Abstract—A fundamental problem in survivable routing in wavelength division multiplexing (WDM) optical networks is the computation of a pair of link-disjoint (or node-disjoint) lightpaths connecting a source with a destination, subject to the wavelength continuity constraint. However, this problem is NP-hard when the underlying network topology is a general mesh network. As a result, heuristic algorithms and integer linear programming (ILP) formulations for solving this problem have been proposed. In this paper, we advocate the use of 2-edge connected (or 2-node connected) subgraphs of minimum isolated failure immune networks as the underlying topology for WDM optical networks. We present a polynomial-time algorithm for computing a pair of link-disjoint lightpaths with shortest total length in such networks. The running time of our algorithm is $O(nW^2)$, where $n$ is the number of nodes, and $W$ is the number of wavelengths per link. Numerical results are presented to demonstrate the effectiveness and scalability of our algorithm. Extension of our algorithm to the node-disjoint case is straightforward.

Index Terms—Disjoint lightpath pairs, minimum isolated failure immune networks, partial 2-trees, wavelength division multiplexing (WDM) optical network.

I. INTRODUCTION

OPTICAL networks implemented using wavelength division multiplexing (WDM) techniques are considered promising candidates for backbone high-speed wide area networks [23], [26]. Among other things, wavelength routing allows logical topologies to be built on top of physical topologies to reflect the traffic intensities between the various nodes as well as to provide reliable services by allowing for reconfiguration in the event of failures [18], [22], [31].

In a WDM optical network, each fiber link can carry many wavelengths. A WDM network is modeled by an undirected graph $G = (V, E, \Lambda)$, where $V$ is the set of vertices, denoting nodes in the network; $E$ is the set of edges, denoting links (or optical fibers) in the network; $\Lambda = \{1, 2, \ldots, W\}$ is the set of wavelengths, and $\Lambda(e) \subseteq \Lambda$ is the set of wavelengths available on link $e$. The terms vertices and nodes are used interchangeably, as well as edges and links. In this model, each undirected edge $[u, v]$ in the network represents a bidirectional link connecting $u$ and $v$. Whenever a link is used by a connection, it is occupied in both directions.

In WDM networks, data packets are transmitted along lightpaths [7]. A lightpath $P(s, t, \lambda)$ between nodes $s \in V$ and $t \in V$ on wavelength $\lambda \in \Lambda$ is an $s$–$t$ path $\pi(s, t)$ in $G$ that uses wavelength $\lambda$ on every link of path $\pi(s, t)$. We assume that the nodes do not have wavelength converters, and as a consequence, each lightpath must maintain the same wavelength throughout the entire path. This is known as the wavelength continuity constraint [3], [36].

Cuts in fibers are one of the most common failures in optical networks, while failures of routers are also possible. Since optical links carry a very high volume of data, survivability is important. Hence, survivable lightpath routing has been extensively studied. In the rest of this section, we give a brief overview of survivable lightpath routing, introduce minimum isolated failure immune networks, state the problem to be studied and our contributions, and differentiate our work from closely related work.

A. Disjoint Lightpath Routing

To tolerate a single link (node, respectively) failure in the network, the path protection scheme of fault management [2], [25] establishes an active lightpath and a link-disjoint (node-disjoint, respectively) backup lightpath, so that in the event of a link failure (node failure, respectively) on the active lightpath, data can be quickly rerouted through the backup lightpath. In the dedicated path protection scheme, each active lightpath has a dedicated backup lightpath—no two backup lightpaths share a common link on a common wavelength. In the shared path protection scheme [24], two backup lightpaths may share a common link on a common wavelength, as long as the corresponding active lightpaths do not share a common link. These are studied under the theme of Shared Risk Link Group (SRLG) [1], [10], [29]. In this paper, we concentrate on dedicated path protection. Also, we concentrate on single failures. For protection schemes against double failures, we refer the readers to [8]. The link-disjoint lightpath routing problem is formally defined in the following.

Definition 1.1 (LDLP): Let a WDM optical network be given by $G = (V, E, \Lambda)$, with node set $V$ and link set $E$, where $\Lambda = \{1, 2, \ldots, W\}$ is the set of wavelengths. For each link $e \in E$,
the set of available wavelengths on link \( e \) is \( \Lambda(e) \subseteq \Lambda \). Let \( s \in V \) be the source node, and \( t \in V \) the destination node. The link-disjoint lightpath routing problem (LDLP) asks for a shortest pair of \( s \rightarrow t \) link-disjoint lightpaths \( \pi_1 \) (on a wavelength \( \lambda_1 \in \Lambda \)) and \( \pi_2 \) (on a wavelength \( \lambda_2 \in \Lambda \)). In other words, we want to find a pair of \( s \rightarrow t \) paths \( \pi_1 \) and \( \pi_2 \) such that we have the following.

1) Paths \( \pi_1 \) and \( \pi_2 \) do not share a common link.
2) Wavelength \( \lambda_1 \) is available on every link on \( \pi_1 \).
3) Wavelength \( \lambda_2 \) is available on every link on \( \pi_2 \).
4) The path pair has the minimum total length (measured by the total number of links used) among all \( s \rightarrow t \) path pairs satisfying the above three conditions.

The LDLP problem deals with dynamic routing, where the connection requests come in a sequential order. Static routing problems aim to satisfy multiple connection requests simultaneously. Dynamic routing algorithms are often used (as heuristics) for solving static routing problems, where we try to establish the connection requests in a sequential manner.

Fig. 1 shows a WDM network and 2 existing connections. The active and backup lightpaths between nodes \( a \) and \( d \) are \((a, h, d)\) on wavelength \( \lambda_1 \) and \((a, f, d)\) on wavelength \( \lambda_2 \), respectively. The active and the backup lightpaths between nodes \( f \) and \( h \) are \((f, g, h)\) on wavelength \( \lambda_2 \) and \((f, d, h)\) on wavelength \( \lambda_1 \), respectively. Note that the active lightpaths are link-disjoint with their corresponding backup lightpaths.

In order to compute a pair of link-disjoint active-backup lightpaths, the current literature uses the shortest active path first (SAPF) heuristic [11], [15], which first computes a shortest lightpath as the active lightpath, then finds a shortest lightpath that is link-disjoint with the active lightpath as the backup lightpath. Computational studies show that the SAPF heuristic is quite effective in practice [3], [11]. However, there is no performance guarantee for the SAPF heuristic. Therefore researchers proposed integer linear programming (ILP) formulations [4], [16], [25]. Although the ILP formulation of the problem can be used to find optimal solutions, solving an ILP takes exponential time in the worst case.

In [3], Andersen et al. proved that the problem of computing a pair of disjoint (either link-disjoint or node-disjoint) lightpaths in a WDM network is NP-hard, asserting a commonly held belief in the WDM networking research community [11]. Yuan and Jue [36] independently proved the hardness of the disjoint lightpath routing problem. Hardness of a related problem was proved by Hu [12]. These hardness results give us a strong belief that there do not exist efficient algorithms for computing a pair of link-disjoint lightpaths in a WDM network with a general mesh underlying topology.

In order to provide network survivability, a price to pay is redundancy. Complete graphs offer the most survivability, but have the highest redundancy. Trees have no redundancy (among connected networks), but offer no survivability. A ring network can survive a single node failure or a single link failure, but cannot survive two link failures.

Farley [13] studied an important class of networks, which can survive many simultaneous node and link failures, as long as they are isolated. Here, two node failures are isolated if the nodes are not adjacent; two link failures are isolated if the links do not meet at a node; a node failure and a link failure are isolated if the failed link is not incident to the failed node.

Definition 1.2 (IFI Networks): A network is said to be isolated-failure-immune (IFI) if all pairs of functional nodes can still communicate, as long as the failures are pairwise isolated. An IFI network is called a minimum IFI network if it has the minimum number of links among all IFI networks on the same number of nodes.

The network in Fig. 2(a) is IFI. When nodes \( D, H, K \), and links \([A, C], [B, J]\) fail simultaneously (note that these network elements are pairwise isolated), the functional nodes are still connected. Simultaneous failures of links \((F, Z)\) and \((I, Z)\) leave node \( Z \) disconnected from the other nodes. However, the links \((F, Z)\) and \((I, Z)\) are not isolated, as they are both incident to node \( Z \). The network in Fig. 2(b) is 2-connected, but not IFI. Simultaneous failures of nodes \( B \) and \( C \) leave nodes \( A \) and \( D \) disconnected from the other functional nodes. Note that nodes \( B \) and \( C \) are isolated.

Farley [13] gave a detailed discussion of both the merits and the usage of IFI networks as survivable networks. In addition, he showed that IFI networks on \( n \) nodes have at least \( 2n - 3 \) links. He further demonstrated that every 2-tree [27] is a minimum IFI network. Wald and Colbourn [33] proved that every minimum IFI network is a 2-tree. Therefore, minimum IFI networks are exactly 2-tree networks.

Minimum IFI networks are closely related to a graph theory technique that has been used by many researchers to design protection schemes for networks. Medard et al. [21] presented a protection scheme using red/blue trees, which has been further studied by Lumetta and Medard [19] and Medard et al. [20] in the design of loop-back recovery schemes. Xue et al. [35] extended the protection scheme of red/blue trees of [21] and discovered a tight connection of the loops used in [20] and [21] with the ear decomposition in graph theory [34]. Zhang et al. [37] further explored this connection and designed faster algorithms to construct red/blue trees. Other research works along this line can be found in [14] and [32]. In reality,
a 2-tree has a special ear decomposition—the first ear is a triangle, and every other ear is a 2-hop path that makes a triangle with an edge of earlier ears.

