Optimal Diverse Routing in Telecommunication Fiber Networks

Ramesh Bhandari

AT&T Bell Laboratories
Crawfords Corner Road, Holmdel, NJ 07733, USA
(908) 949-0693

Abstract

Telecommunication fiber networks can be more complicated than the traditional graph-theoretic networks of nodes and links. This is due to practical and economic considerations. In this paper, we consider two major types of deviations from the traditional graph-theoretic network, and provide an algorithm for the shortest pair of physically-disjoint paths between a given pair of nodes in the network. Such disjoint paths can be used for improving the reliability of the network, e.g., one path may be used as a back up while the other is actually used for transmission of data. Alternatively, the entire traffic between the given pair of nodes in the network may be divided equally over the two disjoint paths so that if a node or link on one of the paths fails, not all of the traffic is lost. Optimizing the length of disjoint paths helps in reducing the amount of fiber usage and network costs.

1. Introduction

With the advent of fiber and its increasing deployment in networks, the risk of losing large volumes of data due to a span cut or node failure has increased dramatically. One way to ensure continuity of service between a pair of nodes in the network is via provisioning of two physically-disjoint paths, i.e., two paths that are node-disjoint as well span-disjoint. The term span refers to physical link in the network, which normally is a buried conduit carrying the communication fiber. When two such paths are provided between a pair of nodes in the network, data may be transmitted over one path with the second acting as a backup; alternatively, the data may be transmitted over the two paths with the better of the two signals selected at the receiving end. In general, robustness of a communication network is improved by splitting and routing traffic between every pair of nodes over the available disjoint paths, since a node or span failure on one of the paths affects only 50% of the traffic. Furthermore, assignment of spare capacity on the alternate path to accomodate the affected traffic then leads to a robust network design. Clearly, the use of optimal physical-diversity algorithms to find diverse routes ensures that the amount of fiber that needs to be laid along the diverse routes is minimal. In diverse provisioning of business services, optimality implies reduced cost.

While algorithms for disjoint paths have been given in the past [1-3], they apply only to networks described by nodes and logical connections called links. Fig. 1 is an example of such a network. Dashed lines are the links. A link indicates that a connection exists between the pair of nodes in question, but does not necessarily represent the physical path between the connected nodes. Because of practical and economic considerations, telecommunication fiber networks may be so constructed that the physical paths of individual links may overlap with each other. For example, in Fig. 1, the fiber connecting nodes B and G and the fiber connecting nodes B and C may pass through the same conduit for part of their ways. Such possible span-sharing by links complicates the network, and leads to the requirement of new algorithms for determination of physically-disjoint paths.

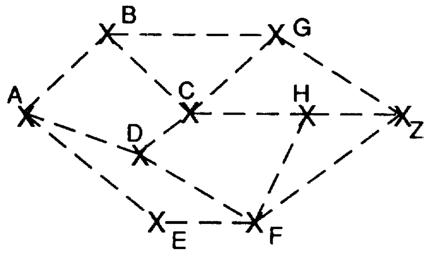

Fig. 1 A network of nodes and links.

In this paper, we consider two common types of deviations from the standard graph-theoretic network of nodes and links, and provide an algorithm for the shortest pair of physically-disjoint paths between a given pair of nodes in the network. To our knowledge, such type of an algorithm for real-life networks has not been given before. Section 2 describes the considered network, and Section 3 constructs the algorithm. Section 4 considers additional deviations from the standard graph-theoretic networks, and the applicability of the developed algorithms to them.

2. Network description

We assume that the network is bidirectional, and is described by nodes, links and spans. Links indicate connections between nodes, while spans comprise the actual physical network. In what follows, we indicate links by dashed lines and spans by continuous lines, and consider the following configurations:

1) *The Standard Configuration*

The connection between a pair of nodes is via a single span, as shown in Fig. 2.

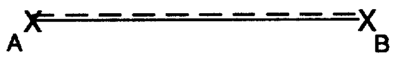

Fig. 2 Link AB which has a single span, AB.

2) *Fork Configuration*

Fig. 3(a)is an example of fork configuration. Nodes A and B are connected physically via spans AO and OB, point O being a junction, i.e., the fiber between nodes A and B is carried within two contiguous conduits. The connection or link between A and B is indicated by the dashed line AB. In a similar way, nodes A and C are physically connected via spans AO and OC, and this link between A and C is indicated by the dashed line AC. However, nodes B and C are not similarly connected, i.e., node C may only be reached from node B via traversal of spans BO, OA, AO, and OC in the order given, or alternatively by some other path (in the network) that does not involve junction O. In short, a link between nodes B and C, similar to the ones between nodes A and B, and nodes A and C, is missing in Fig. 3(a). Fig. 3(b) is a generalization of Fig. 3(a) It is a fork

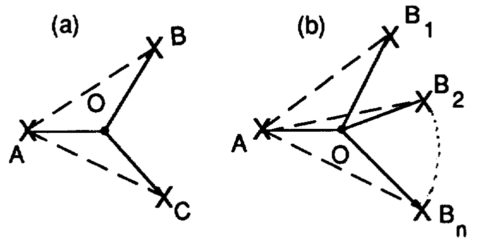

Fig.3(a) Fork configuration with two prongs (also called Y configuration in the text), and (b) fork configuration with n prongs

configuration with n prongs; the same constraints apply to connections between any pair of nodes B_i and B_j, $i \neq j$, i = 1,2,---n, and j=1,2,----n. Span AO is common to n links, AB_1, AB_2, ----, AB_n. Therefore, an accidental cut of span AO results in the simultaneous loss of all these links.

