Wavelength Rerouting in Survivable WDM Networks

Yingyu Wan and Weifa Liang

Department of Computer Science,
The Australian National University,
Canberra, ACT 0200, Australia

Abstract. One limitation of all-optical WDM networks is the wavelength continuity constraint imposed by all-optical cross-connect switches that requires the same wavelength be used on all the links along a path. With random arrivals and departures of connection requests, it happens quite often that a new request has to be blocked due to the fact that there are not enough available resources (e.g. wavelength) to accommodate the request. Wavelength rerouting, a viable and cost-effective method, which rearranges the wavelengths on certain existing routes to free a wavelength continuous route for the new request, has been proposed to improve the blocking probability. In this paper, we study a wavelength rerouting problem in survivable WDM networks as follows. Given a connection request, the problem is to find two link-disjoint paths from the source node to the destination node with an objective to minimize the number of existing routes that have to be wavelength-rerouted. We show that the problem is NP-hard if different wavelengths are assigned to the link-disjoint paths. Otherwise, a polynomial time algorithm is proposed.

1 Introduction

A Wavelength Division Multiplexing (WDM) network consists of optical wavelength routing nodes interconnected by point-to-point fiber links in an arbitrary topology. On each fiber link, the fiber bandwidth is divided into multiple frequency bands (wavelengths) so that several connection requests can be realized over it at the same time, as long as each uses a different wavelength. There are two types of WDM networks, one allows wavelength conversion at its nodes and the other does not. In a network without wavelength conversion, the realization of a connection request is subject to the *wavelength continuity constraint* that the same wavelength is used on all the links in the route for the request. This constraint often reduces the wavelength utilization because a non-wavelength-continuous route cannot be used even if it is available. This is especially severe in a network with random arrivals and departures of requests. Although wavelength conversion can potentially allow the network to accommodate more requests, it is expected that the wavelength converters at nodes are expensive in the near future. Hence, most existing work assumes that no wavelength conversion in the network is allowed. This assumption will be adopted by this paper as well.

To alleviate the inefficiency brought by the wavelength continuity constraint, a viable and cost-effective method called *rerouting* has been proposed, which is described as follows. Whenever a new request arrives, if there is no wavelength-continuous route for it, rearrange a certain number of existing routes to free a wavelength for the request. There are two ways to rearrange an existing route. One is "fully rearranging", which is to find a new route with another wavelength to replace the old route. This is also referred to as *nonblocking rearrangement.* Another is "partially rearranging", which keeps the original route but reassigns a different wavelength to the links in the route. This latter one is referred to as *wavelength rerouting.* Examples of nonblocking rearrangement and wavelength rerouting can be found in [1, 2, 3] and [4, 5, 6, 7, 8] respectively. While rerouting can be used to improve the bandwidth utilization, transmission on each rerouted route must be temporarily shut down to prevent data from being lost during the rerouting processing. This causes a low or even zero throughput on those being rerouted traffic. The throughput loss is particularly prominent in all-optical networks wherein each routing path is expected to carry gigabits of data flow per second, and hence even a tiny period of outage on a single route will cause significant data loss. Thus, to minimize this disruptions (i.e., the number of rerouted routes) is of paramount importance for rerouting in all-optical WDM networks. It is a general belief that nonblocking rearrangement is much harder than wavelength rerouting because in the former not only a new route for a connection request needs to be built but also another available wavelength needs to be assigned to each of the links in the new route. Despite the fact that nonblocking rearrangement may improve the blocking probability significantly, it leads to a much longer disruption than expected. In particular in an arbitrary topology network, it is very difficult to make a nonblocking rearrangement for a new request with the minimum disruption. In reality, most known works on nonblocking rearrangement are carried out on special topology networks like rings and tori [2]. On the other hand, the survivability of routing is a critical design problem for high-speed WDM networks. To protect a mission-critical connection from a single link failure, a common solution is to find a pair of link-disjoint paths from a source node to a destination node. One of the paths serves as the *active path* (or **AP**) which will be used for the data transfer actually. The other serves as the *backup path* (or **BP**) once the active path is disconnected due to a link failure on it. When a mission-critical request arrives, if there are not enough wavelengths in the network to find the AP and BP pair for the request, it is possible (sometimes necessary) to reroute a certain number of existing traffic to free a wavelength for the request.

Lee and Li [5] first introduced the wavelength rerouting concept by studying the unicast routing problem with the objective to minimize the disruption incurred due to wavelength rearranging. For an undirected WDM network of n nodes, m physical links and w wavelengths on each link, they proposed a wavelength rerouting scheme called *Parallel Move-To-Vacant Wavelength-Retuning (MTV-WR)*, which has the following advantages. First, it facilitates control because the old and new routes of rerouted traffic share the same switching nodes.

Second, it reduces the calculation because only the wavelengths on the links of existing routes need to be changed. Third, it significantly reduces the disruption period. An algorithm for implementing the MTV-WR scheme has also been proposed, which takes $O(n^3w + n^2w^2)$ time per unicast request [5]. Mohan and Murthy [6] later provided an $O(n^2w)$ time improved algorithm for the problem. Caprara et al [7] studied the unicast routing problem in a directed WDM network by showing that the problem is not only NP-hard but also hard to approximate. Instead, they proposed an approximation algorithm. Wan and Liang recently [8] studied the wavelength rerouting for multicast requests in both undirected and directed networks. The two link-disjoint paths problem in a weighted graph with the objective to minimize the sum of the lengths (costs) of the two paths was well studied and can be solved in polynomial time [9]. If the objective is to minimize the length of either the longer one or the shorter one of the two paths, the problem was shown to be NP-hard [10, 11].

