stored path-pair such that the number of unnecessary iterations can be reduced.

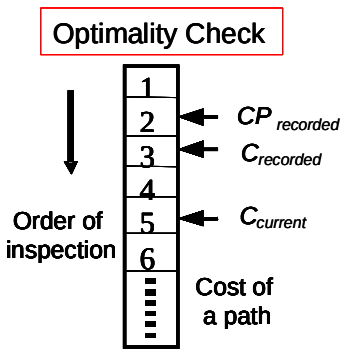


Fig. 7. An example of the optimality check in ITSA.

In general, the computation complexity for deriving each k-shortest path is $O(|N|^2 \cdot \log|N|)$ with Yen's algorithm [16], where *N* is the number of nodes in the network. The total computation complexity is $O(P_v \cdot |N|^2 \cdot \log|N|)$, where P_v is the number of k-shortest paths inspected before the optimal solution is *derived* and *verified*. It is clear that P_v is strongly determined by the network topology, which yields an upper bound growing exponentially with the size of the network. We need an approach that can further reduce the total computation time.

C Maximum Likelihood Relaxation (MLR)

This subsection introduces the MLR scheme, which is a modified Dijkstra's algorithm carrying/handling some additional information during the Dijkstra's relaxation process. The main idea for MLR is that the working path is selected such that *the number of links without enough sharable spare capacity* and working link cost are jointly considered. To be more concise, we called the links in the network with enough sharable spare capacity for the protection path of a working path segment, *ws*, as "*Easy Links*" of *ws*. During the Dijkstra's relaxation process, when node *n* is given a temporary label through link (*x*,*n*),by node *x*, a new working path segment from the source node *s* to node *n* by way of node *x* (denoted as $p(s,x) \cup (x,n)$) is formed. We have the following relationship:

$$S_j^{p(s,x)\cup(x,n)} \geq S_j^{p(s,x)} \text{ for all link } j \in E \text{ and node } n \text{ is not on the path segment } p(s,x) \quad (12)$$

Please see Eq. (5) for the definition of $S_j^{p(s,x)\cup(x,n)}$ and $S_j^{p(s,x)}$. Eq. (12) holds due to the reason that when node *x* gives a temporary label to node *n* and the resultant path segment $p(s,x) \cup (x,n)$, the working paths passing through link (*x*,*n*) are newly included into the SRLG, which yields a fact that some sharable spare capacity in the network may become non-sharable along some Easy Links for $p(s,x)$.

Based on the above discussion, it is clear that during the relaxation process, the amount of sharable spare capacity is getting less and the number of Easy Links is getting smaller. Therefore, one of the objectives in the proposed Dijkstra's relaxation process is to find a working path such that the Easy Links can be as many as possible. In addition, we need to consider the cost of each link, $c_{a,b}$, for $(a,b) \in E$, such that having a long working path is discouraged. In this study, the label (denoted as $L[n]$ for node *n*) given by node *x* in the Dijkstra's relaxation process is defined as the *link cost* $c_{x,n}$ *divided by the log of the number of Easy Links for* $p(s,n)$. The label replacement at node *n* by node *x* will be conducted in such a way that $L(n)$ is the minimal; i.e.,

$$L[n] = \min\{L[n], L[x] + c_{x,n} / \log(\mathit{offset}(x,n)+1)\}, \quad (13)$$

where *offset* (*x*,*n*) is the reduction on the number of Easy Links for $p(s,x) \cup (x,n)$. We have an expression for *offset*(*x*,*n*) as follows:

$$\begin{aligned} \mathit{offset}(x,n) = \{\sum_{j\in E} \mathrm{sgn}(j_j^{LW'}) \,\Big|\, j_j^{LW'} \leftarrow V_j - \max_{l\in LW'} SB_l^j - B(W), \text{ where } LW' = p(s,x)\} - \\ \{\sum_{j\in E} \mathrm{sgn}(j_j^{LW'}) \,\Big|\, j_j^{LW'} \leftarrow V_j - \max_{l\in LW'} SB_l^j - B(W), \text{ where } LW' = \pi(s,x) \cup (x,n)\} \end{aligned} \quad (14)$$

where the function sgn(*x*) returns 1 if $x \geq 0$, and 0 otherwise; V_j is the spare capacity along *j* and can be derived in the link-state database; SB_l^j, as defined in Section II, is the number of spare channels at *j* which protect the working capacity along *l* before the arrival of the current connection request, and is supposed to be non-sharable if *W* traverses *l*. Both V_j and SB_l^j can be derived off-line (i.e., before the

connection request is launched).