We believe that minimum IFI networks (and 2-edge connected or 2-node connected subgraphs of minimum IFI networks) are an excellent candidate for the backbone of future survivable optical networks. There are many reasons leading us to this belief. First, minimum IFI networks are fault-tolerant in the sense that they are immune to multiple simultaneous isolated failures. Second, minimum IFI networks are sparse—for a network with $n$ nodes, the number of links is $2n - 3$, a fact which enhances scalability. Third, there is a close relationship between minimum IFI networks and the loop-back recovery schemes studied in [14], [19]–[21], [32], [35], and [37] and the logical topology mapping scheme of Kurant and Thiran [17]. Fourth, minimum IFI networks have a nice structure that makes routing decisions much easier than in general mesh networks [9], [33].

C. Main Contributions

We advocate the use of minimum IFI networks (or their 2-edge/2-node connected subgraphs) as the underlying topology for WDM optical networks. We present a polynomial-time algorithm for solving the LDLP problem when the underlying topology is a subgraph of a minimum IFI network. In particular, our algorithm has a worst-case time complexity of $O(nW^2)$, where $n$ is the number of nodes in the network, and $W$ is the number of wavelengths per link. Our algorithm can be easily generalized to the node-disjoint case. Given the hardness result discussed in Section I-A, our theoretical contribution is significant. In addition, extensive numerical results show that our algorithm has a noticeable advantage over the shortest active path first heuristic. Given the fault-tolerant properties of minimum IFI networks, our work will have an impact on the future design of survivable networks.

D. Other Related Work

To make our review of the literature as complete as possible, we present a review of works that are related to the problem of finding link/node disjoint paths in a network. Here, the paths are not required to be lightpaths.

Given a pair of vertices, finding edge/vertex-disjoint paths (not lightpaths) between these vertices is a fundamental problem in network survivability. Suurballe [30] presented efficient algorithms for computing disjoint paths for a given source–destination pair with minimum total length. Zheng et al. [39] studied the computation of multiple paths that have minimum link and node sharing. Banner and Orda [5] and She et al. [28] studied multipath routing with minimum sharing. The aforementioned works all deal with paths (rather than lightpaths). Therefore, our work is different from these works.

The rest of this paper is organized as follows. In Section II, we present important properties of minimum IFI networks. In Section III, we present a linear-time algorithm for computing a pair of link-disjoint source-to-destination lightpaths, when the underlying network topology is a subgraph of a minimum IFI network and the wavelengths for the two paths are fixed. In Section IV, we present a polynomial-time algorithm for computing a pair of link-disjoint source-to-destination lightpaths, when the underlying network topology is a subgraph of a minimum IFI network. In Section V, we present numerical results. We conclude the paper in Section VI.

II. Minimum IFI Networks and 2-Tree Networks

In this section, we present some important properties of minimum IFI networks that will be used in our algorithm design. Since minimum IFI networks are equivalent to 2-tree networks [33], we will use these two terms interchangeably.

A. 2-Tree Networks

We use $[u, v]$ to denote an undirected link connecting nodes $u$ and $v$. Since the link is undirected, $[u, v]$ and $[v, u]$ both denote the same link. We use $\triangle uvw$ to denote the triangle with nodes $u$, $v$, and $w$ and links $[u, v]$, $[v, w]$, and $[w, u]$. The nodes in a triangle are not ordered. Therefore, $\triangle uvw$, $\triangle wvu$, and $\triangle uvw$ all denote the same triangle.

Definition 2.1 (2-Tree Networks): A 2-tree is defined recursively as follows, and all 2-trees may be obtained in this way. A triangle is a 2-tree. Given a 2-tree and a link $[x, y]$ of the 2-tree, we can add a new node $z$ adjacent to both $x$ and $y$ (to expand the current 2-tree); the result is a 2-tree.

Fig. 2(a) shows a 2-tree network. Initially, the triangle $\triangle ABC$ is a 2-tree with node set $\{A, B, C\}$ and link set $\{[A, B], [A, C], [B, C]\}$. Next, we add a new node $D$ adjacent to link $[A, C]$ in the current 2-tree, leading to the 2-tree with node set $\{A, B, C, D\}$ and link set $\{[A, B], [A, C], [B, C], [A, D], [C, D]\}$. To describe the above process, we say the 2-tree is expanded by triangle $\triangle ACD$ or node $D$ expands out of link $[A, C]$. We can continue this process to expand the 2-tree constructed so far to obtain the 2-tree shown in Fig. 2(a).

One way to obtain the 2-tree shown in Fig. 2(a) from the 2-tree on node set $\{A, B, C, D\}$ is to execute the following sequence of expansion operations:

1. expand by $\triangle FBC$;
2. expand by $\triangle HCF$;
3. expand by $\triangle IBF$;
4. expand by $\triangle JBI$;
5. expand by $\triangle KIJ$;
6. expand by $\triangle XIF$;
7. expand by $\triangle ZIF$.

Another way to obtain the 2-tree shown in Fig. 2(a) from the 2-tree on node set $\{A, B, C, D\}$ is to execute the following sequence of operations:

1. expand by $\triangle FBC$;
2. expand by $\triangle IBF$;
3. expand by $\triangle ZIF$;
4. expand by $\triangle XIF$;
5. expand by $\triangle JBI$;
6. expand by $\triangle HCF$;
7. expand by $\triangle KIJ$.

Let $\triangle xyz$ be a triangle. We say link $[x, y]$, $[x, z]$, as well as $[y, z]$ is incident to triangle $\triangle xyz$. Let $[x, y]$ be a link in a 2-tree. The degree of link $[x, y]$ is the number of triangles in the 2-tree that are incident to link $[x, y]$. For example, the degree of link $[A, B]$ in Fig. 2(a) is 1, while the degree of link $[I, F]$ in Fig. 2(a) is 3.
A reduced 2-tree with respect to $s$ and $t$ gives a nice picture about the $s$-$t$ paths, as will be shown in Lemma 2.2. A natural question is whether a reduced 2-tree exists and how to compute it. In the following, we present a simple algorithm to compute a reduced 2-tree with respect to a pair of nodes.

Algorithm 1: Reduced2Tree

**Input:** A 2-tree $T$, and two nodes $s$ and $t$ in $T$.

**Output:** A reduced 2-tree $T'$ with respect to $s$ and $t$.

1. $T' \leftarrow T$;
2. while $T'$ is not a triangle and $\exists$ a degree-2 node $z$ other than $s$ and $t$ in $T'$ do
   3. delete $z$ from $T'$;
4. end while
5. output $T'$;

**Theorem 2.1:** Let $s$ and $t$ be two nodes in a 2-tree $T$ with $n$ nodes. Algorithm 1 computes a reduced 2-tree $T'$ with respect to $s$ and $t$ in $O(n)$ time. If no triangle in $T$ contains both $s$ and $t$, the reduced 2-tree with respect to $s$ and $t$ is unique.

**Proof:** We use a queue to store the nodes (other than $s$ and $t$) in $T$ whose degree in $T'$ is 1. With such an implementation, checking the condition in line 2 takes $O(1)$ time. When deleting node $z$ in line 3, we update the degrees of the two nodes of $T$ which form a triangle with $z$. If anyone of them has degree 1 (and is different from $s$ and $t$), insert it into the queue. Hence, the algorithm takes $O(n)$ total time.

The resulting subgraph $T'$ is a 2-tree since the deletion operation is the inverse of an expansion operation (Definition 2.1). In the next two paragraphs, we will prove that the resulting 2-tree $T'$ is indeed a reduced 2-tree with respect to $s$ and $t$.

We first prove that the deletion in line 3 of the algorithm does not change the set of separators of $s$ and $t$. Assume that $[u, v]$ is a separator of $s$ and $t$ before the deletion. Since $[u, v]$ is a separator of $s$ and $t$ before the deletion, the set $[u, v]$ is a separator of $s$ and $t$ after the deletion of node $z$. Clearly, if $[u, v]$ is not a separator of $s$ and $t$ before the deletion, it cannot become a separator of $s$ and $t$ after the deletion, even if it remains a link of $T'$ after the deletion.

We next prove that the resulting 2-tree $T'$ at the end of the algorithm does not contain any 2-separator that is not a separator of $s$ and $t$. To the contrary, assume that $[u, v]$ is not a separator of $s$ and $t$, but is a 2-separator of the resulting 2-tree at the end of the algorithm. Since $[u, v]$ is a 2-separator of the 2-tree, the 2-tree can be decomposed into two or more parts that only intersect in $[u, v]$. Since $[u, v]$ is not a separator of $s$ and $t$, both $s$ and $t$ belong to a single component. Hence, there is a component that contains neither $s$ nor $t$ (with $u$ and $v$ excluded). Since this component is a 2-tree, it must contain a degree-2 node other than $u$ and $v$. Therefore, the algorithm should not have stopped. This contradiction proves our claim. Hence, the 2-tree $T'$ at the end of the algorithm is a reduced 2-tree with respect to $s$ and $t$.

Finally, we prove the uniqueness of the reduced 2-tree with respect to $s$ and $t$, when no triangle contains both $s$ and $t$. Since no triangle in $T$ contains both $s$ and $t$, the resulting 2-tree $T'$ at the end of the algorithm cannot be a triangle. Hence, the set of separators of $s$ and $t$ is nonempty. As a result, the reduced

![Fig. 3. The 2-tree in Fig. 2(a) can be partitioned into three components that pairwise intersect in $[I, F]$; (a) the component that contains node $A$; (b) the component that contains node $X$; (c) the component that contains node $Z$.](image-url)
2-tree $T$ computed by Algorithm 1 has $s$ and $t$ as the only degree-2 nodes, and has the set of separators of $s$ and $t$ as its only 2-separators.