In this paper, our focus will be on wavelength rerouting in an arbitrary topology WDM network for link-disjoint paths routing. In particular, given a connection request, the problem is to find two link-disjoint paths for it such that the number of existing routes that need to be rerouted is minimized. Notice that the problem that we study here is different from those previous works [9, 10, 11, 12, 13, 14] that use different cost metrics. We show that the problem of concern is NP-hard if different wavelengths need to be assigned to the two paths. Otherwise, a polynomial time algorithm is devised, which takes $O(m^5 \log n)$ time, where m and n are the number of links and nodes in the network.

The rest of this paper is organized as follows. In Section 2 we define the network model and provide an overview of the wavelength rerouting scheme for link-disjoint paths routing. In Section 3 we provide an NP-hard proof of the problem. In Sections 4 we propose a polynomial algorithm.

2 Preliminaries

Network Model: An undirected WDM network $M = (N, L)$ without wavelength conversion is considered, where N is the set of communication nodes and L the set of communication links. The bandwidth of each link is divided into a set of w wavelengths $\Lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_w\}$. It is assumed that each node has sufficient optical transmitters and receivers so that no connection request will be blocked due to lack of such resources. A connection request is denoted by a pair (s, t), where s is the source node and t is the destination node. At the time a request arrives, on a given link $e \in L$ there may be some wavelengths unavailable due to that they have been occupied by the other existing routes. Denote by Λ_e ($\subseteq \Lambda$) the set of available wavelengths on link e. Let $\mathbf{P}$ be the set of existing routes (*lightpaths*). Depending on the wavelengths used by the lightpaths, $\mathbf{P}$ can be partitioned into w disjoint subsets $\mathbf{P}_1, \mathbf{P}_2, \cdots, \mathbf{P}_w$, where the links in each path in $\mathbf{P}_i$ is assigned wavelength λ_i, $1 \le i \le w$. If two paths share at least one link, the paths are said to *link-intersect* with each other, otherwise they are *link-disjoint*. Obviously, any two lightpaths in set $\mathbf{P}_i$ are pairwise link-disjoint.

```
Wavelength Rerouting Scheme for Link Disjoint Paths
1. For each wavelength $\lambda_i \in \Lambda$ do
1.1  For each lightpath $P \in \mathbf{P}_i$, if there is another available wavelength $\lambda_j \in \Lambda$ $(i \neq j)$
       on each link in $P$, i.e., $\forall e \in E(P), \lambda_j \in \Lambda_e$, then $P$ can be assigned wavelength $\lambda_j$,
       and $P$ is said to be tunable. Otherwise, $P$ is said to be untunable. Let $\mathbf{P}'_i$ be the
       subset of $\mathbf{P}_i$ containing all the tunable lightpaths, and $\mathbf{P}''_i = \mathbf{P}_i - \mathbf{P}'_i$.
2. Find a pair $\{AP_i, BP_j\}$ of lightpaths between $s$ and $t$ with minimum cost such that
   (i) $AP_i$ and $BP_j$ are link-disjoint;
   (ii) wavelength $\lambda_i$ $(\lambda_j)$ is assigned to each link in $AP_i$ $(BP_j)$;
   (iii) $AP_i$ $(BP_j)$ does not link-intersect with any lightpath in $\mathbf{P}''_i$ $(\mathbf{P}''_j)$;
   (iv) the cost is defined as $|\{P \in \mathbf{P}'_i : P \cap AP_i \neq \emptyset\} \cup \{P \in \mathbf{P}'_j : P \cap BP_j \neq \emptyset\}|$, where
        $P \cap AP_i \neq \emptyset$ or $P \cap BP_j \neq \emptyset$ mean that $P$ link-intersects with $AP_i$ or $BP_j$.
3. If no such a pair exists, then the request $(s, t)$ cannot be supported by the
   wavelength rerouting scheme and will be rejected. Otherwise, the request $(s, t)$
   can be realized by rerouting a certain number of existing traffic.
```

Fig. 1. Overview of the wavelength rerouting scheme

For the sake of convenience, denote by $E(P)$ and $V(P)$ the sets of links and nodes in a path P respectively in the rest of the paper.

Overview of the Wavelength Rerouting Scheme: To accommodate a new connection request (s, t) with the survivability requirement, we need to find two link-disjoint lightpaths (a link-disjoint AP and BP pair) in the current network between s and t, and the wavelength continuity constraint on both AP and BP is imposed. This can be achieved by two phases. Phase 1: find a link-disjoint AP and BP pair between s and t, using only those available wavelengths. Phase 2: find a link-disjoint AP and BP pair with rerouting some existing lightpaths if Phase 1 fails. Phase 1 is to find two link-disjoint paths using an available wavelength, which has been studied extensively [12, 13, 14]. We therefore focus on Phase 2. If Phase 1 fails, it means that there does not exist two link-disjoint paths in the network using only the available wavelengths, and a wavelength rerouting scheme is adopted, which is described in Fig. 1. The scheme first partitions each set $\mathbf{P}_i$ into two subsets, one containing tunable paths and the other containing untunable paths, then aims to find two link-disjoint paths for the new request (s, t) which link-intersect with the tunable paths only. At Step 2 of the scheme, to minimize the total cost of the two paths is equivalent to minimize the disruption for rerouting. This step can be implemented by finding a candidate solution with the minimum cost for every possible pair of λ_i and λ_j $(1 \leq i, j \leq w)$ and choosing one with the minimum cost from these candidate solutions. Depending on whether or not $i = j$, two combinatorial optimization problems arise from Step 2. (i) If $i \neq j$, a graph $G = (V, E)$ can be constructed as follows. $V = N$ and $E = E_i \cup E_j$, where $E_i = \{e \in L \mid \lambda_i \in \Lambda_e \text{ or } \exists P \in \mathbf{P}'_i, e \in E(P)\}$ and $E_j = \{e \in L \mid \lambda_j \in \Lambda_e \text{ or } \exists P \in \mathbf{P}'_j, e \in E(P)\}$.