The algorithm proceeds as follows: at the beginning, $L[n] = 0$ for $n = s$ and $L[n] = \infty$ otherwise. The extra information other than the ordinary Dijkstra's relaxation process required to be recorded is an $|E| \times |p(s,x)|$ array storing SB_l^j, where $l \in p(s,x)$ and $j \in E$. When the relaxation process attempts to replace the label of node n from node x, the "MAX" operation for the array storing SB_l^j will have to be performed for $l \in p(s,x)$ and $l \in p(s,x) \cup (x,n)$ so that Eq. (13) can function. In addition, when a node is relaxed, the array storing SB_l^j is also updated. After the derivation of the working path, the corresponding protection path can be derived by Two-Step-Approach defined in Section II.

The computation complexity in implementing the MLR algorithm is $O(|E| \cdot |N|^2 \cdot \log|N|)$. To see its detail, the complexity for performing the regular Dijkstra's algorithm yields $|N| \cdot \log|N|$. An extra computation effort to scan and update the array of SB_l^j is required each time when a temporary label is sent, which yields computation complexity $O(|E| \cdot |N|)$. Therefore, the computation complexity for the MLR scheme is $O(|E| \cdot |N|^2 \cdot \log|N|)$ in the worst case.

The MLR method cannot guarantee the derivation of the best working and shared protection path-pair. However, the computation efficiency can be tremendously improved compared with all the other two proposed schemes. We will verify the schemes with simulation in the subsequent section.

IV. PERFORMANCE EVALUATION

Experiments are conducted to verify the ILP formulation (denoted as ILP in the following context) and the ITSA and MLR algorithms on four networks with 22, 30, 79, and 100 nodes with Ultra-80 SUN workstations. The network topologies are shown in Fig. 8. Assume each directional link in the network supports 32 independent connections. We first examine the capacity efficiency in terms of blocking probability for the dynamically arrived connection requests following the Poisson model and a holding time with an exponential distribution function. Let node pair (*i,j*) be subject to traffic load $r_{i,j} = h \cdot (l_{i,j} / m_{i,j})$, where $l_{i,j}, m_{i,j}$ are arrival and departure rate upon the node pair (*i,j*), respectively. $m_{i,j}$ is set to 1, while $l_{i,j}$ is a random number between 0.5~1.5, for all node (*i,j*). The scaling parameter h represents the traffic load level of the network. Therefore, each node pair yields different bandwidth demand. Without losing of generality, every connection request is a single lightpath that occupies a wavelength channel as traversing through the corresponding link. For the ITSA scheme, in order to limit the computation time, the maximum number of allowable iterations for solving a connection request is 50 for each case.

For a comparison purpose, we also implement the scheme provided in [11] (denoted as APF-PBC) and the scheme without any resource sharing (denoted as NS). With APF-PBC, the cost function for the working path is $L_l(W) = F_l \cdot (1 + \frac{\max_j S_l^j}{\max_l \max_j S_l^j})$, where F_l is the residual bandwidth of link l, S_l^j is the total spare capacity required along link j to protect the working capacity along link l. With NS, the Suurballe's algorithm [1] is adopted to find the least-cost working and protection path-pair in the network, where the resource sharing is not allowed. The link cost c_j in Eq. (1) and (2) and $c_{a,b}$ in Eq. (5) and Eq. (13) is a random positive integer between 1 and 100 for all j and (a,b) when implementing NS, ITSA, MLR, and ILP.