3) *Express Link*

In Fig. 4, links AB, BC, and CD (dashed lines) consist of single spans (continuous line) AB, BC, and CD, respectively. In addition, nodes A and D are also connected directly by a fiber that originates at node A and terminates at node D, traversing spans AB, BC, and CD. This connection or link is indicated by the dashed line AD, and called an express link. The spans of an express link are always the spans of the links it traverses. In Fig. 4, it traverses 3 links. In general, an express link may span n links, where n is an integer greater than 1. Furthermore, the individual links may consist of more than one span, as in the fork configuration.

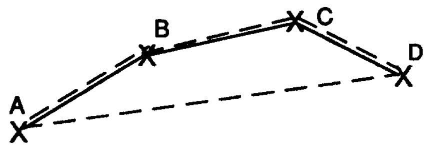

Fig. 4 Express link AD traversing three links, AB, BC, and CD.

In what follows, we assume that the network under consideration is made up of network configurations depicted in Figs. 2, 3, and 4. Figs. 3 and 4 are configurations where spans are shared by different links. Fig. 5 is an example of a network that includes such configurations. This network is identical to the network of Fig. 1 at the link-level (dashed lines). There

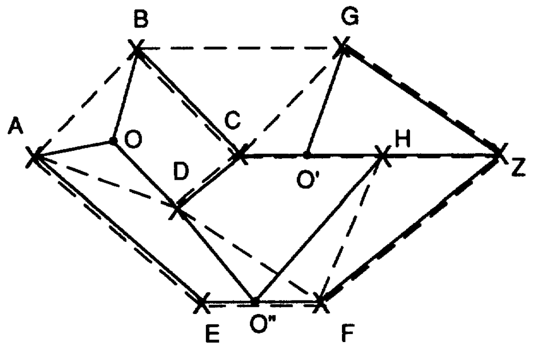

Fig.5 A network with three fork configurations (junction points O, O', O") and express link BG (span compostion: BC, CO', O'G).

are 3 fork configurations with junction points: O, O', O", and an express link, BG, traversing links BC and CG. The problem is to find an optimal pair of physically-disjoint paths between a given pair of nodes in a network such as Fig. 5. Optimality refers to minimum total span miles for the pair of paths.

Clearly, the available shortest pair of node-disjoint paths (SPNP) algorithm [1-3] valid for a standard graph-theoretic network (as in Fig. 1) cannot be applied to the network of Fig. 5 at the span (physical) level, because the junction nodes O, O', O" are not true nodes; as discussed above, while node O in Fig. 3(b) can be accessed from the neighboring nodes, A, B_1, --- B_n, traffic routed to node O from node B_i cannot be routed to node B_j ($j \neq i$), in the next step. On the other hand, the network at the link level (dashed lines) consists of true nodes, and the SPNP algorithm is applicable at the link-level. The links may be weighted by the sum of the length of the spans comprising the link. The SPNP algorithm when applied at the link level in Fig. 5 then gives the shortest pair of paths, each path in the pair defined by a sequence of links. Optimality is with respect to span mileage. However, the paths, although node-disjoint, are not necessarily span-disjoint. For example, for nodes A and Z, the SPNP algorithm may find the optimal pair of node-disjoint paths to be ABGZ and ADCHZ, in which case spans AO and CO' would be common to the two paths found. To avoid such span commonness, a new algorithm is needed. In the next section, we not only give an algorithm for finding physically-disjoint paths (node as well as span-disjoint), but also ensure that the pair of paths obtained is shortest with respect to span mileage.

3. Algorithm for the shortest pair of physically-disjoint paths

Below we develop the algorithm for the shortest pair of physically-disjoint paths for a given pair of nodes in a practical network of the type depicted in Fig. 5. In fact, the algorithm we provide for this network may be considered from a graph-theoretic standpoint a solution of a special case of the general problem, which appears to be NP-complete [4]. In other words, the general problem, which considers all types of span-sharing topologies, cannot be solved in polynomial time. The special case we are addressing, however, comprises span-sharing topologies that appear to be common in telecommunication fiber networks. As we shall see below, this particular problem is solvable in polynomial time.

Our strategy here is to perform network transformations such that the SPNP algorithm (which is a polynomial-time algorithm [1-3]) can eventually be used at the link-level, while preserving physical disjointness in the searched paths. Therefore, as a first step, we describe the SPNP algorithm. The SPNP algorithm we give below [3] is an improved version of the Suurballe algorithm [1,2] in that it does not require a general shortest path algorithm like that of Ford's [5,7] or, alternatively a special canonic transformation in order to facilitate the use of the popular Dijkstra algorithm [6,7]. Rather, the Dijkstra algorithm here is altered slightly to circumvent the need for the usual canonic transformation. This modified Dijkstra algorithm is given in Appendix A.

3.1 Shortest pair of node-disjoint paths algorithm

Because the network under consideration is bidirectional, each link is equivalent to two oppositely directed arcs of equal length. For example, in Fig. 1, link BC is equivalent to two arcs, BC and CB, the former implying that node C can be reached from node B via link BC, and the latter that node B can be reached from node C via the same link.

An SPNP algorithm is essentially a shortest pair of link-disjoint algorithm applied in a network which has been suitably modified by splitting a certain subset of nodes. It is the operation of node-splitting that gives rise to node-disjointness in the pair of paths. Using the standard node-splitting procedure [5,1,2], the algorithm for the shortest pair of node-disjoint paths is a series of following steps:

1. For the given pair of nodes under consideration, find the shortest path using the algorithm given in Appendix A. For illustration, refer to Fig. 6a.