Definition 1 (Minimum Disruption Link-Disjoint Paths 1, MDLDP1). *Given an undirected graph $G = (V, E)$ with $E = E_1 \cup E_2$, a collection $\mathcal{P}_1$ of link-disjoint paths satisfying that $\forall P \in \mathcal{P}_1, E(P) \subseteq E_1$, another collection $\mathcal{P}_2$ of* *link-disjoint paths satisfying that $\forall P \in \mathcal{P}_2, E(P) \subseteq E_2$, a source node s and a destination node t, the objective is to construct two link-disjoint paths AP and BP between s and t such that $E(AP) \subseteq E_1$, $E(BP) \subseteq E_2$ and the sum of the number of paths in $\mathcal{P}_1$ link-intersecting with AP and the number of paths in $\mathcal{P}_2$ link-intersecting with BP is minimized.*

(ii) Otherwise ($i = j$), a graph $G = (V, E)$ can be constructed as follows. $V = N$ and $E = \{e \in L \mid \lambda_i \in \Lambda_e$ or $\exists P \in \mathbf{P}'_i, e \in E(P)\}$.

Definition 2 (Minimum Disruption Link-Disjoint Paths 2, MDLDP2). *Given an undirected graph $G = (V, E)$, a collection $\mathcal{P}$ of paths which are pairwise link-disjoint, a source node s and a destination node t, the objective is to construct two link-disjoint paths AP and BP between s and t such that the number of paths in $\mathcal{P}$ that link-intersect with them is minimized.*

3 NP-Hard Proof of MDLDP1

We prove that MDLDP1 is NP-hard by a reduction from the following problem. Link Disjoint Path Problem in Graphs with Red, Green, Blue Links (LDP-PRGB).

Instance: A graph $G = (V, E)$, where each link $e \in E$ is colored red, blue or green; a source node s and a destination node t.
Question: Is it possible to establish two link-disjoint paths between s and t such that one of the paths uses only the red and green links and the other uses the blue and green links?

LDPPRGB was shown to be NP-hard by a reduction from the 3SAT problem in [14]. Given an instance of LDPPRGB, a corresponding instance of MDLDP1 can be constructed as follows. Let $G(V, E)$ be the graph with $n = |V|$ nodes and $m = |E|$ links in the given LDPPRGB instance. Define $V' = V$. Let E'_1 be the set of links in E colored red or green, E'_2 the set of links in E colored blue or green, and $E' = E'_1 \cup E'_2$. For each link $e \in E$ colored red, define a *red path* consisting only of link e. For each link $e \in E$ colored blue, define a *blue path* consisting only of link e. Let $\mathcal{P}'_1$ be the set of red paths and $\mathcal{P}'_2$ the set of blue paths. Then $G' = (V', E')$, $\mathcal{P}'_1$, $\mathcal{P}'_2$ and (s, t) form an instance of MDLDP1.

If there is a feasible solution AP and BP for the MDLDP1 instance, there is also a feasible solution for the LDPPRGB instance because $E(AP) \subseteq E'_1$ and $E(BP) \subseteq E'_2$. This reduction can be implemented in polynomial time, MDLDP1 thus is NP-hard, following that LDPPRGB is NP-hard.

Theorem 1. *MDLDP1 is NP-hard.*

4 A Polynomial Time Algorithm for MDLDP2

In this section we first give the structure properties of an optimal solution for MDLDP2. We then propose a polynomial time algorithm for the problem.