Let $T'$ be any reduced 2-tree with respect to $s$ and $t$. Then, $T'$ cannot have a degree-2 node other than $s$ and $t$. To the contrary, assume that $z$ (other than $s$ and $t$) is a degree-2 node, which is contained in a triangle $\Delta xyt$. Then, link $[u, v]$ is a 2-separator of $T'$, but is not a separator of $s$ and $t$. This is a contradiction. Therefore, the only degree-2 nodes of $T'$ are $s$ and $t$. Since $T$ and $T'$ both have $s$ and $t$ as their only degree-2 nodes, and both have the same set of 2-separators, they must be identical. This proves the uniqueness of the reduced 2-tree with respect to $s$ and $t$.

**Example 2:** Given the 2-tree in Fig. 2(a), we can compute the reduced 2-tree with respect to $A$ and $Z$ using Algorithm 1. Initially, the degree-2 nodes other than $A$ and $B$ are $D$, $H$, $K$, and $X$. Following the algorithm, node $D$ is deleted, followed by $H$ and $K$. When $K$ is deleted, $J$ becomes a degree-2 node. Thus, we continue to delete nodes $X$ and $J$. Now the only degree-2 nodes are $A$ and $Z$. Hence, the algorithm stops, producing the reduced 2-tree shown in Fig. 4(a).

Since no triangle in Fig. 2(a) contains both $A$ and $Z$, the reduced 2-tree with respect to $A$ and $Z$ is unique, in which $A$ and $Z$ are the only degree-2 nodes. If there is a triangle containing both $s$ and $t$, the reduced 2-tree with respect to $s$ and $t$ will be a triangle [see Fig. 4(b)], which might not be unique. For the 2-tree in Fig. 2(a), both $\Delta ACB$ and $\Delta ACD$ are reduced 2-trees with respect to $A$ and $C$.

**Remark 1:** When the set of separators of $s$ and $t$ is nonempty, the reduced 2-tree $T'$ with respect to $s$ and $t$ is unique, which will be denoted by $T(s, t)$. In addition, there is a unique sequence of $K$ deletions of the degree-2 node other than $t$ to convert $T'$ to a triangle. A linear ordering of the $K$ separators of $s$ and $t$ is formed in the order they change from a 2-separator to a peripheral during the above deletion process. For the reduced 2-tree $T(A, Z)$ in Fig. 4(a), we can delete node $A$ to get $T(C, Z)$, then delete $C$ to get $T(B, Z)$, then delete $B$ to get the triangle $\Delta ZFI$. Therefore, the three separators of $A$ and $Z$ are a linear ordering of $[B, C, [B, F, [F, I, I]]]$. Hence, $[B, C]$ is a 2-separator of $[B, F, I]$ and $[F, I]$ is a 2-separator of $[B, F]$. We will use this ordering and the concepts of predecessor and successor in the design of our algorithm in Section III.

A reduced 2-tree $T(s, t)$ gives a nice picture about the $s$–$t$ paths, as shown in the following lemma. It is this property that makes the LDLP problem tractable when the underlying network topology is a subgraph of a minimum IFI network.

**Lemma 2.2:** Let $G$ be a 2-tree, and $T(s, t)$ be the reduced 2-tree with respect to $s$ and $t$. Let $\{P, Q\}$ be any one of the 2-separators in $T(s, t)$, and let $\pi_1$ and $\pi_2$ be two link-disjoint paths in $T$ connecting $s$ and $t$. Then, at least one of the following four statements is true.

1) Both $\pi_1$ and $\pi_2$ go through node $P$.
2) $\pi_1$ goes through node $P$, and $\pi_2$ goes through node $Q$.
3) $\pi_1$ goes through node $Q$, and $\pi_2$ goes through node $P$.
4) Both $\pi_1$ and $\pi_2$ go through node $Q$.

**Proof:** By Lemma 2.1, both $\pi_1$ and $\pi_2$ have to go through a node in $\{P, Q\}$. Therefore, at least one of the four combinations in this lemma must be true.

**III. COMPUTING A PAIR OF LINK-DISJOINT LIGHTPATHS IN A MINIMUM IFI NETWORK, WITH GIVEN WAVELENGTHS**

Our goal is to design a polynomial-time algorithm for computing a shortest pair of link-disjoint $s$–$t$ lightpaths in a WDM optical network whose underlying topology is a partial 2-tree, i.e., a subgraph of a minimum IFI network. For easier understanding of the overall algorithm, we present the core of our algorithm in this section, concentrating on a restricted version of the problem where the underlying topology is a 2-tree, and the wavelengths of the two paths are chosen/fixed.

Before formally defining this restricted problem, we use an example to illustrate link-disjoint lightpath routing. Fig. 5 shows a WDM network whose underlying topology is a 2-tree with 11 nodes, and $\Delta = \{1, 2\}$. The attributes of the links are described in the caption of the figure. We will also use the WDM network in Fig. 5 as our running example while illustrating major steps of our algorithm.

**Example 3:** The shortest $A$–$Z$ lightpath on wavelength 1 (or 2) is $\pi_a = (A, H, F, Z)$. However, there does not exist another $A$–$Z$ lightpath that does not share a common link with $\pi_a$. One can verify that $\pi_1 = (A, B, J, K, I, Z)$ on wavelength 1 and $\pi_2 = (A, D, C, H, F, Z)$ on wavelength 2 form a pair of link-
disjoint lightpaths, with a total length of 10. Hence, computing the shortest active lightpath first may fail to find a pair of disjoint lightpaths when such a pair exists.

In this section, we study LDLP-CW, a restricted version of the LDLP problem with the wavelengths on π1 and π2 predetermined, and the underlying topology being a 2-tree.

Definition 3.1 (LDLP-CW): Let a WDM optical network be given by \( G = (V, K, A) \), where \( 1 = (V, K) \) is a 2-tree, and \( A = \{1, 2, \ldots, W\} \) is the set of wavelengths. For each link \( e \in E \), the set of available wavelengths on link \( e \in \mathcal{A}[e] \subseteq A \). Let \( s \in V \) be the source node, and \( t \in V \) the destination node. Let \( \lambda_1, \lambda_2 \in \Lambda \) be given. The link-disjoint lightpath routing problem with chosen wavelengths (LDLP-CW) asks for a shortest pair of \( s-t \) link-disjoint lightpaths \( \pi_1 \) and \( \pi_2 \) on wavelength \( \lambda_1 \) and \( \lambda_2 \) on wavelength \( \lambda_2 \). In other words, we want to find a pair of \( s-t \) paths \( \pi_1 \) and \( \pi_2 \) such that we have the following.

1) Paths \( \pi_1 \) and \( \pi_2 \) do not share a common link.
2) Wavelength \( \lambda_1 \) is available on every link on \( \pi_1 \).
3) Wavelength \( \lambda_2 \) is available on every link on \( \pi_2 \).
4) The path pair has the minimum total length (measured by the total number of links used) among all \( s-t \) path pairs satisfying the above three conditions.

The development of our algorithm for solving LDLP-CW is organized in the remainder of this section. In Section III-A, we describe the link attributes, where each link is associated with a constant number of attributes. In Section III-B, we describe the contraction operation. We also show the \( O(1) \) time update of the link attributes associated with each contraction operation. In Section III-C, we describe how to apply the contraction operations to obtain a reduced 2-tree with respect to \( s \) and \( t \). In Section III-D, we describe how to apply the contraction operations to convert the reduced 2-tree to a triangle. We also show how to find an optimal solution to LDLP-CW at the triangle. In Section III-E, we assemble these building blocks together to present our \( O(n) \)-time algorithm for LDLP-CW.

### A. Link Attributes

Assume that wavelengths \( \lambda_1, \lambda_2 \in \Lambda \) are chosen for the LDLP-CW problem. For each undirected link \( [u, v] \), there are two ordered node pairs \( [u, v] \), \( [v, u] \), one in each direction. For each ordered pair \( [u, v] \), we associate three attributes: \( \alpha_1(u, v), \alpha_2(u, v), \) and \( \beta(u, v) \). The meaning and value of these attributes are explained in the following. We will use \( G(u, v) \) to denote the subgraph currently represented by link \( [u, v] \). Before performing any contraction operations, \( G(u, v) \) consists of the nodes \( u \) and \( v \), and the link \( [u, v] \in E \). When we perform a contraction operation, \( G(u, v) \) will expand to include the subgraph that has been contracted to it. We will explain this in detail in Section III-B. We will use \( \pi = \phi \) to denote that lightpath \( \pi \) does not exist. In this case, we say the lightpath is empty. We use \( l(\pi) \) to denote the length of lightpath \( \pi \). When \( \pi \neq \phi \), \( l(\pi) \) is the number of hops in \( \pi \). When \( \pi = \phi \), we have \( l(\pi) = l(\phi) = \infty \). The meaning and value of the link attributes are explained in Table I.