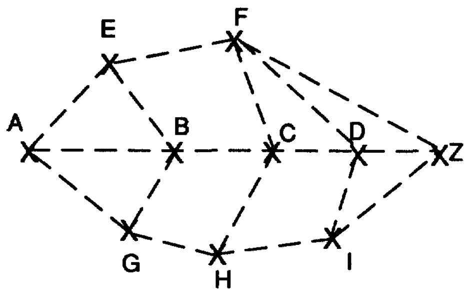

Fig. 6a Network of nodes and links; shortest path is assumed to be ABCDZ with A and Z, the starting and terminating nodes, respectively in the shortest path algorithm.

2. Replace each link on the shortest path by an arc directed towards the originating node (see Fig. 6b). Make the length of the arcs negative.

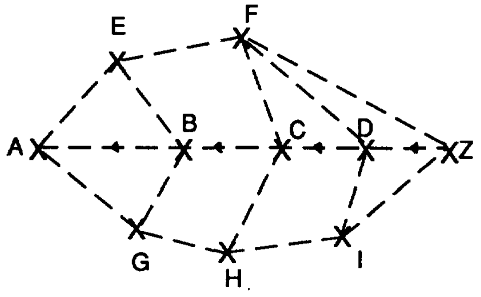

Fig. 6b Network with shortest path edges replaced with negative arcs directed towards the starting node, A.

3. Split each node on the shortest path (except the end point nodes) into two colocated subnodes, joined by an arc of length zero and directed towards the starting node. Replace external links connected to nodes on the shortest path by two arcs of the same and original length, and connected to the two subnodes as shown in Fig. 6c.

4. Run the shortest path algorithm (Appendix A) again. .

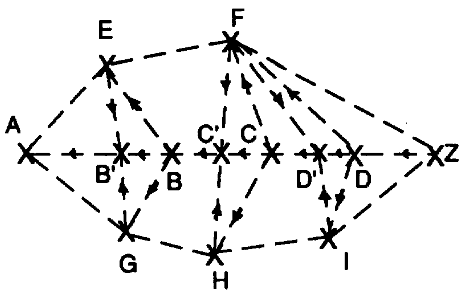

Fig. 6c Network modified by node-splitting.

5. Remove the zero length arcs; coalesce the subnodes into their parent nodes. Replace the single arcs of the shortest path with their original links (of positive length). Remove overlapping arcs of the two paths found to obtain the shortest pair of paths.

Step 2 above permits interlacing of the second path (found in Step 4) with the first; e.g., if the second path determined in Step 4 is AEFD'CHIZ (see Fig. 6c), it is said to interlace with the part CD of the first path found in Step 1 (see Fig. 6a); the pair of vertex-disjoint paths obtained in this case would be (AEFDZ,ABCHIZ).

We now consider a network interspersed with express links and fork configurations, as in Fig. 5. We first focus on the express links.

3.2 Express Links

Assumption 1 *The span content of an express link is exactly the same as the sum of the span contents of the individual links comprising the express link.*

We now define the following network:

Network 2 *This network is the same as the original network, except that express links are removed.*

For example, in such type of a network, express link BG in Fig. 5 would be absent.

Theorem 1 *An optimal pair of physically-disjoint pair of paths in Network 2 is also an optimal pair of physically-disjoint paths in the original network, optimality being with respect to span mileage.*

Proof: Suppose physically-disjoint paths exist between a given pair of end points (A and Z) in Network 2. Consider the optimal pair, denoted by P2, in Network 2 (see Fig. 7a). By optimal pair is meant a pair of paths that is shortest with respect to span mileage. In the original network, this pair of paths has the same span structure, i.e., it remains physically-disjoint with the

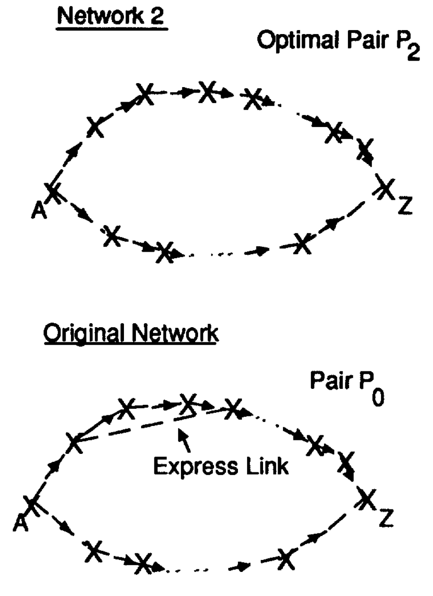

Fig. 7a The optimal pair of physically-disjoint paths, P_2, in Network 2 shown by the arcs it traverses; in the original network, it is shown as pair P_0.