4.1 Structure Properties

Let $\{AP, BP\}$ be a feasible solution for an instance of MDLDP2. Define the cost of the solution as $c(AP, BP) = |\{P \in \mathcal{P} : P$ link-intersects with AP or $BP\}|$. For the convenience of description, we assume that $\{AP, BP\}$ is laid out in a plane with s and t on the left end and right end of AP and BP respectively which is illustrated in Fig. 2(a). Each path $P_i \in \mathcal{P}$ may link-intersect with $\{AP, BP\}$ or not. If P_i link-intersects with AP, define by $A_i = P_i \cap AP = \{P^a_{i,1}, P^a_{i,2}, \cdots\}$ the set of *maximal subpaths* shared by P_i and AP, where the "maximal subpath" means that there is no other shared subpath between P_i and AP which properly includes it. Without loss of generality, let $P^a_{i,1}, P^a_{i,2}, \cdots$ be indexed in the order from left to right along AP, i.e., a subpath closer to s has a smaller index and a subpath closer to t has a larger index. $B_i = P_i \cap BP = \{P^b_{i,1}, P^b_{i,2}, \cdots\}$ can be defined similarly. If P_i link-intersects with only either AP or BP but not both of them, P_i is called *single-intersected*. Otherwise, P_i is called *double-intersected*. If P_i is single-intersected and either $|A_i| = 1$ or $|B_i| = 1$, P_i is called *trivial-intersected*. If P_i is double-intersected, define an ordered relation on $A_i \cup B_i$ based on the order those subpaths appearing in P_i. To do so, first define a direction along P_i from one endpoint to another endpoint. For every two subpaths $P_{i,j}, P_{i,k} \in A_i \cup B_i$ in P_i, if $P_{i,j}$ appears before $P_{i,k}$ in the defined direction, $P_{i,j}$ is a *predecessor* of $P_{i,k}$, denoted by $P_{i,j} \prec P_{i,k}$, and $P_{i,k}$ is a *successor* of $P_{i,j}$, denoted by $P_{i,k} \succ P_{i,j}$. Denote by $prec(P_{i,j})$ and $succ(P_{i,j})$ the immediate predecessor and immediate successor of $P_{i,j}$ respectively if they do exist. If a double-intersected path P_i satisfies two conditions: **C1:** For any $P^a_{i,j} \in A_i$, $prec(P^a_{i,j}) \in B_i$ and $succ(P^a_{i,j}) \in B_i$ if they exist. For any $P^b_{i,j} \in B_i$, $prec(P^b_{i,j}) \in A_i$ and $succ(P^b_{i,j}) \in A_i$ if they exist. **C2:** $P^a_{i,1} \prec P^a_{i,2} \prec \cdots$ and $P^b_{i,1} \prec P^b_{i,2} \prec \cdots$, or $P^a_{i,1} \succ P^a_{i,2} \succ \cdots$ and $P^b_{i,1} \succ P^b_{i,2} \succ \cdots$. Then, P_i is called *regular-intersected*. Such an example is given in Fig. 2(b). Assume that both P_i and P_j are double-intersected. If there are two subpaths P^a_{i,k_1}, P^b_{i,k_2} in P_i and two subpaths P^a_{j,l_1}, P^b_{j,l_2} in P_j such that (i) $prec(P^a_{i,k_1}) = P^b_{i,k_2}$ or $succ(P^a_{i,k_1}) = P^b_{i,k_2}$, (ii) $prec(P^a_{j,l_1}) = P^b_{j,l_2}$ or $succ(P^a_{j,l_1}) = P^b_{j,l_2}$, (iii) P^a_{i,k_1} is on the left side of P^a_{j,l_1} in AP and P^b_{j,l_2} is on the left side of P^b_{i,k_2} in BP, then P_i and P_j are called crossing with each other. Then, we show that there is an optimal solution that meets the following properties.

Property 1. Any single-intersected path is also trivial-intersected.
Property 2. Any double-intersected path is regular-intersected.
Property 3. No two double-intersected paths cross with each other.

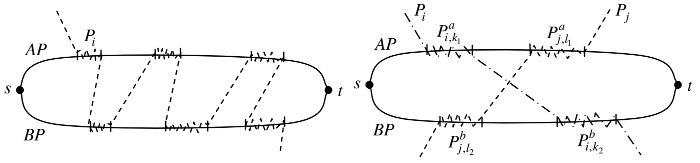

Fig. 2. (a) Regular-intersected path; and (b) P_i and P_j cross with each other

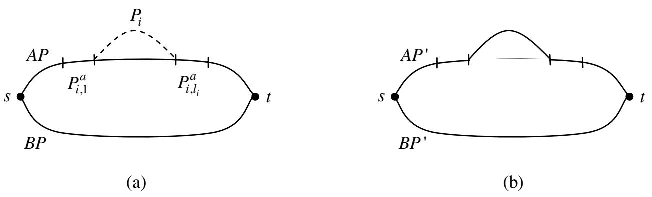

Fig. 3. Transformation to trivial-intersected path

Lemma 1. *For a feasible solution $\{AP, BP\}$, if any properties 1,2 and 3 is not met, then there is another feasible solution $\{AP', BP'\}$ such that $c(AP', BP') \leq c(AP, BP)$ and these three properties are met at the same time.*

Proof. The basic idea is to modify the solution step by step until all the properties are met. If Property 1 does not hold, without loss of generality, assume that P_i is single-intersected and $|A_i| > 1$. Let l_i be the maximum index in A_i. Let $P^a_{i,1}$ and P^a_{i,l_i} be the leftmost and rightmost subpaths of P_i in AP respectively. A new path AP' can be constructed as follows (see Fig. 3). It starts from s and traverses along AP. When it passes through $P^a_{i,1}$, it traverses along the links in P_i to P^a_{i,l_i}. It then traverses from P^a_{i,l_i} along AP to t. Because P_i is single-intersected, AP' and BP are link-disjoint. Let $BP' = BP$. A new feasible solution $\{AP', BP'\}$ is obtained. It is obvious that $c(AP', BP') \leq c(AP, BP)$ and P_i is trivial-intersected. If there are more than one single-intersected but non-trivial-intersected paths, repeat this procedure until each single-intersected path is trivial-intersected. If Property 2 does not hold, let P_i be double-intersected but non-regular-intersected. If Condition **C1** does not hold, without loss of generality, assume $P_{i,j} \in A_i$ and $succ(P_{i,j}) \in A_i$. Following the similar argument as for Property 1, $P_{i,j}$ and $succ(P_{i,j})$ can be merged into a subpath by a shortcut through the part of P_i between $P_{i,j}$ and $succ(P_{i,j})$.