<table>
<thead>
<tr>
<th>attribute</th>
<th>meaning and value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha_1(u, v) )</td>
<td>contains a shortest ( u-v ) lightpath ( p_1(u, v) ) on wavelength ( \lambda_1 ) using only links in ( G(u, v) ). If there is no ( u-v ) lightpath on ( \lambda_1 ) in ( G(u, v) ), we set ( p_1(u, v) = \phi ). The length of ( \alpha_1(u, v) ) is defined as ( l(\alpha_1(u, v)) = l(p_1(u, v)) ).</td>
</tr>
<tr>
<td>( \alpha_2(u, v) )</td>
<td>contains a shortest ( u-v ) lightpath ( p_2(u, v) ) on wavelength ( \lambda_2 ) using only links in ( G(u, v) ). If there is no ( u-v ) lightpath on ( \lambda_2 ) in ( G(u, v) ), we set ( p_2(u, v) = \phi ). The length of ( \alpha_2(u, v) ) is defined as ( l(\alpha_2(u, v)) = l(p_2(u, v)) ).</td>
</tr>
<tr>
<td>( \beta(u, v) )</td>
<td>contains an ordered pair ( (q_1(u, v), q_2(u, v)) ) with the shortest total length, where ( q_1(u, v) ) is an ( u-v ) lightpath on ( \lambda_1 ) in ( G(u, v) ), and ( q_2(u, v) ) is an ( u-v ) lightpath on ( \lambda_2 ) in ( G(u, v) ) such that the two lightpaths are link-disjoint. If such a pair of link-disjoint lightpaths does not exist, we set ( q_1(u, v) = \phi ) and ( q_2(u, v) = \phi ). The length of ( \beta(u, v) ) is defined as ( l(\beta(u, v)) = l(q_1(u, v)) + l(q_2(u, v)) ).</td>
</tr>
</tbody>
</table>

Example 4: We use the network in Fig. 5 to illustrate the initial link attribute values, assuming \( \lambda_1 = 1 \) and \( \lambda_2 = 2 \). Since for each link \( [u, v] \), the current value of \( G(u, v) \) is the subgraph with nodes \( u \) and \( v \) and link \( [u, v] \), there does not exist a pair of link-disjoint \( u-v \) lightpaths in \( G(u, v) \). Therefore, for each of the 19 links \( [u, v] \) in Fig. 5, we have \( q_1(u, v) = q_2(u, v) = \phi \), i.e., \( \beta(u, v) = (\phi, \phi) \).

### B. Contraction Operations

We have seen in Section II-A that a 2-tree is either a triangle, or obtained by expanding a degree-2 node out of a link in an existing 2-tree. The reverse operation of expansion is the contraction of a degree-2 node \( z \) onto the link \( x, y \), with which it forms a triangle. The link attributes are updated in \( O(1) \) time according to defined rules when a contraction operation is performed. The contraction operations are used in our algorithm to compute a reduced 2-tree with respect to \( s \) and \( t \), while updating the link attributes that will be used to compute the optimal solution. We make use of the concatenation operator and the optimization operator, defined in the following.

We use \( \circ \) to denote the concatenation operator. If \( p_1(x, z) \) and \( p_2(z, y) \) are two lightpaths on wavelength \( \lambda_1 \), which share no common node except \( z \), the concatenation of \( p_1(x, z) \) and
Fig. 6. If \( z \) is a degree-2 node that makes a triangle with link \([x, y]\), we can contract node \( z \) to link \([x, y]\). After the contraction, \( G(x, y) \) is expanded to \( G^{\text{new}}(x, y) \), which is the union of \( G(x, y) \), \( G(z, x) \), and \( G(z, y) \).

\( p_{1}(z, y) \) is a lightpath from \( x \) to \( y \) on wavelength \( \lambda_{1} \). We use \( p_{1}(z, y) \) to denote this concatenation, whose length is \( l(p_{1}(z, y)) = l(p_{1}(z, x)) + l(p_{1}(x, y)) \). Clearly, \( p_{1}(z, y) \) is expanded to \( p_{1}(x, y) \), the union of \( p_{1}(z, x) \) and \( p_{1}(x, y) \).

The second operator is the optimization operator, denoted by \( \text{opt} \). This operator can be applied to either two lightpaths connecting nodes \( x \) and \( y \) or four pairs of lightpaths connecting nodes \( x \) and \( y \). If \( p_{1}(x, y) \) and \( p_{2}(x, y) \) are \( x-y \) lightpaths on \( \lambda_{1} \), then \( \text{opt}(p_{1}(x, y), p_{2}(x, y)) \) is a \( p_{1}(x, y) \) if \( l(p_{1}(x, y)) \leq l(p_{2}(x, y)) \), and \( p_{2}(x, y) \) otherwise.

Lemma 3.1: Assume that \( z \) is a degree-2 node, and \( \cap x y z \) is a triangle. Then, we can contract node \( z \) onto link \([x, y]\) as shown in Fig. 6. The attributes associated with \([x, y]\) are updated by performing (3.1)–(3.5) in the given order.

\[
\begin{align*}
\text{opt}(p_{1}(x, z), p_{1}(z, y)) &= \min_{k \leq 4} l(p_{k}(x, y)), \quad \text{if } p_{1}(x, y) \text{ is shorter} \\
\text{opt}(p_{2}(x, z), p_{2}(z, y)) &= \min_{k \leq 4} l(p_{k}(x, y)), \quad \text{if } p_{2}(x, y) \text{ is shorter} \\
\text{opt}(p_{1}(x, y), p_{2}(x, y)) &= \min_{k \leq 4} l(p_{k}(x, y)), \quad \text{if } p_{1}(x, y) \text{ is shorter} \\
\text{opt}(p_{2}(x, y), p_{2}(x, y)) &= \min_{k \leq 4} l(p_{k}(x, y)), \quad \text{if } p_{2}(x, y) \text{ is shorter} \\
\end{align*}
\]

\( p_{1}(x, y) \) is a lightpath from \( x \) to \( y \) on wavelength \( \lambda_{1} \). We use \( p_{1}(x, y) \) to denote this concatenation, whose length is \( l(p_{1}(x, y)) = l(p_{1}(x, z)) + l(p_{1}(z, y)) \). Clearly, \( p_{1}(x, y) \) is expanded to \( p_{1}(x, z) \), the union of \( p_{1}(x, z) \) and \( p_{1}(z, y) \). If \( p_{1}(x, z) \) or \( p_{1}(z, y) \).

\( \text{opt}(p_{1}(x, y), p_{2}(x, y)) = \min_{k \leq 4} l(p_{k}(x, y)), \quad \text{if } p_{1}(x, y) \text{ is shorter} \\
\text{opt}(p_{2}(x, y), p_{2}(x, y)) = \min_{k \leq 4} l(p_{k}(x, y)), \quad \text{if } p_{2}(x, y) \text{ is shorter} \\
\text{opt}(p_{1}(x, y), p_{2}(x, y)) = \min_{k \leq 4} l(p_{k}(x, y)), \quad \text{if } p_{1}(x, y) \text{ is shorter} \\
\text{opt}(p_{2}(x, y), p_{2}(x, y)) = \min_{k \leq 4} l(p_{k}(x, y)), \quad \text{if } p_{2}(x, y) \text{ is shorter} \\
\end{align*}
\]

\( q_{1}(x, y) \) is a shortest pair of link-disjoint lightpaths on \( \lambda_{1} \) in \( G(x, z) \), and \( G(z, y) \). Then, \( q_{1}(x, y) \) and \( q_{2}(x, y) \) are link-disjoint.

C. From 2-Tree \( T \) to Reduced 2-Tree \( T(s, t) \)

We can use the contraction operation described in Section III-B to obtain \( T(s, t) \) from \( T \), one contraction at a time, and update the link attributes in \( G(1) \) time per contraction. The order of contraction follows Algorithm 1. The update of link attributes follows the rules stated in Lemma 3.1.
Table II

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Meaning and Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma_1(P, Q)$</td>
<td>When nonempty, $\gamma_1(P, Q) = (q_{11}^{PQ}, q_{12}^{PQ})$, where $q_{11}^{PQ}$ is an $s$-$P$ lightpath in $a(s, P)$ on wavelength $\lambda_1$, $q_{12}^{PQ}$ is an $s$-$P$ lightpath in $a(s, Q)$ on wavelength $\lambda_2$, and the two lightpaths are link-disjoint. In addition, $l(q_{11}^{PQ}) + l(q_{12}^{PQ})$ is minimized among all such lightpath pairs.</td>
</tr>
<tr>
<td>$\gamma_2(P, Q)$</td>
<td>When nonempty, $\gamma_2(P, Q) = (q_{21}^{PQ}, q_{22}^{PQ})$, where $q_{21}^{PQ}$ is an $s$-$P$ lightpath in $a(s, P)$ on wavelength $\lambda_1$, $q_{22}^{PQ}$ is an $s$-$Q$ lightpath in $a(s, Q)$ on wavelength $\lambda_2$, and the two lightpaths are link-disjoint. In addition, $l(q_{21}^{PQ}) + l(q_{22}^{PQ})$ is minimized among all such lightpath pairs.</td>
</tr>
<tr>
<td>$\gamma_3(P, Q)$</td>
<td>When nonempty, $\gamma_3(P, Q) = (q_{31}^{PQ}, q_{32}^{PQ})$, where $q_{31}^{PQ}$ is an $s$-$Q$ lightpath in $a(s, P)$ on wavelength $\lambda_1$, $q_{32}^{PQ}$ is an $s$-$P$ lightpath in $a(s, Q)$ on wavelength $\lambda_2$, and the two lightpaths in link-disjoint. In addition, $l(q_{31}^{PQ}) + l(q_{32}^{PQ})$ is minimized among all such lightpath pairs.</td>
</tr>
<tr>
<td>$\gamma_4(P, Q)$</td>
<td>When nonempty, $\gamma_4(P, Q) = (q_{41}^{PQ}, q_{42}^{PQ})$, where $q_{41}^{PQ}$ is an $s$-$Q$ lightpath in $a(s, P)$ on wavelength $\lambda_1$, $q_{42}^{PQ}$ is an $s$-$Q$ lightpath in $a(s, Q)$ on wavelength $\lambda_2$, and the two lightpaths are link-disjoint. In addition, $l(q_{41}^{PQ}) + l(q_{42}^{PQ})$ is minimized among all such lightpath pairs.</td>
</tr>
</tbody>
</table>

For a given $i \in \{1, 2, 3, 4\}$, $\gamma_i(P, Q)$ is either nonempty with the following meaning, or empty (denoted by $(\phi, \phi)$) when the corresponding lightpath pair does not exist.