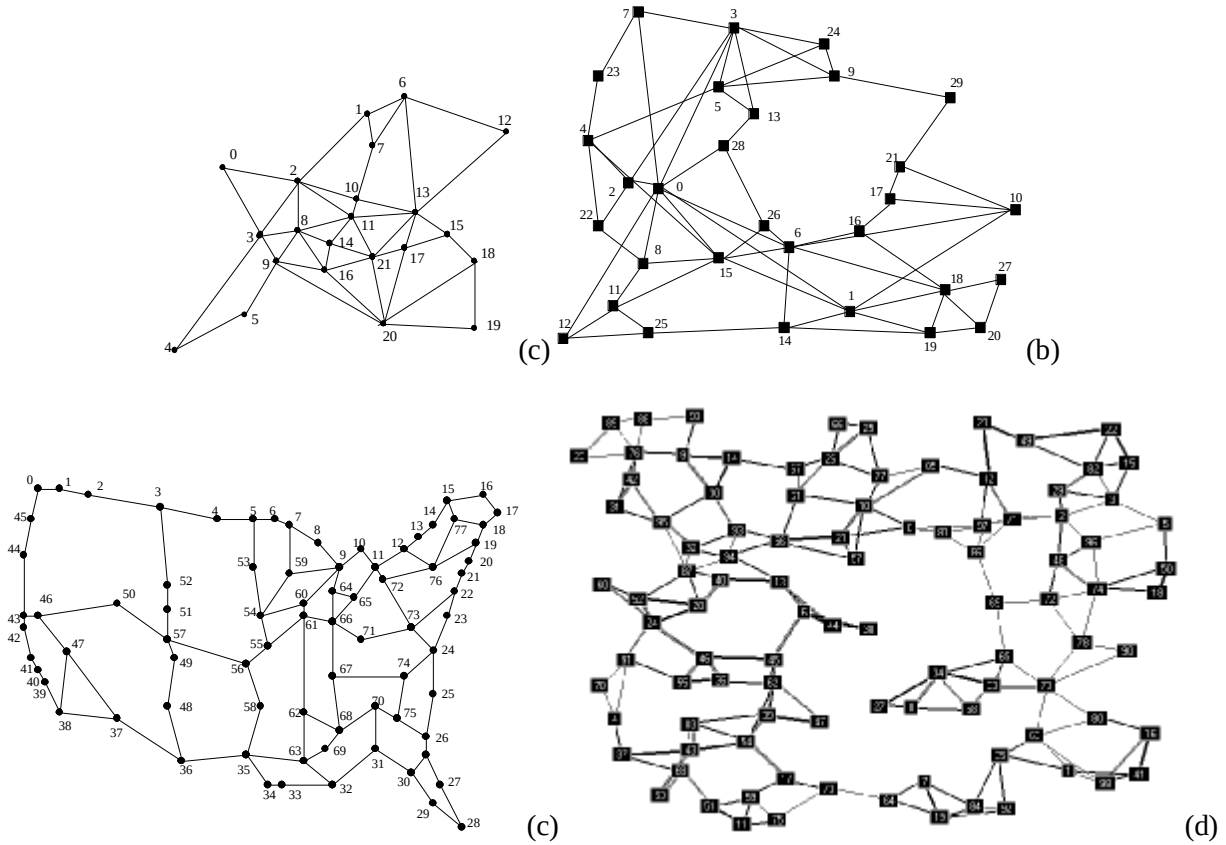


Fig. 8(a)-(d). 22-node, 30-node, 79-node and 100-node network topologies.

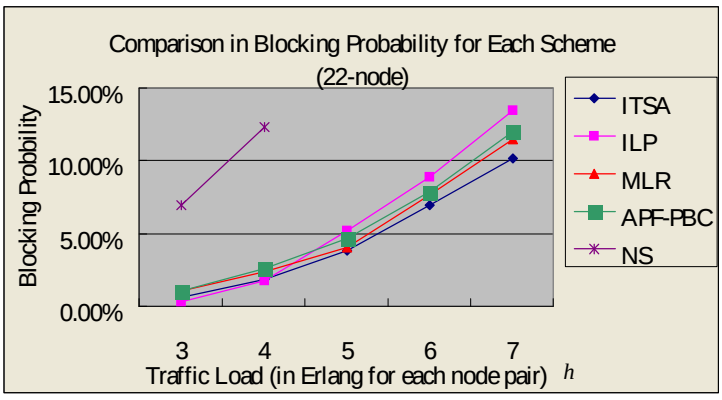
Fig. 9(a) ~ Fig. 9(d) show the simulation results for all the schemes in terms of blocking probability on the four network topologies. Since the solving of the ILP takes computation time much longer than are the cases of the other heuristics, we only verify ILP with the 22-node and 30-node networks as shown in Fig. 9(a) and (b). The computation time for solving the ILP formulation on the 22-node and the 30-node networks is around several seconds and 10~20 seconds by using CPLEX 7.5 on an Ultra 80 SUN Workstation with 4 GB memory. However, it may take more than half an hour to allocate working and protection path-pair for a single connection request upon the 100-node network.