Now assume that Condition **C1** but Condition **C2** holds. Consider $P^a_{i,1}$ and $P^b_{i,1}$ first. If the subpaths $prec(P^a_{i,1})$, $succ(P^a_{i,1})$, $prec(P^b_{i,1})$ and $succ(P^b_{i,1})$ exist at the same time, then $prec(P^a_{i,1}) \in B_i$, $succ(P^a_{i,1}) \in B_i$ and $prec(P^b_{i,1}) \in A_i$, $succ(P^b_{i,1}) \in A_i$. At least one of $prec(P^a_{i,1})$ and $succ(P^a_{i,1})$ is on the right of $P^b_{i,1}$ because $P^b_{i,1}$ is the leftmost subpath on BP, denoted by $P^b_{i,k}$. At least one of $prec(P^b_{i,1})$ and $succ(P^b_{i,1})$ is on the right of $P^a_{i,1}$ because $P^a_{i,1}$ is the leftmost subpath on AP, denoted by $P^a_{i,j}$. As shown in Fig. 4, a new feasible solution $\{AP', BP'\}$ can be obtained as follows. Traverse along AP from s to $P^a_{i,1}$, then traverse along P_i to $P^b_{i,k}$, finally traverse along BP from $P^b_{i,k}$ to t. Denote by AP' the resulting path. BP' can be obtained similarly. Repeat this procedure until at least one of the four subpaths $prec(P^a_{i,1})$, $succ(P^a_{i,1})$, $prec(P^b_{i,1})$ and $succ(P^b_{i,1})$ does not exist. Without loss of generality, assume $prec(P^a_{i,1})$ does not exist. If $succ(P^a_{i,1}) \neq P^b_{i,1}$, a similar transformation can be applied until $succ(P^a_{i,1}) = P^b_{i,1}$, i.e., $P^a_{i,1} \prec P^b_{i,1}$. Next consider $P^b_{i,1}$ and $P^a_{i,2}$, apply the transformation again such that $P^b_{i,1} \prec P^a_{i,2}$. Finally we have $P^a_{i,1} \prec P^b_{i,1} \prec P^a_{i,2} \prec P^b_{i,2} \prec \cdots$. Condition **C2** now holds. If Property 3 does not hold, there exist two crossing double-

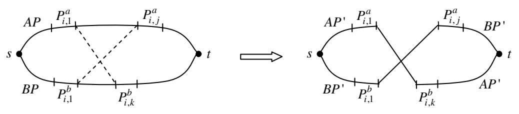

Fig. 4. Transformation to regular-intersected path

intersected paths P_i and P_j as shown in Fig. 4, by a similar transformation as for Property 2, a new feasible solution $\{AP', BP'\}$ can be found as follows. Traverse along AP from s to P^a_{i,k_1}, then along P_i to P^b_{i,k_2}, finally along BP from P^b_{i,k_2} to t. Denote by AP' the resulting path. Traverse along BP from s to P^b_{j,l_2}, then along P_j to P^a_{j,l_1}, finally along AP from P^a_{j,l_1} to t. Denote by BP' the resulting path. P_i and P_j do not cross with each other for the resulting solution $\{AP', BP'\}$.

Repeat the above transformation to the routing paths until all of the three properties in the resulting paths are met. Note that each transformation will reduce the number of subpaths by one at least, so, the procedure terminates after a certain number of transformations. Because each transformation does not increase the cost of the resulting solution, the cost of the final solution is no more than $c(AP, BP)$. □

We thus have the following theorem.

Theorem 2. *There is an optimal solution for MDLDP2 which meets Properties 1,2 and 3.*

4.2 A polynomial Algorithm

We first consider a special case of the problem where there is an optimal solution such that there is no double-intersected paths in $\mathcal{P}$ by presenting a polynomial time algorithm for it. We then propose a polynomial algorithm for the general case by removing the constraint.

No Double-Intersected Path: Assume there is an optimal solution such that there is no double-intersected paths in $\mathcal{P}$. So, there are only single-intersected or non-intersected paths in $\mathcal{P}$. If this optimal solution does not meet Property 1, then there is another optimal solution meeting Property 1, following Theorem 2. Because all the paths in $\mathcal{P}$ that are link-intersected with the optimal solution $\{AP, BP\}$ are single-intersected, the paths that are link-intersected with AP are different from the paths that are link-intersected with BP. Moreover, all the single-intersected paths are trivial-intersected. For a path P in $\mathcal{P}$ that is trivial-intersected with AP, if we traverse AP from the source node to the destination node, we will enter and leave P once and only once, which means that P can be treated like a link for AP. Based on this observation, we propose an algorithm `MDLDP-S`(G,$\mathcal{P}$,s,t), which proceeds as follows. It first constructs an auxiliary graph by compressing each path in $\mathcal{P}$ into a single directed link. Specifically, for each path $P \in \mathcal{P}$, remove all links in P from G. Add two new

```
Algorithm MDLDP-G($G$,$\mathcal{P}$,$s$,$t$)
Input: an undirected graph $G$, a set $\mathcal{P}$ of link-disjoint paths, a pair $(s, t)$.
Output: $\{AP^*, BP^*\}$, a pair of link-disjoint paths in $G$ between $s$ and $t$, such that
    the number of paths in $\mathcal{P}$ that link-intersect with $AP^*$ and $BP^*$ is minimized.
1. Construct a weighted directed auxiliary graph $G' = (V', E')$;
2. Find a shortest path in $G'$ from $s$ to $t$;
3. Construct two link-disjointed paths $\{AP^*, BP^*\}$ in $G$ between $s$ and $t$
   from the above shortest path.
```