The computation of the attributes $\gamma_i$ starts with the link $[x, y]$ in the reduced 2-tree $T(s, t)$ that makes a triangle with the source node $s$. Lemma 3.2 shows how to compute these attributes for the link $[x, y]$ in $T(s, t)$ that makes a triangle with the source node $s$. Lemmas 3.3, 3.4, 3.5, and 3.6 show the rules for computing $\gamma_1(P, R), \gamma_2(P, R), \gamma_3(P, R),$ and $\gamma_4(P, R)$, respectively, when $\gamma_i(P, Q)$ are known, where $[P, Q]$ is the predecessor of $[P, R]$ in $T(s, t)$.

**Lemma 3.2:** Let $[x, y]$ be the link in $T(s, t)$ that makes a triangle with the source node $s$, where $x \neq t$. Then, $\gamma_i(x, y)$ can be computed as follows:

\[
\gamma_1(x, y) = \beta(s, x) \quad (3.6)
\]
\[
\gamma_2(x, y) = (p_1(s, y), p_2(s, y)) \quad (3.7)
\]
\[
\gamma_3(x, y) = (p_1(s, y), p_2(s, x)) \quad (3.8)
\]
\[
\gamma_4(x, y) = \beta(s, y) \quad (3.9)
\]

where $p_i(\cdot)$, and $\beta(\cdot)$ are defined in Section III-A and computed according the rules in Section III-B.

**Proof:** In this case, $s[x, y]$ contains the nodes $s, x, y$. When nonempty, $\beta(s, x) = (q_1(s, x), q_2(s, x))$, where $q_1(s, x)$ is an $s$-$x$ lightpath on $\lambda_1$ in $G(s, x)$, and $q_2(s, x)$ is an $s$-$x$ lightpath on $\lambda_2$ in $G(s, x)$ such that $q_1(s, x)$ and $q_2(s, x)$ are link-disjoint and the two form a shortest link-disjoint lightpath pair. Therefore, we have (3.6). When nonempty, $p_1(s, x)$ is a shortest $s$-$x$ lightpath on $\lambda_1$ in $G(s, x)$, $p_2(s, y)$ is a shortest $s$-$y$ lightpath on $\lambda_2$ in $G(s, y)$. The two lightpaths are guaranteed to be link-disjoint. Hence, we have (3.7). Equation (3.8) is symmetric to (3.7). Equation (3.9) is symmetric to (3.6). Hence, they can be proved similarly to (3.7) and (3.6).

Having computed the $\gamma_i(P, Q)$ values for 2-separator $[P, Q]$, we can compute the $\gamma_i(P, R)$ values for its succeeding 2-separator $[P, R]$ (if $[P, Q]$ is the 2-separator which makes a triangle with the destination node, we can compute the values $\gamma_i(P, t)$). The two cases are illustrated in Fig. 7(a) and (b) and Fig. 7(c) and (d), respectively.

The rules for computing $\gamma_i(P, R)$ from the values of $\gamma_i(P, Q)$ are given in the next four lemmas. Their proofs are straightforward (illustrated in Figs. 8–11). We provide the proof for Lemma 3.3 and omit those for Lemmas 3.4–3.6 because they follow similar logic.

**Lemma 3.3:** Let $[P, Q]$ be a 2-separator in the reduced 2-tree $T(s, t)$. Let $[P, R]$ be a successor of $[P, Q]$, as shown in Fig. 7(a), or $[P, R]$ is a peripheral where $R$ coincides with $t$, as shown in Fig. 7(c). Then, $\gamma_1(P, R) = (q_{11}^{PR}, q_{12}^{PR})$ is
a shortest lightpath pair among the following four (possibly empty) lightpath pairs, as illustrated in Fig. 8:

\( c-1.1: (q_{11}^{PR}, q_{12}^{PR}) \)
\( c-1.2: (q_{21}^{PQ}, q_{22}^{PQ}) \)
\( c-1.3: (q_{31}^{PQ} \circ p_1(Q, P), q_{32}^{PQ} \circ p_2(Q, P)) \)
\( c-1.4: (q_{41}^{PQ} \circ q_1(Q, P), q_{42}^{PQ} \circ q_2(Q, P)) \).

**Proof:** By definition, \( \gamma_1(P, R) = (q_{11}^{PR}, q_{12}^{PR}) \) is the shortest link-disjoint lightpath pair in \( s(P, R) \), with \( q_{11}^{PR} \) an \( s-P \) lightpath on \( \lambda_1 \), and \( q_{12}^{PR} \) an \( s-R \) lightpath on \( \lambda_2 \). There are exactly four possible cases, as shown in Fig. 8(a)–(d).

In \( c-1.1 \), neither \( q_{11}^{PR} \) nor \( q_{12}^{PR} \) uses any link in \( G(P, Q) \), as shown in Fig. 8(a). Hence, \( (q_{11}^{PR}, q_{12}^{PR}) = (q_{11}, q_{12}) \) since no links in \( G(P, Q) \) are used by either lightpath.

In \( c-1.2 \), \( q_{11}^{PR} \) (on \( \lambda_1 \)) does not use any link in \( G(P, Q) \), but \( q_{12}^{PR} \) (on \( \lambda_2 \)) goes from \( Q \) to \( P \) through \( G(P, Q) \), as shown in Fig. 8(b). Hence, \( (q_{11}^{PR}, q_{12}^{PR}) = (q_{21}^{PQ} \circ p_1(Q, P), q_{22}^{PQ} \circ p_2(Q, P)) \). Recall that \( (q_{21}^{PQ}, q_{22}^{PQ}) \) is a shortest link-disjoint lightpath pair in \( s(P, Q) \) such that \( q_{21}^{PQ} \) is an \( s-P \) lightpath on \( \lambda_1 \), and \( q_{22}^{PQ} \) is an \( s-Q \) lightpath on \( \lambda_2 \), and that \( p_2(Q, P) \) is a shortest \( Q-P \) lightpath on \( \lambda_2 \) in \( G(Q, P) \).

In \( c-1.3 \) (symmetric to \( c-1.2 \)), \( q_{11}^{PR} \) (on \( \lambda_1 \)) goes from \( Q \) to \( P \) through \( G(Q, P) \), as shown in Fig. 8(c). Hence, \( (q_{11}^{PR}, q_{12}^{PR}) = (q_{31}^{PQ} \circ p_1(Q, P), q_{32}^{PQ} \circ p_2(Q, P)) \). Recall that \( (q_{31}^{PQ}, q_{32}^{PQ}) \) is a shortest link-disjoint lightpath pair in \( s(P, Q) \) such that \( q_{31}^{PQ} \) is an \( s-Q \) lightpath on \( \lambda_1 \), and \( q_{32}^{PQ} \) is an \( s-P \) lightpath on \( \lambda_2 \), and that \( p_1(Q, P) \) is a shortest \( Q-P \) lightpath on \( \lambda_1 \) in \( G(Q, P) \).

In \( c-1.4 \), both \( q_{11}^{PR} \) and \( q_{12}^{PR} \) go from \( Q \) to \( P \) through \( G(Q, P) \), as shown in Fig. 8(d). Hence, \( (q_{11}^{PR}, q_{12}^{PR}) = (q_{41}^{PQ}, q_{42}^{PQ}) \). Recall that \( (q_{41}^{PQ}, q_{42}^{PQ}) \) is a shortest link-disjoint lightpath pair in \( s(P, Q) \) such that \( q_{41}^{PQ} \) is an \( s-Q \) lightpath on \( \lambda_1 \), and \( q_{42}^{PQ} \) is an \( s-P \) lightpath on \( \lambda_2 \). Also recall that \( (q_1(Q, P), q_2(Q, P)) \) is a shortest \( Q-P \) lightpath on \( \lambda_1 \) and \( q_2(Q, P) \) is a \( Q-P \) lightpath on \( \lambda_2 \).

**Lemma 3.4:** Let \( [P, Q] \) be a 2-separator in the reduced 2-tree \( T[s, t] \). Let \( [P, R] \) be the successor of \( [P, Q] \), as shown in Fig. 7(a), or \( [P, R] \) is a peripheral where \( R \) coincides with \( t \), as shown in Fig. 7(c). Then, \( \gamma_2(P, R) = (q_{21}^{PR}, q_{22}^{PR}) \) is the shortest lightpath pair among the following four (possibly empty) lightpath pairs, as illustrated in Fig. 9:

\( c-2.1: (q_{1}^{PQ} \circ p_1(Q, P), q_2^{PQ} \circ p_2(Q, P)) \)
\( c-2.2: (q_{21}^{PQ}, q_2^{PQ} \circ p_2(Q, P)) \)
\( c-2.3: (q_{31}^{PQ} \circ q_1(Q, P), q_2^{PQ} \circ q_2(Q, P)) \)
\( c-2.4: (q_{41}^{PQ} \circ p_1(Q, P), q_2^{PQ} \circ p_2(Q, P)) \).

**Proof:** \( \gamma_2(P, R) = (q_{21}^{PR}, q_{22}^{PR}) \) is a shortest link-disjoint lightpath pair in \( s(P, R) \), with \( q_{21}^{PR} \) an \( s-P \) lightpath on \( \lambda_1 \), and \( q_{22}^{PR} \) an \( s-R \) lightpath on \( \lambda_2 \). There are exactly four possible cases, as shown in Fig. 9(a)–(d). The rest of the proof follows similar logic as used in the proof of Lemma 3.3.