It is clear from Fig. 9 that ITSA yields the best performance while NS yields the worst under the provisioned traffic load and the given network topologies. It is worth noting that although ILP can

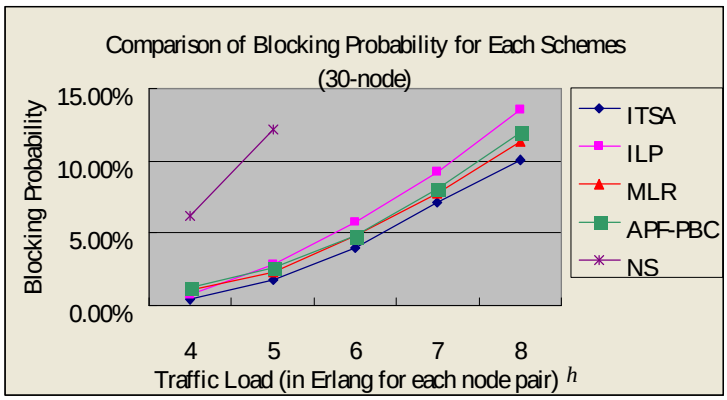
provide the optimum allocation of working and shared protection paths for each connection request, the overall performance is outperformed by ITSA, MLR, and APF-PCB when the traffic load is high. We find out the reason for this unexpected and non-intuitive result as follows: The protection paths in ILP can be very long due to its non-adaptation of the sharable spare capacity with the cost of the working paths. It is extremely hard to find a suitable scale between the cost of working and protection paths. As mentioned earlier in the paper, a long protection path consuming a large amount of sharable spare capacity may increase the potential non-sharable spare capacity for the subsequent connection requests. As a result, the overall performance can be significantly impaired. This effect deteriorates the performance especially when the traffic load is high, in which the algorithm tries its best to find links with sufficient sharable spare capacity in the network to yield a very long protection path. However, if an attempt is made to impose an adaptive link cost on the protection path when it takes sharable spare capacity, the formulation becomes non-linear and can hardly be solved by any Linear Programming solver.

From the experiment results the ITSA scheme can find a solution very close to the optimal one with 50 iterations allowed before terminating the algorithm, which, however, differs from the ILP formulation in that it adopts the adaptive spare link-state (shown in Eq. (3)). As a result, a proper weighting on taking a spare link by a protection path is imposed, which yields the best performance among the five. It is observed from the experiment that the average length (in hops) of the path-pairs in the ITSA scheme is significantly less than is the case derived in the ILP formulation.

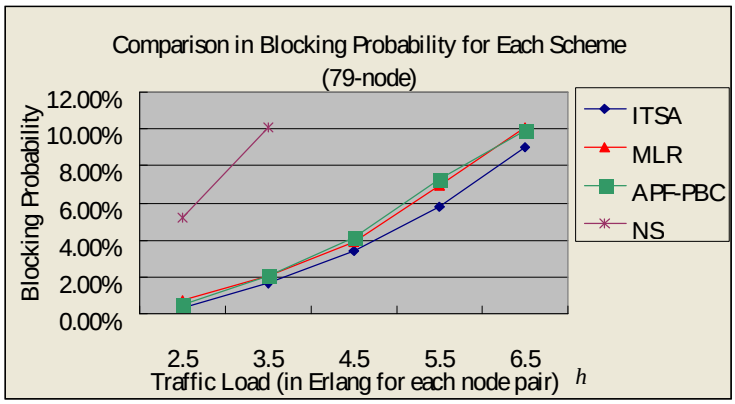
It is also worth noting that MLR slightly outperforms APF-PBC because the former provides a specific path-based approach during the selection of a working path such that the corresponding protection path can find maximum number of links with enough sharable spare capacity in the network. The latter scheme, on the other hand, encourages the selection of the working path minimizing the sum of the maximum non-sharable spare capacity along each link, which is nonetheless not quite straightforward and may be far from being an effective approach to improve the resource sharing.