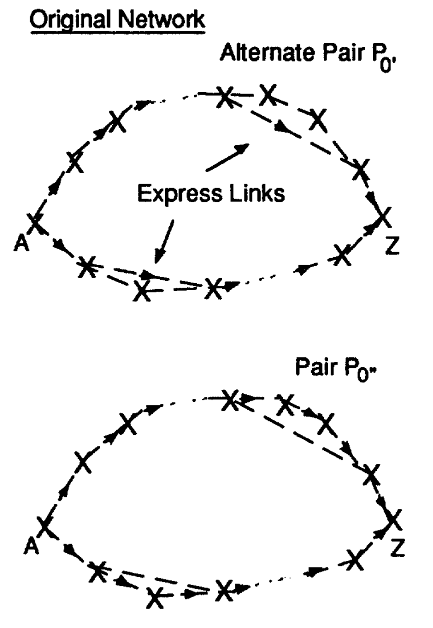

Fig. 7b Alternate pairs of physically-disjoint paths, $P_{0'}$ and $P_{0''}$; the latter is derived from the former.

same span mileage. We call this pair P_0 in the original network. Now, the following important question arises: Is the pair P_0 also an optimal pair of physically-disjoint paths in the original network? In other words, is there another physically-disjoint pair of paths in the original network that is shorter than the pair P_0 ? Let us suppose the answer is yes. We show this alternate pair, denoted by P_0', in Fig. 7b. Since express links are present in the original network, express link traversal is always a possibility. As a result, Fig. 7b depicts the paths traversing such links. Because of Assumption 1, traversal of the paths over express links can be replaced with traversals over the component links without changing the span mileage. Thus, this pair, denoted by P_0'' in Fig. 7b, is also an optimal pair in the original network. Clearly, this optimal pair P_0'' is also a solution in Network 2. We denote this pair of paths in Network 2 by P_2'. Now we invoke the following general theorem:

Theorem 2 *An optimal solution in a given network, if also a solution in a less flexible network, must necessarily be an optimal solution in the less flexible network.*

Thus, the pair of paths P_0'' in Fig. 7b, which is an optimal physically-disjoint pair of paths in the original network, is also an optimal pair of physically-disjoint paths in Network 2 (less flexible due to the absence of express links). But this pair P_2' is different (and shorter) than pair P_2 initially assumed to be optimal in Network 2. The contradiction implies that the alternate pair P_0' considered in the original network and shown in Fig. 7b, cannot be shorter than the pair P_0 (see Fig. 7a). Therefore, P_0 is indeed an optimal pair in the original network. This proves Theorem 1.

Thus, the problem of finding an optimal physically-disjoint pair of paths in the original network reduces to the problem of finding such a pair in Network 2, where express links are absent. Once such a pair is found in Network 2, the links present on the paths can be pieced together, whenever possible, to form express links that exist in the original network. The ensuing paths represent the desired pair of paths in the original network. Note that the step of piecing together the links to form express links, although not necessary, is desirable, since it reduces the total number of hops, and thus the dollar cost of provisioning the physically-disjoint circuits.

3.3 Fork Configuration

The network to be considered is Network 2, which is made up of the standard links (Fig. 2) and links corresponding to the fork configuration (Fig. 3), with express links absent.

Assumptions:

1) The fork configuration consists of two prongs (this assumption is only for the sake of simplicity in discussion). See Fig. 3a. We call this configuration theY

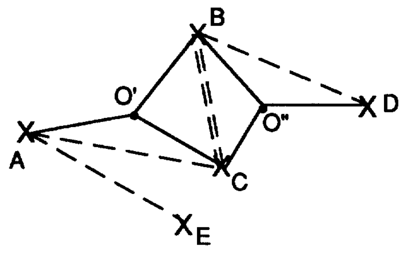

Fig. 8a Two adjacent Y configurations giving rise to multiple links between B and C

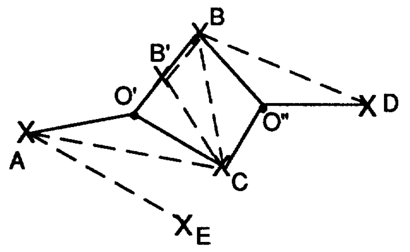

Fig. 8b Introduction of dummy node B' results in the disappearance of multiple links; link BC via junction O' is divided into two links , B'B and B'C.

configuration (as we shall see, the results for the Y configuration also hold for the fork configuration with more than 2 prongs).

2) There are no multiple links. If they do occur as in Fig. 8a, they can be eliminated by introducing a dummy node of degree 2 (see Fig. 8b). Degree of a node is the number of neighbors of the node. Dividing a given link into two links (as in Fig. 8b) via introduction of a dummy node of degree 2 does not affect routing in the network.

Lemma 1 *Constraint of node-disjointness at the link-level ensures absence of span-sharing in the region of the network away from the common end points of the two paths.*

Fig. 9 shows two paths encountering a Y configuration..The three different orientations of the Y configuration are considered [8]. In each of these cases, the constraint of node-disjointness at the link-level ensures that span OD common to links CD and BD is not traversed by both the paths simultaneously. The reason is that sharing of span OD requires each path to meet at node

D. Since the constraint of node disjointness

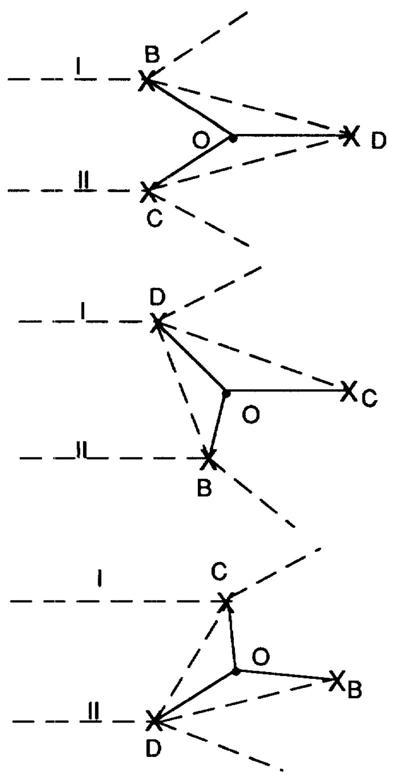

Fig. 9 Paths I and II encounter a Y configuration in a different orientation in each one of the above figures.

prevents the two paths to have a common node, the two paths cannot meet at node D; hence, sharing of span OD cannot occur.