**Lemma 3.5:** Let \( [P, Q] \) be a 2-separator in the reduced 2-tree \( T[s, t] \). Let \( [P, R] \) be the successor of \( [P, Q] \), as shown in Fig. 7(a), or \( [P, R] \) is a peripheral where \( R \) coincides with \( t \), as shown in Fig. 7(c). Then, \( \gamma_3(P, R) = (q_{31}^{PR}, q_{32}^{PR}) \) is the shortest lightpath pair among the following four (possibly empty) lightpath pairs, as illustrated in Fig. 10:

\( c-3.1: (q_{1}^{PQ} \circ p_1(Q, P), q_{12}^{PQ} \circ p_2(Q, P)) \)
\( c-3.2: (q_{21}^{PQ} \circ q_1(Q, P), q_{22}^{PQ} \circ q_2(Q, P)) \)
\( c-3.3: (q_{31}^{PQ} \circ p_1(Q, R), q_{32}^{PQ} \circ p_2(Q, R)) \)
\( c-3.4: (q_{41}^{PQ} \circ p_1(Q, R), q_{42}^{PQ} \circ p_2(Q, R)) \).

**Proof:** \( \gamma_3(P, R) = (q_{31}^{PR}, q_{32}^{PR}) \) is a shortest link-disjoint lightpath pair in \( s(P, R) \) such that \( q_{31}^{PR} \) is an \( s-R \) lightpath on \( \lambda_1 \), and \( q_{32}^{PR} \) is an \( s-P \) lightpath on \( \lambda_2 \). There are exactly four possible cases, as shown in Fig. 10(a)–(d). The rest of the proof follows similar logic as used in the proof of Lemma 3.3.

Now we have most of the building blocks for our algorithm.

Let us use the reduced 2-tree in Fig. 4(a) as an example. Using
Lemma 3.2, we can compute the values for $\gamma_i(B, C)$ for $i = 1, 2, 3, 4$. Using Lemmas 3.3–3.6, we can compute the values for $\gamma_i(B, F)$, then for $\gamma_i(F, I)$, and for $\gamma_i(I, Z)$. The solution to LDLP-CW can be decided by the following lemma.

Lemma 3.7: Suppose that we have computed $\gamma_i(P, t)$ for $i = 1, 2, 3, 4$, where $P \neq s$ and $P, t$ is a peripheral in the reduced 2-tree $T(s, t)$. Then, an optimal solution to LDLP-CW is the lightpath pair $(\pi_1, \pi_2)$, which is the shortest among the following four (possibly empty) lightpath pairs, as illustrated in Fig. 12(a)–(d):

- c-F.1: $(q_1^{P, t} \circ q_3(P, t), q_2^{P, t} \circ q_2(P, t))$;
- c-F.2: $(q_3^{P, t} \circ q_1(P, t), q_2^{P, t} \circ q_2(P, t))$;
- c-F.3: $(q_1^{P, t} \circ q_3(P, t), q_2^{P, t} \circ q_2(P, t))$;
- c-F.4: $(q_4^{P, t} \circ q_4(P, t))$.

The instance of LDLP-CW has no solution when all of the above four lightpath pairs are $(\phi, \phi)$. □

**Proof:** The proof is similar to that of Lemma 3.3. As illustrated in Fig. 12, there are four possible cases.

- c-F.1: Both $\pi_1$ and $\pi_2$ go from $P$ to $t$ through $G(P, t)$.
- c-F.2: $\pi_1$ goes from $P$ to $t$ through $G(P, t)$, but $\pi_2$ does not use any link in $G(P, t)$.
- c-F.3: $\pi_2$ does not use any link in $G(P, t)$, but $\pi_2$ goes from $P$ to $t$ through $G(P, t)$.
- c-F.4: Neither $\pi_1$ nor $\pi_2$ uses any link in $G(P, t)$.

In c-F.1, the solution uses $\gamma_i(P, t) = (q_1^{P, t}, q_1^{P, t})$ and $\beta(P, t) = (q_1^{P, t}, q_2^{P, t})$. Since $\gamma_1(P, t)$ is a shortest pair of link-disjoint $s$–$P$ lightpaths in $s \in G(P, t)$, and $\beta(P, t)$ is a shortest pair of link-disjoint $P$–$t$ lightpaths in $G(P, t)$, $(q_1^{P, t} \circ q_3(P, t), q_2^{P, t} \circ q_2(P, t))$ is an optimal solution for LDLP-CW in this case.

The other three cases can be similarly proved, where c-F.2 uses $\gamma_2(P, t)$ and $\alpha_1(P, t)$; c-F.3 uses $\gamma_3(P, t)$ and $\alpha_2(P, t)$; c-F.4 uses $\gamma_4(P, t)$.

**E. Linear-Time Algorithm for LDLP-CW**

After the detailed descriptions of all the building blocks, we can summarize our design of the algorithm in Algorithm 2. On a high level, Algorithm 2 can be divided into three phases. In the first phase (lines 1–9), we perform contraction operations to compute the reduced 2-tree $T(s, t)$ from $T$. The order of the contraction operations follows that of Algorithm 1, while the update of the link attributes $(\alpha_1, \alpha_2, \beta)$ follows the rules in Lemma 3.1. In the second phase (lines 10–19), we perform further contraction operations to convert the reduced 2-tree $T(s, t)$ into a triangle. The order of contraction operations follows that described in Remark 1, while the link attributes $(\gamma_1, \gamma_2, \gamma_3, \gamma_4)$ are initialized according to Lemma 3.2 (line 10) and updated according to Lemmas 3.3–3.6 (lines 11–19). Finally, the optimal solution is computed according to Lemma 3.7 (lines 20–21).

**Algorithm 2: Alg-LDLP-CW**

**Input:** A WDM optical network $T$ with a 2-tree topology. A source node $s$. A destination node $t$. Two chosen wavelengths $\lambda_1 \in A$ and $\lambda_2 \in A$.

**Output:** A shortest pair of link-disjoint $s$–$t$ lightpaths $\pi_1$ on $\lambda_1$ and $\pi_2$ on $\lambda_2$, if such a pair exists.

1: for each link $[u, v]$ in $T$ do
2: Initialize $\alpha_1(u, v), \alpha_2(u, v), \beta(u, v), \alpha_1(v, u), \alpha_2(v, u), \beta(v, u)$, using the rules in Section III-A;
3: end for
4: while $\exists$ a degree-2 node $z$ other than $s$ and $t$ in $T$ do
5: if the two nodes adjacent to $z$ are $s$ and $t$ then
6: break;
7: end if
8: Assume $z$ makes a triangle with link $[x, y]$. Contract node $z$ onto link $[x, y]$, and update $\alpha_1(x, y), \alpha_2(x, y), \beta(x, y)$, using the rules in Lemma 3.1.
9: end while

[By now we have computed the reduced 2-tree]

10: Let $\Delta_{xys}$ be the triangle in the reduced 2-tree that contains $s$. Initialize $\gamma_1(x, y), \gamma_2(x, y), \gamma_3(x, y)$, and $\gamma_4(x, y)$ according to Lemma 3.2.
11: while $\exists$ a degree-2 node $v \neq t$ in the reduced 2-tree do
12: if the reduced 2-tree $T(s, t)$ is a triangle then
13: break;
14: end if
15: Let $[P, Q]$ be the 2-separator that forms a triangle with $v$. Let $[P, R]$ be the successor of $[P, Q]$, or $[P, R]$ be a peripheral and $R$ coincides with $t$.
16: Compute $\gamma_i(P, R)$ for $i = 1, 2, 3, 4$ according to Lemmas 3.3–3.6.
17: Delete node $v$ in the reduced 2-tree.
18: STOP if
$\gamma_1(P, R) = \gamma_2(P, R) = \gamma_3(P, R) = \gamma_4(P, R) = (\phi, \phi)$;
19: end while
20: Compute the shortest pair of link-disjoint lightpaths $(\pi_1, \pi_2)$ according to Lemma 3.7. STOP if such a pair does not exist;
21: Extract the link-disjoint lightpaths $\pi_1$ and $\pi_2$ using a top-down method;

**Example 6:** We use the network in Fig. 5 to illustrate the algorithm. Here, we assume that $\lambda_1 = 1$ and $\lambda_2 = 2$. By the time the reduced 2-tree in Fig. 4(a) is constructed, we have the following nonempty link attributes (for each link $[u, v]$, we only list the attribute for one ordered pair):

- $\alpha_1(A, B) = (A, B)$, $\alpha_2(A, B) = (A, B)$,
- $\alpha_1(A, C) = (A, D, C)$, $\alpha_1(B, C) = (A, B, C)$,
- $\alpha_2(B, F) = (B, F)$, $\alpha_2(B, F) = (B, F)$,
- $\alpha_1(B, I) = (B, J, K, I)$, $\alpha_1(C, F) = (C, F)$,
- $\alpha_2(C, F) = (C, F)$, $\beta(G, H, F) = (C, F, F)$,
- $\alpha_1(F, I) = (F, I)$, $\alpha_2(F, I) = (F, X, I)$,
- $\beta(F, I) = (F, I)$, $\beta(F, I) = (F, X, I)$,
- $\alpha_1(F, Z) = (F, Z)$, $\alpha_2(F, Z) = (F, Z)$.

The sequence of 2-separators of the reduced 2-tree $T(A, Z)$ is $[B, C], [B, F], [F, I]$.

At $[H, C]$, the values of $\gamma_i(H, C)$ are computed according to Lemma 3.2:

- $\gamma_1(H, C) = (\phi, \phi)$, $\gamma_2(H, C) = (A, B), \gamma_2(H, C) = (A, B), \gamma_2(H, C) = (A, D, C)$.