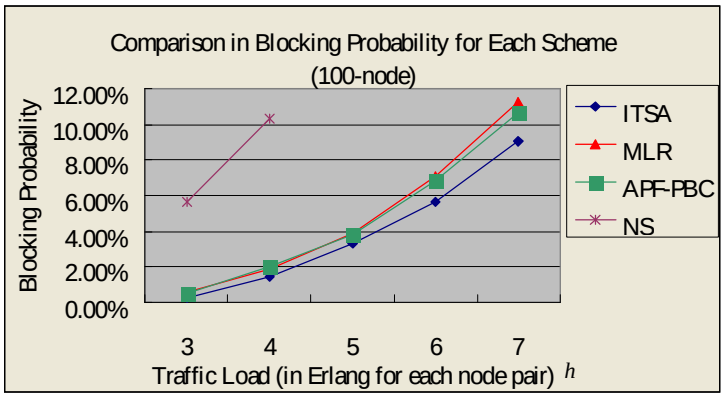
(a)



(b)



(c)



(d)

Fig. 9(a) ~ (d). Simulation results on the four network topologies with different traffic load.

The average computation time spent for allocating each connection request on each of the experiments is shown in Fig. 10. We do not show the case of ILP because it yields too long a computation time to have any meaning by including it into the comparison. From the results, the NS consumes the least amount of time since no resource sharing needs to be investigated. Both MLR and APF-PBC only invoke Dijkstra's algorithm twice, which are scalable to the network size. MLR needs an extra amount of time to inspect the number of Easy Links in the network during the Dijkstra's relaxation process. The construction of spare link-state takes a worst-case complexity of $O(|E| \cdot |N|)$, where the factor $|N|$ comes from the length of the working path, while $|E|$ is the complexity taken to

mark each directional link in the network with non-sharable spare capacity for every link traversed by the working path. However, the practical implementation can be much faster, because neither a working path can be as long as $|N|$ hops, nor can the number of links that need to be re-marked with a new sharable spare capacity in each iteration of relaxation be as large as $|E|$. The NS case is even faster with a complexity $O(|E| \cdot \log_{(1+|E|/|N|)}|N|)$ in implementing Suurballe's algorithm [16]. The complexity in performing ITSA is much larger than the other heuristics due to its nature of iterative trial-and-error. The number of iterations required to find and verify the optimality in each case is strongly determined by the network size and the length of the working path. It is observed not practical to pursue the optimal solution in large-sized networks since the number of iterations required to verify the optimality can be more time-consuming than that an on-line algorithm should be afforded.

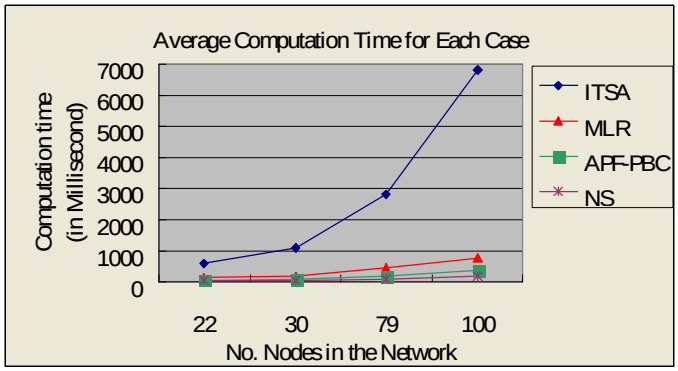


Fig. 10. Comparison of computation time between each case.