At the end points, A and Z, between which a pair of physically-disjoint paths is sought, two cases arise:

1. No Y configuration is present.
2. Y configuration at end point A and/or Z is present.

3.3.1. No Y configuration at the end points We have already noted above that the constraint of node-disjointness ensures absence of span sharing away from the end points. If, in addition, Y configurations do not occur at the end points, an application of the SPNP algorithm at the link-level will guarantee a shortest physically-disjoint (node as well as span-disjoint) pair of paths.

3.3.2 Y configuration at the end points When Y configurations occur at the end points, they occur in two orientations. These orientations, denoted by Y1 and Y2, are displayed in Fig. 10.

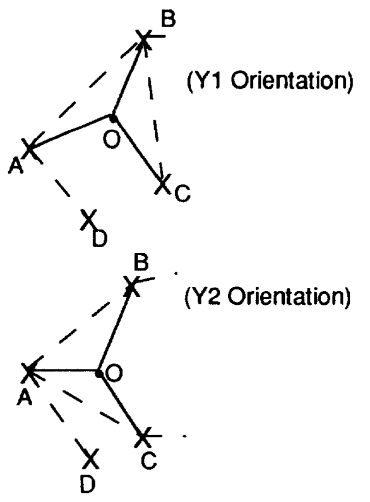

Fig. 10 Two orientations, Y1 and Y2, of the Y configuration at the end point A

Y1 Orientation Constraint of node-disjointness (at the link-level) ensures absence of span-sharing at node A; for example, if one path leaves node A via link AB, the other path necessarily leaves node A via link AD, and span OB is never common to the two paths originating from node A. Thus, the SPNP algorithm suffices.

Y2 Orientation In this case, node-disjointness at the link level does not guarantee span diversity. For example, the two paths found by the SPNP algorithm may traverse links AB and AC in which case span AO will be common to both the paths. Thus, the SPNP algorithm fails, and a new algorithm is needed.

Construction of a New Algorithm :

Consider the most general case where Y configurations occur at both the end points (A and Z). The shortest path between A and Z falls into one of three categories:

1. It does not pass through the Y configurations present at the end points.
2. It passes through a Y configuration at one of the end points.
3. It passes through a Y configuration at each end point.

Case 1

In this case, a second path (node-disjoint from the first) will not share any spans with the first path. Thus, the SPNP algorithm, when applied, yields an optimal pair of paths that is not only node-disjoint but also span-disjoint.

Case 2

Since the second path node-disjoint from the first can share a span (the stem of Y configuration), we need to modify the existing algorithm. In what follows, we first consider some network transformations.

Network 3 *In Network 2, replace the end point Y configuration (through which the shortest path passes) with a new Y configuration in which the stem of the Y configuration is of length 0, and the prongs of length equal to their respective parent links (see Fig. 11).*

Because this construction leaves the Y configuration link lengths unchanged, the network at the link level remains unchanged..Thus, the shortest path (in the network of links) remains invariant under this transformation (see Fig. 11). In other words, the shortest path in Network 3 will pass through the same link of the Y configuration, and the same component spans (of different individual, but same total length) are traversed. Furthermore, because the basic topology of the new Y configuration replacing the original Y configuration remains unchanged, Networks 2 and 3 are equivalent, validating the following lemma:

Lemma 2 *An optimal physically-disjoint pair of paths in Network 3 is also an optimal physically-disjoint pair of paths in Network 2.*

We now define another network transformation.

Network 4 *In Network 3, replace the span node of the new Y configuration through which the shortest path passes) with a network node so that spans of the Y configuration become links of the network (see Fig. 11).*

We observe that Network 4 behaves like a traditional network in the application of the SPNP algorithm for the given end points, A and Z. The SPNP algorithm when applied between the given end points produces the shortest node-disjoint pair of paths with no span overlap. We now state and prove the following theorem:

Theorem 3 *An optimal pair of node-disjoint paths in Network 4 transforms to an optimal pair of physically-disjoint paths in Network 2.*

Proof: We first show that the shortest path between A and Z in Network 3 remain invariant in Network 4.

An extra node in Network 4 makes the network less constrained (or more flexible) compared to Network 3, and leads to the possibility of the shortest path (between the end points, A and Z) in Network 4 being shorter than the shortest path between the same end points in Network 3. Let us assume that the shortest path found in Network 2 passes through node B, as shown in Fig. 11. As discussed earlier, this path will also be the shortest path in Network 3. Denoting it by SP0, it may be reexpressed as SP0 = AO + OB + shortest path from B to Z.