Fig. 5. An algorithm for the general case of MDLDP

nodes $entry(P)$ and $exit(P)$. Add a directed link $< entry(P), exit(P) >$ from $entry(P)$ to $exit(P)$ and assign it weight 1. For each node v in P, add a directed link $< v, entry(P) >$ from v to $entry(P)$ and a directed link $< exit(P), v >$ from $exit(P)$ to v, and assign each of them weight 0. For those original undirected links in G that do not appear in any $P \in \mathcal{P}$, add them to the auxiliary graph and assign them weight 0s. Denote by $G' = (V', E')$ the resulting graph.

It then finds two link-disjoint paths $\{AP', BP'\}$ in G' with the minimum cost, using Suurballe's algorithm. An optimal solution $\{AP, BP\}$ in G is finally derived from $\{AP', BP'\}$ by a reverse procedure of the above construction. That is, for each directed link with weight 1 in AP' and BP', replace the link by the corresponding path in $\mathcal{P}$. It is easy to show that $\{AP, BP\}$ is an optimal solution for MDLDP2 in G. Obviously, the proposed algorithm takes $O(m \log n)$ time due to the facts that there are at most m paths in $\mathcal{P}$ and the total number of links in $\mathcal{P}$ is $O(m)$. Thus, there are $O(m)$ nodes and $O(m)$ links in G'. The time complexity of Suurballe's algorithm is $O(m \log n)$.

General case: Recall that Theorem 2 shows that there is an optimal solution $\{AP^*, BP^*\}$ meeting Properties 1, 2 and 3 simultaneously. Assume that $P_{i_1}, P_{i_2}, \cdots, P_{i_k}$ are all the regular-intersected paths in the order from left to right according to their appearances in $\{AP^*, BP^*\}$. For each $1 \leq j \leq k$, let a_j be the right node of the rightmost link P_{i_j} intersects with AP^* and b_j be the right node of the rightmost link P_{i_j} intersects with BP^*. Obviously, $a_1, a_2, \cdots, a_k$ and $b_1, b_2, \cdots, b_k$ are in the order from left to right in $\{AP^*, BP^*\}$. $\{AP^*, BP^*\}$ then can be partitioned into at most $k+1$ pairs of link-disjoint paths by cutting at k pairs of nodes $(a_1, b_1), (a_2, b_2), \cdots, (a_k, b_k)$, and each of these pairs of nodes is referred to as a *partition-pair*. Denote by $\{AP_0^*, BP_0^*\}, \{AP_1^*, BP_1^*\}, \cdots,$ $\{AP_k^*, BP_k^*\}$ the partitions of $\{AP^*, BP^*\}$.

Let $a_0 = b_0 = s$ and $a_{k+1} = b_{k+1} = t$. For $0 \leq j \leq k$, each $\{AP_j^*, BP_j^*\}$ is a pair of link-disjoint paths between a_j, b_j and a_{j+1}, b_{j+1}. For each $\{AP_j^*, BP_j^*\}$, there is at most one regular-intersected path, and the other link-intersected paths are trivial-intersected. So, $\{AP_j^*, BP_j^*\}$ can be found in polynomial time, using Algorithm `MDLDP-S`. The optimal solution $\{AP^*, BP^*\}$ can be obtained by concatenating these $\{AP_j^*, BP_j^*\}$. Motivated by this observation, an algorithm is proposed in Fig. 5. The algorithm first constructs an auxiliary graph $G'(V', E')$, then computes a shortest path from s to t in the auxiliary graph,

```
Construction of $G'(V',E')$
1 $V' = \{s,t\}$ and $E' = \{< s,t >\}$, assign the directed link $< s,t >$ a weight, which is
    equal to the cost of the solution returned by MDLDP-S($G$,$\mathcal{P}$,$s$,$t$);
2 For each pair of nodes $\{u,v\}$, and each path $P \in \mathcal{P}$, if $u,v \in V(P)$, a partition-node
    $X_{u,v,P}$ is added to $V'$;
3 For each partition-node $X_{u,v,P}$ do
3.1 Construct a graph $G'' = (V'',E'')$: $V'' = V \cup \{t''\}$, $E'' = E$, remove all the links
      incident to $u$ or $v$ from $E''$, except those links in $P$. $E'' = E'' \cup \{(u,t''),(v,t'')\}$;
3.2 Add a new link $\{< s, X_{u,v,P} >\}$ into $E'$, and assign it a weight equal to the cost of
      the solution returned by MDLDP-S($G''$,$\mathcal{P} \setminus \{P\}$,$s$,$t''$) plus one;
4 For each partition-node $X_{u,v,P}$ do
4.1 Construct a graph $G'' = (V'',E'')$: $V'' = V \cup \{s''\}$, $E'' = E$, remove all the links
      in $P$ from $E''$. $E'' = E'' \cup \{(s'',u),(s'',v)\}$;
4.2 Add a new link $\{< X_{u,v,P}, t >\}$ into $E'$, and assign it a weight equal to the cost of
      the solution returned by MDLDP-S($G''$,$\mathcal{P} \setminus \{P\}$,$s''$,$t$);
5 For each ordered pair $< X_{u_1,v_1,P_1}, X_{u_2,v_2,P_2} >$ satisfying that $P_1 \neq P_2$, do
5.1 Construct a graph $G'' = (V'',E'')$: $V'' = V \cup \{s'',t''\}$, $E'' = E$, remove all the links
      in $P_1$ and all the links incident to $u_2$ or $v_2$ from $E''$, except those links in $P_2$.
      $E'' = E'' \cup \{(s'',u_1),(s'',v_1),(u_2,t''),(v_2,t'')\}$;
5.2 Add a new link $\{< X_{u_1,v_1,P_1}, X_{u_2,v_2,P_2} >\}$ into $E'$, and assign it a weight equal
      to the cost of the solution returned by MDLDP-S($G''$,$\mathcal{P} \setminus \{P_1,P_2\}$,$s''$,$t''$) plus one;
```