IV. CONCLUSIONS

In this paper we study the diverse routing problem for shared protection dealing with dynamic connection requests arriving at networks one after the other without any prior knowledge of future arrivals. A suite of algorithms are presented and verified with extensive simulation. We first define Two-Step-Approach in shared protection, which can explore the maximum extent of resource sharing for a protection path given the working path. To jointly consider the working and protection paths, we first formulate the diverse routing problem into an Integer Linear Programming (ILP) process, in which the working and shared protection paths corresponding to a connection request are solved in a single step. Due to its time-consuming solving process and bad scalability with the network size induced by the ILP formulation, we propose two heuristic algorithms, called Iterative Two-Step-Approach (ITSA) and Maximum Likelihood Relaxation (MLR), aiming at the improvement of the computation efficiency without losing much performance. The ITSA scheme iteratively takes k-shortest path for inspection, and can guarantee the derivation of the best solution. The MLR scheme, on the other hand, is a modification of the Dijkstra's algorithm jointly taking the link cost and the number of links with sufficient sharable spare capacity into consideration. To verify the proposed approaches, simulation is connected on four networks launched with dynamic traffic following the Poisson traffic model. The simulation results

show that the ITSA scheme can achieve the best performance at the expense of much longer computation time. On the other hand, MLR can provide an ultra-fast path selection process, which behaves as a good tradeoff between computation efficiency and performance. It is worth noting that although the ILP formulation can achieve the greediest path selection process, the overall performance in terms of blocking probability is outperformed by the other heuristic counterparts due to the non-adaptation in the cost imposed on the sharable spare capacity.

Acknowledge

This work has been partly sponsored by National Science and Engineering Research Council (NSERC), Canada, grant No. 109968; and partly sponsored by OTKA 30122 of the Hungarian National Science and by High-Speed Networks Laboratory (HSN*Lab*).

REFERENCES

[1] R. Bhandari, Survivable networks: algorithms for diverse routing, *Kluwer Academic Publishers*, Boston, 1999.

[2] C. Xin, Y. Ye, S. Dixit, and C. Qiao, "A Joint Lightpath Routing Approach in Survivable Optical Networks," Optical Network Magazines, May/June, 2002, pp. 23-32.

[3] E. Q. V. Martins and M. M. B. Pascoal, "A new implementation of Yen's ranking loop-less paths algorithm", *Optimization 2001, Aveiro*, July 2001

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

[5] S. Datta, S. Sengupta, and S. Biswa¶, "Efficient Channel Reservation for Backup Paths in Optical Mesh Networks," *Proceedings IEEE Globecom'01*, San Antonio, Texas, Nov. 2001, OPC01-7.

[6] J. Tapolcai, P. Laborczi, P. -H. Ho, T. Cinkler, A. Recski, and H. T. Mouftah, "Algorithms for Asymmetrically Weighted Pair of Disjointed Paths in Survivable Networks", *Proceedings Third International Workshop on Design of Reliable Communication Networks (DRCN 2001)*, Budapest, Hungry, Oct. 2001, pp. 239-249.

[7] 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.

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

[9] P. –H. Ho and H. T. Mouftah, "Issues on Diverse Routing for WDM Mesh Networks with Survivability", *Proceedings IEEE 2001 International Conference on Computer and Communication Networks (ICCCN'01),* Scottsdale, AZ, Oct. 2001, pp. 60-65.

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

[11] D. Xu, C. Qiao, and Y Xiong, "An Ultra-fast Shared Path Protection Scheme -- Distributed Partial Information Management, Part II", *Proceedings IEEE International Conference on Network Protocols (ICNP 2002)*, Paris, France, Nov. 2002.

[12] 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.

[13] 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.

[14] P. –H. Ho and H. T. Mouftah, "Allocation of Protection Domains in Dynamic WDM Mesh Networks", *ICC 2003 (accepted)*.

[15] P. –H. Ho and H. T. Mouftah, "A Framework of Scalable Optical Metropolitan Networks for Improving Survivability and Class of Service", *IEEE Network, Special issue on Scalability in Communication Networks*, July/Aug. 2002, pp. 29-35.

[16] J. Y. Yen, "Finding the k shortest loopless paths in a network", *Management Science*, vol. 17, pp. 712-716, 1971.

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