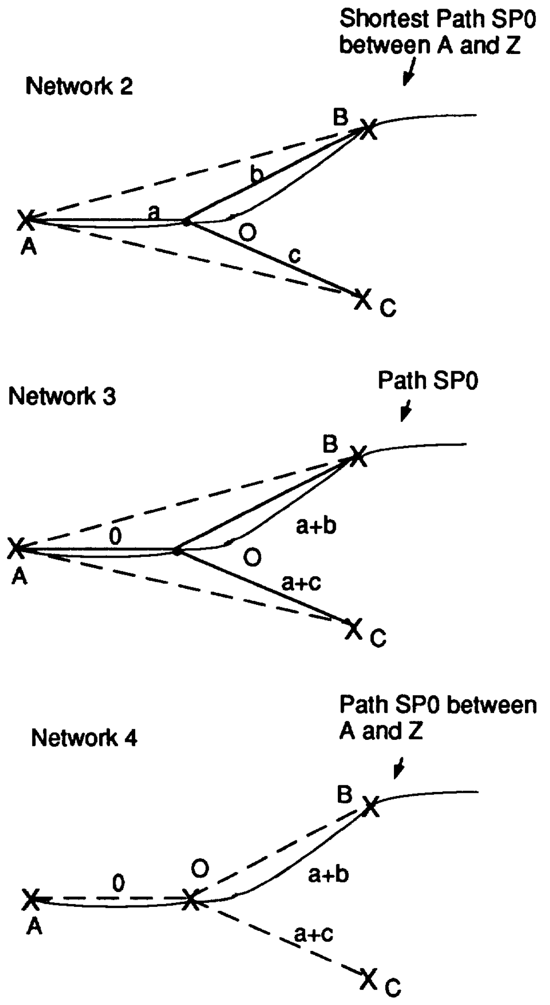

Fig. 11 Network transformations applied to Network 2 yield Network 3 and Network 4

Suppose the shortest path in Network 4 is indeed shorter than path SP0. Two possibilities arise for the new shorter path (see Fig. 12). It exits from

i) node C via path P, node B and node O (denote this shortest path by SP1).

ii) node B via path Q, node C and node O (denote this shortest path by SP2).

Case i)

SP1 = Path P + BO + OC + shortest path from C to Z=Path P +BO+ (AO +OC+shortest path from C to Z) (since AO=0) = Path P + BO + an alternate path between

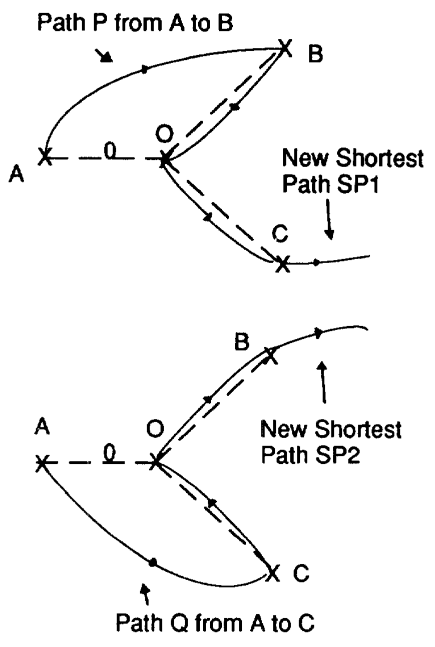

Fig. 12 Two new possible shortest paths, SP1 and SP2, in Network 4

A and Z in Network 3≥SP0, since Path P≥ 0, BO≥0, and any alternate path in Network 3≥ SP0. This result is a contradiction, implying case i) is invalid.

Case ii)

SP2 = Path Q + CO + OB + shortest path from B to Z = Path Q + CO + (AO + OB + shortest path from B to Z) (sinceAO=0)= Path Q+ CO + SP0 ≥ SP0, since Path Q≥ 0 and CO≥0. This result is a contradiction, ruling out case ii) also.

Since cases i and ii are both invalid, the shortest path in Network 4 cannot be shorter than the shortest path in Network 3. In other words, a shortest path determined in Network 3 will also be shortest in Network 4.

Because the shortest path (path SP0) in Network 4 passes through links AO and OB at end point A, and a non Y configuration at the other end (assumption in Case 2, which is presently being addressed), Network 4 behaves like a traditional network in the application of the SPNP algorithm between the given end points. In other words, application of SPNP algorithm, in conjunction with the shortest path found in Network 3, will result in the shortest pair of physically-disjoint paths in Network 4. Because the SPNP algorithm involves the possibility of interlacing of the second path with the first shortest path, two classes of solutions exist for the pair, paths I and II, obtained via Step 5 of the SPNP algorithm (see Section 3.1). At end point A,

1) Path I traverses links AO and OB; path II traverses a link other than AO (this is a case where no interlacing takes place or if it takes place, the second path exits from a node before node O (see Fig. 13)).

2) Path I traverses links AO and OC; path II traverses a link other than AO (this is a case where the second path interlaces with the shortest path and exits from node O (see Fig. 13)).

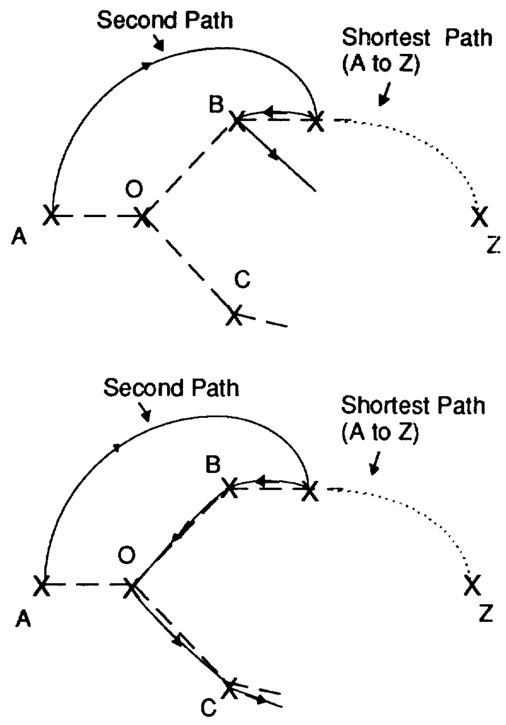

Fig. 13. The second path interlacing with the shortest path in Network 4 (which is the same as the shortest path in Network 3), and exiting from a node before node O in the upper figure, and from node O in the lower figure.