Note that we also know the values of $\gamma_i(C, B)$ by the relationship $\gamma_i(C, B) = \gamma_{5-i}(B, C)$.
At \([B, F]\), the values of \(\gamma_1(B, F)\), \(\gamma_2(B, F)\), \(\gamma_3(B, F)\), and \(\gamma_4(B, F)\) are computed according to Lemmas 3.3, 3.4, 3.5, and 3.6, respectively: \(\gamma_1(B, F) = (\phi, \phi), \gamma_2(B, F) = (A, (A, D, C, H, F)), \gamma_3(B, F) = (\phi, \phi),\) and \(\gamma_4(B, F) = (A, (A, D, C, H, F))\). Note that we also know the values of \(\gamma_i(B', B)\) by the relationship \(\gamma_i(B, F) = \gamma_{i-1}(B, F)\).

At \([F, I]\), the values of \(\gamma_1(F, I)\), \(\gamma_2(F, I)\), \(\gamma_3(F, I)\), and \(\gamma_4(F, I)\) are computed according to Lemmas 3.3, 3.4, 3.5, and 3.6, respectively: \(\gamma_1(F, I) = (A, (A, B, C, F)), (A, D, C, H, F), \gamma_2(F, I) = (\phi, \phi), \gamma_3(F, I) = (A, B, J, K, I), (A, D, C, H, F)),\) and \(\gamma_4(F, I) = (\phi, \phi)\). Note that we also have the values for \(\gamma_i(I, F)\) by the relationship \(\gamma_i(I, F) = \gamma_{i-1}(I, F)\).

At \([I, Z]\), the values of \(\gamma_1(I, Z)\), \(\gamma_2(I, Z)\), \(\gamma_3(I, Z)\), and \(\gamma_4(I, Z)\) are computed according to Lemmas 3.3, 3.4, 3.5, and 3.6, respectively: \(\gamma_1(I, Z) = (A, (A, B, F, I), (A, D, C, H, F, X, I)), \gamma_2(I, Z) = (A, (A, B, F, I), (A, D, C, H, F, Z)), \gamma_3(I, Z) = (A, (A, B, F, Z), (A, I, C, H, F, X, I)),\) and \(\gamma_4(I, Z) = (\phi, \phi)\).

Finally, in line 20 of the algorithm, we find that the shortest pair of \(A\)–\(Z\) lightpaths is \((A, (A, B, F, I, Z), (A, D, C, H, F, Z))\). Meanwhile in the illustrations, we have list out the lightpaths during each step. In actual implementations, we only need to remember the length of the lightpath, as well as its composition (the concatenation of one or two lightpaths that are previously computed). Once we reached the final answer, the pair of lightpaths can be extracted in \(O(n)\) time.

Our analysis in this section leads to the following theorem.

**Theorem 3.1:** The worst-case time complexity of Algorithm 2 is \(O(n)\). If there exists a pair of link-disjoint \(s\)–\(t\) lightpaths \(\pi_1\) and \(\pi_2\) such that \(\lambda_1\) is available on each link on \(\pi_1\) and that \(\lambda_2\) is available on each link on \(\pi_2\), Algorithm 2 finds such a pair of lightpaths with minimum total length. Otherwise, Algorithm 2 stops without outputting any lightpath.

**Proof:** Each contraction operation takes \(O(1)\) time. There are \(O(n)\) contraction operations. This leads to the time complexity of the algorithm.

The correctness of the algorithm follows from Lemma 2.2 and Lemmas 3.1–3.7. Let \(\pi_1^{opt}\) and \(\pi_2^{opt}\) be the optimal solution for LDSLP-CW, where \(\pi_1^{opt}\) is an \(s\)–\(t\) lightpath on \(\lambda_1\), and \(\pi_2^{opt}\) is an \(s\)–\(t\) lightpath on \(\lambda_2\) such that the two are link-disjoint and form the shortest pair. Let \([P, Q]\) be a separator of \(s\) and \(t\) (i.e., \([P, Q]\) is a 2-separator in \(T(s, t)\)). Lemma 2.2 implies that at least one of the following four cases is true.

1) Both \(\pi_1^{opt}\) and \(\pi_2^{opt}\) go through \(P\). In this case, the subpath of \(\pi_1^{opt}\) from \(s\) to \(P\) can be replaced by \(q_{11}^{PQ}\) (the first element of \(\gamma_1(P, Q)\)) without losing optimality, and the subpath of \(\pi_2^{opt}\) from \(s\) to \(P\) can be replaced by \(q_{12}^{PQ}\) (the second element of \(\gamma_1(P, Q)\)) without losing optimality.

2) \(\pi_1^{opt}\) goes through \(P\), and \(\pi_2^{opt}\) goes through \(Q\). In this case, the subpath of \(\pi_1^{opt}\) from \(s\) to \(P\) can be replaced by \(q_{21}^{PQ}\) (the first element of \(\gamma_2(P, Q)\)) without losing optimality, and the subpath of \(\pi_2^{opt}\) from \(s\) to \(Q\) can be replaced by \(q_{22}^{PQ}\) (the second element of \(\gamma_2(P, Q)\)) without losing optimality.

3) \(\pi_1^{opt}\) goes through \(Q\), and \(\pi_2^{opt}\) goes through \(P\). In this case, the subpath of \(\pi_1^{opt}\) from \(s\) to \(Q\) can be replaced by \(q_{31}^{PQ}\) (the first element of \(\gamma_3(P, Q)\)) without losing optimality, and the subpath of \(\pi_2^{opt}\) from \(s\) to \(P\) can be replaced by \(q_{32}^{PQ}\) (the second element of \(\gamma_3(P, Q)\)) without losing optimality.

4) Both \(\pi_1^{opt}\) and \(\pi_2^{opt}\) go through \(Q\). In this case, the subpath of \(\pi_1^{opt}\) from \(s\) to \(Q\) can be replaced by \(q_{41}^{PQ}\) (the first element of \(\gamma_4(P, Q)\)) without losing optimality, and the subpath of \(\pi_2^{opt}\) from \(s\) to \(Q\) can be replaced by \(q_{42}^{PQ}\) (the second element of \(\gamma_4(P, Q)\)) without losing optimality.

Lemma 3.1 ensures that lines 1–9 of Algorithm 2 correctly compute the reduced 2-tree \(T(s, t)\) and the link attributes \(\alpha_1, \alpha_2, \beta\). Lemmas 3.2–3.6 ensure that lines 10–19 correctly compute the attributes \(\gamma_1, \gamma_2, \gamma_3, \gamma_4\), while converting the reduced 2-tree \(T(s, t)\) into a triangle. Lemma 3.7 ensures that Algorithm 2 computes a pair of link-disjoint \(s\)–\(t\) lightpaths that are as good as the optimal pair. Hence, Algorithm 2 solves the LDSLP-CW problem correctly.

**IV. POLYNOMIAL-TIME ALGORITHM FOR LDLP WITH A PARTIAL 2-TREE TOPOLOGY**

For any given WDM network with a partial 2-tree topology, we can apply the linear-time algorithm of [33] to complete the partial 2-tree to a 2-tree. The algorithm of [33] may add artificial links to augment the partial 2-tree into a 2-tree. Note that we will not change the network topology. To ensure this, as a technical convention, we mark these added artificial links as if they have no wavelength available at all.

The underlying topology of the WDM network in Fig. 13 is a partial 2-tree, but not a 2-tree. It can be completed to the 2-tree in Fig. 5 by adding artificial links \([A, C], [B, I], [I, J]\), which have no wavelength available. Note that any link added in the completing process has no wavelength available—because the link simply does not exist in the physical network. Assuming a 2-tree topology simplifies the description of our algorithm in Section III.

Now we are ready to present our polynomial-time algorithm for solving the LDLP problem when the underlying network topology is a partial 2-tree. The algorithm can be divided into two phases. In the first phase, we augment the network topology \(T\) from a partial 2-tree to a 2-tree using the algorithm of [33]. Since the links added in the process have no wavelength available, the augmenting process does not change the solution to the LDLP problem. In the second phase, we solve LDSLP-CW for all possible pairs of \(\lambda_1, \lambda_2\) and choose the shortest lightpath pair computed as the overall solution. The algorithm is listed as Algorithm 3.
Algorithm 3: Alg-LDLP

Input: A WDM optical network $T$ with a partial 2-tree topology. A source node $s$. A destination node $t$.

Output: A shortest pair of link-disjoint lightpaths $\pi_{1}^{opt}$ (on $\lambda_{1}^{opt}$) and $\pi_{2}^{opt}$ (on $\lambda_{2}^{opt}$).

1: Apply the linear-time algorithm of [33] to augment the partial 2-tree $T$ to a 2-tree. Mark all links added in this process as having no wavelength available.
2: $\pi_{1}^{opt} \leftarrow \phi, \pi_{2}^{opt} \leftarrow \phi, \ell(\pi_{1}^{opt}, \pi_{2}^{opt}) \rightarrow \infty$;
3: for $\lambda_1 \leftarrow 1, 2, \ldots, W$ do
4:     for $\lambda_2 \leftarrow 1, \lambda_1 + 1, \ldots, W$ do
5:         Apply Alg-LDLPCW to compute a shortest pair of link-disjoint lightpaths: $\pi_{1}$ on $\lambda_{1}$, and $\pi_{2}$ on $\lambda_{2}$;
6:         if $\ell(\pi_{1}) + \ell(\pi_{2}) < \ell(\pi_{1}^{opt}) + \ell(\pi_{2}^{opt})$ then
7:             $\pi_{1}^{opt} \leftarrow \pi_{1}, \pi_{2}^{opt} \leftarrow \pi_{2}, \lambda_{1}^{opt} \leftarrow \lambda_{1},$ \( \lambda_{2}^{opt} \leftarrow \lambda_{2} \);
8:         end if
9:     end for
10: end for
11: if $(\pi_{1}^{opt}, \pi_{2}^{opt}) = (\phi, \phi)$ then
12:     STOP: no disjoint lightpath pair exists;
13: else
14:     OUTPUT $(\pi_{1}^{opt}, \pi_{2}^{opt})$ and $(\lambda_{1}^{opt}, \lambda_{2}^{opt})$;
15: end if

Theorem 4.1: Assume that $T$ is a partial 2-tree. Then, Algorithm 3 solves the LDLP problem in $O(nW^2)$ time. □

Proof: Augmenting $T$ to a 2-tree takes $O(nW)$ time. Since we are solving LDLP-CW for $O(W^2)$ wavelength pairs $(\lambda_{1}, \lambda_{2})$, the time complexity of Algorithm 3 is $O(nW^2)$. The instance of LDLP has a solution if and only if one of the $O(W^2)$ instances of LDLP-CW has a solution. Therefore, Algorithm 3 correctly solves the LDLP problem.