Fig. 6. Construction of the auxiliary graph in Algorithm `MDLDP-G`

finally constructs two link-disjoint paths in G. The construction of G' is shown in Fig. 6. In the construction, when calculate the weight of $< s, X_{u,v,P} >$, we assume that (u,v) is a partition-pair and P is the regular-intersected path. So, we remove all the links incident to u or v except those in P and remove P from $\mathcal{P}$. Other intersected paths except P are trivial-intersected. By calling `MDLDP-S`(G'',$\mathcal{P} \setminus \{P\}$,$s$,$t''$), we can find two link-disjoint paths between s and u, and between s and v with the minimum cost. We assign $< s, X_{u,v,P} >$ a weight that is equal to the returned cost plus one, because P is also intersected. When calculate the weight of $< X_{u,v,P}, t >$, we assume that (u,v) is a partition-pair and P is the regular-intersected path incident to u and v. We remove all the links in P and remove P from $\mathcal{P}$. All the intersected paths are trivial-intersected. By calling `MDLDP-S`(G'',$\mathcal{P} \setminus \{P\}$,$s''$,$t$), we find two link-disjoint paths between u and t, and between v and t. Assign the cost as the weight of $< X_{u,v,P}, t >$. The weight of $< X_{u_1,v_1,P_1}, X_{u_2,v_2,P_2} >$ can be calculated similarly. After the construction of G', we apply Dijkstra's algorithm to find a shortest path in it from s to t. Let $SP'_{s,t}$ be such a shortest path. To construct two link-disjoint paths in G between s and t using $SP'_{s,t}$, we construct a subgraph $G(SP'_{s,t})$ of G by replacing each link in $SP'_{s,t}$ with the corresponding link-disjoint paths. If a path $P \in \mathcal{P}$ is link-intersected by $G(SP'_{s,t})$, we add all the links in P into $G(SP'_{s,t})$. In the end, the number of paths in $\mathcal{P}$ appearing in $G(SP'_{s,t})$ is equal to the weight of $SP'_{s,t}$.

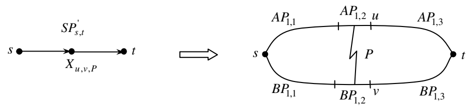

Fig. 7. Construction from $SP'_{s,t}$ to $G(SP'_{s,t})$

Lemma 2. *There are two link-disjoint paths in the subgraph $G(SP'_{s,t})$ between s and t.*

Proof. If $SP'_{s,t}$ consists of only one link $< s,t >$, then there are two link-disjoint paths found at step 1 in the construction of G'. Otherwise, there must be at least one partition-node in $SP'_{s,t}$. In the following we only consider the case where there is only one partition-node, and the case with multiple partition-nodes can be shown by similar arguments. Let $X_{u,v,P}$ be the partition-node in $SP'_{s,t}$. As shown in Fig. 7, replace the link $< s, X_{u,v,P} >$ with two paths $\{AP_{1,1} \cup AP_{1,2}, BP_{1,1} \cup BP_{1,2}\}$. And replace $< X_{u,v,P}, t >$ with the two paths $\{AP_{1,3}, BP_{1,3}\}$. $AP_{1,2}$ and $BP_{1,2}$ are the two rightmost shared subpaths of P. Because P is regular-intersected, $AP_{1,2}$ and $BP_{1,2}$ are connected by a part of P which is not link-intersected with any of $AP_{1,1}, BP_{1,1}, AP_{1,3}$ and $BP_{1,3}$, i.e., $(AP_{1,1} \cup AP_{1,2}) \cap (BP_{1,1} \cup BP_{1,2}) = \emptyset$, $AP_{1,1} \cap AP_{1,2} = \emptyset$, $BP_{1,1} \cap BP_{1,2} = \emptyset$,$(AP_{1,2} \cup AP_{1,3}) \cap (BP_{1,2} \cup BP_{1,3}) = \emptyset$, $AP_{1,2} \cap AP_{1,3} = \emptyset$, $BP_{1,2} \cap BP_{1,3} = \emptyset$. For each link e in $AP_{1,2}$, because $e \notin BP_{1,1} \cup BP_{1,2} \cup BP_{1,3}$, there is a path in $G(SP'_{s,t})$ between s and t which does not include e and e is not an st-bridge in $G(SP'_{s,t})$, where an st-bridge in $G(SP'_{s,t})$ is a link in $G(SP'_{s,t})$ such that every path in $G(SP'_{s,t})$ between s and t passes through this link. Similarly for each link e in $BP_{1,2}$,e is not an st-bridge. For each link e in $AP_{1,1}$, if $e \notin BP_{1,3}$, because $e \notin BP_{1,1} \cup BP_{1,2}$, there is a path between s and t through $BP_{1,1} \cup BP_{1,2} \cup BP_{1,3}$ not through e. Thus, e is not an st-bridge. Otherwise if $e \in BP_{1,3}$, because $AP_{1,3} \cap BP_{1,3} = \emptyset$, $e \notin AP_{1,3}$. There is a path between s and t not via e: traverse from s along $BP_{1,1}$ to $BP_{1,2}$, then traverse from $BP_{1,2}$ along P to $AP_{1,2}$, finally traverse from $AP_{1,2}$ along $AP_{1,3}$ to t. Thus e is not an st-bridge. Similarly, each link in $AP_{1,3} \cup BP_{1,1} \cup BP_{1,3}$ is not an st-bridge either. There is no st-bridge in $G(SP'_{s,t})$. The lemma follows. $\square$