The above solutions in Network 4 become solutions in Network 3, since traversal of links AO and OB (or OC) by path I above transforms to a traversal of link AB (or AC) in Network 3. Thus, the optimal solutions obtained by using the SPNP algorithm in Network 4 are valid physically-disjoint solutions in Network 3. Invoking Theorem 2, these physically-disjoint solutions in Network 3 (less flexible compared to Network 4) are optimal solutions also. This result in conjunction with Lemma 2 proves Theorem 3.

Case 3

For the third case in which the shortest path passes through Y configuration at each of the end points, A and Z, the network transformation is performed at both the end points, i.e., the junction span nodes of the Y configurations are converted to network vertices with the stems of the Y configuration reduced to zero length, while assigning link lengths to the individual prongs. Similar proof can be given to show that optimal solutions given by the SPNP algorithm in the transformed network are also optimal physically-disjoint solutions in Network 2.

3.3.3 Algorithms for Y Configuration The results of Sec. 3.2.1 and Sec. 3.2.2 can now be combined into the following algorithm:

Algorithm 1 *In a network containing Y configurations (Network 2), a shortest pair of physically-disjoint paths between a given pair of nodes, A and Z, is obtained from the following steps:*

1. Find the shortest path from A to Z. .
2. Examine the end point spans of the shortest path found. If an endpoint span is the stem of a Y configuration, perform the following transformation:
   Replace the junction span node of the Y configuration by a node and alter the length of the stem of the Y configuration to zero, while increasing the length of the individual prongs to the length of the individual links, as in Fig. 11.
3. Modify the network (at the link-level) as in the SPNP algorithm (see Section 3.1).
4. Run the shortest path algorithm again (modified Dijkstra given in Appendex A).
5. Erase overlapping parts and coalesce split nodes, as in the SPNP algorithm, to obtain a shortest pair of node-disjoint paths. The pair obtained is also span-disjoint.
6. Transform back to Network 2 by replacing any added nodes in step 2 by span nodes and resetting the individual span lengths of the Y configuration to original lengths. The pair of paths obtained in Step 5 becomes an optimal pair in Network 2.

An Alternative to Algorithm 1 An alternative to Algorithm 1 can be derived from Fig. 13 and the discussion pertaining to Fig. 12 in the derivation of Algorithm 1. In Fig. 13, an exit of the second path from a node before node O leads to a pair of paths in which path I traverses link AB in Network 2, and path II any link other than AB and AC. Clearly, this optimal solution is also a solution in a network where link AC is blocked. Since the latter is a constrained network, by virtue of Theorem 2, this solution is an optimal solution in the constrained network (link AC blocked). Similarly, when the exit of the second path takes place from node O, we obtain a solution in which path I traverses link AC in Network 2, and path II traverses a link which is neither AB nor AC. By similar reasoning, this solution is an optimal solution in a network constrained by the blocking of link AB. Since it is not known a priori which of the two cases (path I traversing link AB or AC) actually occurs, one may run the SPNP algorithm twice, with link AB blocked once, and with link AC blocked the second time, and choose that result for the shortest pair for which the value of the pair length is the lower of the two. Similarly, traversal (by the first shortest path) of a Y2 configuration at the other end point, Z, would require two runs of the SPNP algorithm, with one of the two span-sharing links blocked in each run. When the shortest path traverses Y2 configurations present at both the end points, 2x2 (=4) runs of the SPNP algorithm would be required (see Fig. 14).

Algorithm 2 *This algorithm is an alternative to Algorithm 1, and consists of the following steps:*

1. Find the shortest path from A to Z.
2. Examine the end-point spans of the shortest path found.
3. Following cases arise:
   • If these spans are not shared spans, run the second shortest path algorithm as in the SPNP algorithm (Section 3.1); terminate.
   The pair of paths obtained after erasure of any overlapping parts will be the desired shortest pair with node-disjointness as well as span-disjointness.
   • If one of the end point spans is a shared span, run the SPNP algorithm twice, with one of the two span-sharing links blocked in one run, and the other span-sharing link blocked in the other run; if both the end points have shared spans, run the SPNP algorithm 2 x 2 (=4) times (see Fig. 14).
   Select the pair of paths for which the total length is the least.
   Perform erasure of overlapping parts, as in the SPNP algorithm.
   Resulting paths form the shortest node, span-disjoint pair between the given end points.

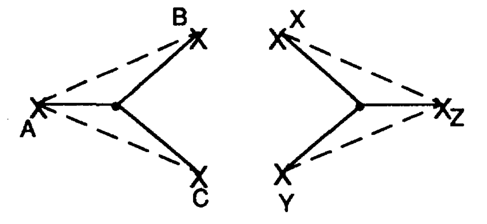

Fig. 14 If the shortest path between A and Z passes through shared spans at both ends, run the SPNP algorithm four times, blocking the following pairs of links, one at a time: (AB,XZ), (AB,YZ), (AC,XZ), and (AC,YZ)

Note that, in Algorithms 1 and 2, a tacit assumption is that the links are weighted by the total length of the component spans. Thus, the optimality is with respect to span mileage. It is important to mention here that these algorithms are general enough that weights corresponding to a different physical quantity such as dollar cost for transmission over the link, etc., can also be assigned to the links, in which case optimality is with respect to that physical quantity.