For example, assume that the input to Algorithm 3 is the WDM network shown in Fig. 13 with source node $s = A$ and destination node $t = Z$. The algorithm will first augment the partial 2-tree shown in Fig. 13 to the 2-tree shown in Fig. 5. The algorithm will run Alg-LDLPCW three times, with $(\lambda_{1}, \lambda_{2})$ being set to $(1,1), (1,2), \text{ and } (2,2)$, respectively.

With $\{\lambda_{1}, \lambda_{2}\} = \{1, 1\}$, Alg-LDLPCW returns no lightpath pair. With $\{\lambda_{1}, \lambda_{2}\} = \{1, 2\}$, Alg-LDLPCW returns the lightpath pair $\{(A, B, F, I, Z), (A, D, C, H, F, Z)\}$. With $\{\lambda_{1}, \lambda_{2}\} = \{2, 2\}$, Alg-LDLPCW returns no lightpath pair. Hence, Alg-LDLPCW returns the lightpath pair $\{(A, B, F, I, Z), (A, D, C, H, F, Z)\}$, which is an optimal solution.

V. NUMERICAL RESULTS

To verify the effectiveness of our proposed algorithm, we implemented our algorithm (denoted by LDLP) and compared it to the well-known shortest active path first heuristic (denoted by SAPF). The comparison was done on randomly generated networks whose underlying topologies are 2-edge connected subgraphs of minimum isolated failure immune networks. The number of nodes $n$ varied from 100 to 500 in steps of 100. For each value of $n$, we first randomly generated a 2-tree network on $n$ nodes. Then, for a given decimal number $p \in (0,1)$, we randomly deleted $p \times n$ links in the 2-tree to obtain a partial 2-tree (we sequentially deleted links whose deletion does not destroy the 2-edge connectivity of the resulting graph). The resulting 2-edge connected partial 2-tree was used as the underlying topology. We tested with each link having $W$ wavelengths, with $W = 2, 4, 8, 16$. We studied four different network load values $R = 0, 25, 50, 75$, where $R\%$ of the network resources are being used. Here, each available wavelength on a link counts as one unit of resource. Therefore, a network with $m$ links has $mW$ resources before any of the wavelengths is used. For a partial 2-tree with $m$ links, and any given value $R = 0, 25, 50, 75$, we randomly reserved $(R/100) \times mW$ units resource so that these resources become unavailable. For each network so obtained, we generated $5 \times n$ connection requests with the source and destination nodes randomly chosen. Under each of these combinations, we compared LDLP to SAPF. All the tests were performed on a 3.2-GHz Linux PC. The results reported here are the average over 50 test cases.

To compare LDLP to SAPF, we used the same network state information and the same connection request, given by a source–destination pair. For this reason, we did not reserve the resource (i.e., wavelength) after the requests are accepted. For each connection request, we used both algorithms to find a pair of link-disjoint lightpaths. There are four possible results.

1) Neither algorithm finds a pair of link-disjoint lightpaths.
2) LDLP finds such a pair, but SAPF fails.
3) Both algorithms find a pair of lightpaths, but the pair found by LDLP is shorter.
4) Both algorithms find a pair of lightpaths, and the two pairs are of equal length.

We are only interested in the results where at least one algorithm finds a pair (i.e., 2)–4). Let $N_{2}$, $N_{3}$, and $N_{4}$ denote the number of requests leading to 2), 3), and 4), respectively. The improvement on the number of successful requests achieved by LDLP over SAPF is then defined as $N_{2}/(N_{3} + N_{4})$. The improvement on the quality of the lightpaths is defined as $N_{2}/N_{4}$. Figs. 14 and 15 show these two improvements when $W = 4$. Specifically, Fig. 14(a) and (b) illustrates the improvement on the number of successful requests when $p = 0.2$ and $p = 0.5$, respectively. We observe that the improvement can reach up to 25% in certain conditions for both levels of partial 2-trees. The most improvement happens when the network is heavily loaded but not severely loaded, e.g., $p = 0.2$ with $R = 50$ and $p = 0.5$ with $R = 25$ or 50. When the network is load-free or severely loaded, the improvement is small. The reason for the load-free network is that SAPF is highly likely to succeed in such networks. The reason for the severely loaded network is that there are not enough resources to have two link-disjoint lightpaths even for LDLP in this case. The last observation is that the number of nodes in the networks almost does not affect
The running time of LDLP is shown in Fig. 18. We observe that the running time is proportional to \( p \) for any fixed \( W \), and proportional to \( W^2 \) for any fixed \( n \), which verifies our theoretical analysis. We also observe that our algorithm is very fast in practice.

VI. CONCLUSION

We have presented a polynomial-time algorithm for computing a pair of link-disjoint lightpaths in a WDM optical network whose underlying topology is a subgraph of a minimum IFI network. Our algorithm can be easily modified to compute a pair of node-disjoint lightpaths in such networks.

IFI networks have received a lot of attention in the literature [13], [38] because they are fault-tolerant to multiple failures as long as the faults are isolated. Minimum IFI networks are attractive because they are scalable and often admit simple solutions to many networking problems [6], [9], [33]. Besides providing guidelines for the design of future survivable optical networks, our work can have an impact on improved decision making in existing optical networks. While the backbone of existing optical networks may not be subgraphs of minimum IFI networks, we can compute a subgraph of the given network that has the structure assumed in this paper. This can be achieved, using offline computation, by hiding a few network links so that the resulting network has no subgraph that is homeomorphic to the four-vertex complete graph \( K_4 \) [34]. Our algorithm can be applied to this resulting network, while the links that have been hidden in the above process can be used to provide additional protection.

ACKNOWLEDGMENT

The authors thank the Associate Editor and the anonymous reviewers whose comments on earlier versions of this paper have helped to significantly improve the presentation of the paper.

REFERENCES


Guoliang Xue (M’96–SM’99–F’11) received the B.S. degree in mathematics and M.S. degree in operations research from Qufu Normal University, Qufu, China, in 1981 and 1984, respectively, and the Ph.D. degree in computer science from the University of Minnesota, Minneapolis, MN, USA, in 1991. He is a Professor of computer science and engineering with Arizona State University, Tempe, AZ, USA. His research interests include survivability, security, and privacy issues in smart grid, computer networks, and social networks. He also applies game theory to model selfish behavior in cooperative networks. Prof. Xue is an Associate Editor of the IEEE/ACM TRANSACTIONS ON NETWORKING and IEEE Network. He was a Keynote Speaker at IEEE LCN 2011 and a TPC Co-Chair of IEEE INFOCOM 2010.

Ravi Gottapu received the M.S. degree in computer science and engineering from Arizona State University, Tempe, AZ, USA, in 2003. He is with Amazon Inc., Seattle, WA, USA. Prior to joining Amazon, he was with Microsoft, Inc., Seattle, WA, USA. His research interests include efficient algorithms for optimization problems in networking, with applications to QoS and survivability.

Xi Fang (S’09) received the B.S. and M.S. degrees from Beijing University of Posts and Telecommunications, Beijing, China, and will receive the Ph.D. degree in computer science from Arizona State University, Tempe, AZ, USA, in 2013. He is a Software Development Engineer with Microsoft, Inc., Redmond, WA, USA. His research interests include big data, cloud computing, wireless networks, and smart grids. Mr. Fang received Best Paper awards at IEEE ICC 2012, IEEE MASS 2011, and IEEE ICC 2011.

Dejun Yang (S’08) received the B.S. degree in computer science from Peking University, Beijing, China, in 2007, and is currently pursuing the Ph.D. degree in computer, informatics, and decision systems engineering (CIDSE) at Arizona State University, Tempe, AZ, USA.

His research interests include economic and optimization approaches to networks, crowdsourcing, smart grid, big data, and cloud computing. Mr. Yang has received Best Paper awards at IEEE MASS 2011 and IEEE ICC 2011 and 2012, and was a Best Paper Award runner-up at IEEE ICNP 2010.

Krischnaiyan Thulasiraman (M’72–SM’84–F’90) received the Bachelor’s and Master’s degrees from the University of Madras, India, in 1963 and 1965, respectively, and the Ph.D. degree from the Indian Institute of Technology (IIT), Madras, India, in 1968, all in electrical engineering.

He is the Hitachi Chair Professor with the School of Computer Science, University of Oklahoma, Norman, OK, USA. His research interests have been in graph theory, algorithms, and applications in electrical networks and fault tolerance in optical networks.

Dr. Thulasiraman has received several awards and honors, including being an elected member of the European Academy of Sciences in 2002 and receiving the IEEE CAS Society Charles A. Desoer Technical Achievement Award in 2006.