It is clear that the weight of $SP'_{s,t}$ is no more than the cost of the optimal solution for MDLDP2. Algorithm `MDLDP-G` thus is correct. The rest is to analyze the running time of the proposed algorithm. Assume there are k paths in $\mathcal{P}$ and their lengths are $l_1, l_2, \cdots, l_k$ with $l_1 + l_2 + \cdots + l_k \leq m$. There are $l_1^2 + l_2^2 + \cdots + l_k^2 \leq (l_1 + l_2 + \cdots + l_k)^2 \leq m^2$ partition-nodes in G'. The dominant step in the construction of G' is step 5, which takes $O(m^5 \log n)$-time due to that there are $O(m^2)$ nodes and $O(m^4)$ links in G'. Step 2 of algorithm `MDLDP-G` takes $O(m^4 \log n)$-time. The time complexity of algorithm `MDLDP-G` is thus $O(m^5 \log n)$.

Theorem 3. *There is a polynomial algorithm for MDLDP2 which takes $O(m^5 \log n)$ running time, where m and n are the number of links and nodes in the graph.*

Acknowledgment

It is acknowledged that the work by the authors is fully funded by a research grant No:DP0449431 by Australian Research Council under its Discovery Schemes.

References

1. A. Narula-Tam, P. J. Lin, and E. Modiano, "Efficient routing and wavelength assignment for reconfigurable WDM networks," *IEEE Journal on Selected Areas of Communications*, vol. 20, pp. 75–88, 2002.
2. P. Saengudomlert, E. Modiano, and R. Gallager, "On-line routing and wavelength assignment for dynamic traffic in WDM ring and torus networks," in *Proc. IEEE INFOCOM'03*, San Francisco, CA, April 2003, pp. 1805 – 1815.
3. L.-W. Chen and E. Modiano, "Efficient routing and wavelength assignment for reconfigurable WDM networks with wavelength converters," in *Proc. IEEE INFOCOM'03*, San Francisco, CA, April 2003, pp. 1785 – 1794.
4. P. Saengudomlert, E. Modiano, and R. Gallager, "Dynamic wavelength assignment for WDM all optical tree networks," *Allerton Conference on Communications, Control and Computing*, October 2003.
5. K. C. Lee and V. O. K. Li, "A wavelength rerouting algorithm in wide-area all-optical networks," *J. of Lightwave Tech.*, vol. 14, pp. 1218–1229, 1996.
6. G. Mohan and C. Murthy, "A time optimal wavelength rerouting for dynamic traffic in WDM networks," *J. of Lightwave Tech.*, vol. 17, pp. 406–417, 1999.
7. A. Caprara, G. F. Italiano, G. Mohan, A. Panconesi, and A. Srinivasan, "Wavelength rerouting in optical networks, or the venetian routing problem," *Journal of Algorithms*, vol. 45, no. 2, pp. 93–125, 2002.
8. Y. Y. Wan and W. Liang, "Wavelength rerouting for on-line multicast in WDM networks," in *Proc. IEEE LCN'04*, 2004.
9. J. Suurballe, "Disjoint paths in a network," *Networks*, vol. 4, pp. 125–145, 1974.
10. C. Li, S. McCormick, and D. Simchi-Levi, "The complexity of finding two disjoint paths with min-max objective function," *Discrete Applied Mathematics*, vol. 26, no. 1, pp. 105–115, 1990.
11. D. Xu, Y. Chen, Y. Xiong, C. Qiao, and X. He, "On finding disjoint paths in single and dual link cost networks," in *Proc. IEEE INFOCOM'04*, 2004.
12. A. Sen, B. Shen, S. Bandyopadhyay, and J. Capone, "Survivability of lightwave networks - path lengths in WDM protection scheme," *Journal of High Speed Networks*, vol. 10, no. 4, pp. 303–315, 2001.
13. S. Ramamurthy and B. Mukherjee, "Survivable WDM mesh networks part i-protection," in *Proc. IEEE INFOCOM'99*, New York, NY, March 1999, p. 744–751.
14. R. Andersen, F. Chung, A. Sen, and G. Xue, "On disjoint path pairs with wavelength continuity constraint in WDM networks," in *Proc. IEEE INFOCOM'04*, 2004.