3.3.4 Algorithms for the general fork configuration Algorithms 1 and 2 specifically developed for a network in which Y configurations are present, extend easily to networks that contain fork configurations with more than two prongs (see Fig. 3(b)). Algorithm 1 is generalized to fork configurations with an arbitrary number of prongs by replacing the expression, Y configuration, with fork configuration everywhere in the statements of the algorithm. In Algorithm 2, the SPNP algorithm will have to be run as many times as n x m, where n is the number of links which share the span at one end point and m the number shared by the span at the other end. Clearly, Algorithm 1, which requires only two runs of the shortest path algorithm, is more efficient than Algorithm 2 which may require the repeated use of the SPNP algorithm.

3.4 Overall Algorithm

An algorithm for the overall fiber network obtains upon combining the results of Section 3.2 and Section 3.3. Since results of Section 3.2 are valid for optimality with respect to span mileage, the overall algorithm performs optimization with respect to span miles. In other words, the links are weighted by the total physical length of the spans comprising the links.

Algorithm 3: *When a network contains express links, and fork configurations, the shortest pair of physically-disjoint paths between a given pair of nodes is obtained from the following steps:*

1. Remove the express links in the network.
2. Perform steps of Algorithm 1 or 2.
3. Piece together links on the two paths found to form express links, if possible. These express links must belong to the set of express links removed in Step 1.

Step 3 is optional. Its utility lies in the fact that it reduces the total number of links in the two physically-disjoint paths found by the algorithm. The total number of span miles remains unaffected.

4. Application to Other Possible Configurations

We enumerate other types of configurations to which the developed algorithm can be applied:

1. *Multiple Fork Configurations:* These are fork configurations with more than one junction, and may occur frequently in fiber networks. An example with two junctions is depicted in Fig. 15. The traffic from node A to node B is via spans AO' and O'B, while the traffic from node A to node C (or D) traverses spans AO', O'O", O"C (or D). Although physically-different from the fork configuration of Fig. 3(b), from an algorithmic standpoint, it can be replaced with the standard fork

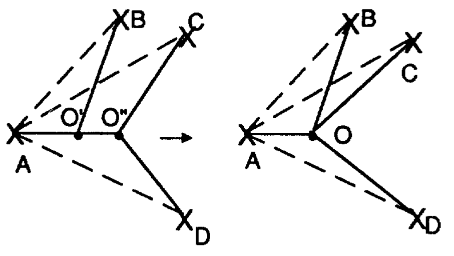

Fig. 15 Reduction to the fork configuration

configuration (see Fig. 15), provided the weights (or the lengths) of the individual links are preserved. In other words, the length of link AC (=AO+OC) in Fig. 15, should be the same as the length of link AC (=AO'+O'O"+O"C), and the same holds true for the other links AB and AD, provided the lengths are unchanged.

2. *Node-to-Node Connections via Junctions:* This type of a configuration is illustrated in Fig. 16. It is not

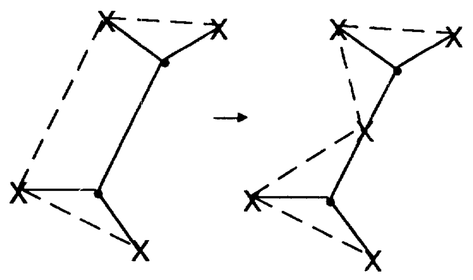

Fig. 16 Introduction of a dummy node reduces the configuration on the left to the basic fork configurations.

fundamentally different, since a judicious introduction of a dummy node reduces it to a pair of fork configurations.

3. *Triangle Configuration:* Fig. 17 shows the triangle configuration and its equivalent. Unlike the Y configuration, which has a missing link, all the nodes in the triangle configuration are connected by a pair of spans via junction O. This leads to the equivalence depicted in Fig. 17.

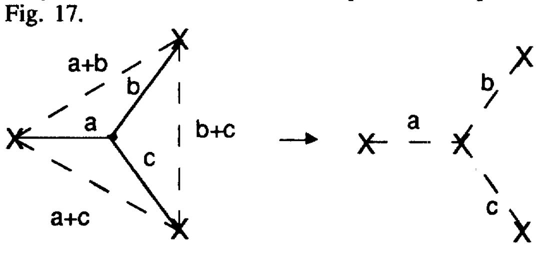

Fig. 17 Reduction of the triangle configuration

5. Summary

In this paper, we have considered networks, which are described by links (logical connections) and physical connections called spans. Two different links may, however, share the same span. We have considered different types of span-sharing (including two main ones) that may occur in actual telecommunication fiber networks, and have provided an algorithm for the shortest pair of physically-disjoint paths for a given pair of nodes. We give two versions of the algorithm, with the more efficient one requiring only two runs of an appropriate shortest path algorithm. Disjoint-path algorithms are useful in diverse provisioning of business services, and when computationally fast can also be employed in real-time diverse provisioning of services in a switched service environment. Additionally, they can also be utilized in a robust design of telecommunication networks based on the concept of traffic flow over two-disjoint paths for every pair of nodes in the network.

6. Acknowledgements

The problem originated in a project brought to the author's attention by Clayton Lockhart, and the author thanks him for his encouragement. Casimir Wierzynski, Hernando Garcia, Fung-Mei Chin, and Art O'Leary wrote the computer codes for the algorithms and their simulation.

References

1. J.W. Suurballe, "Disjoint Paths in a Network", Networks, 4 (1974) 125-145.
2. J.W. Suurballe and R. E. Tarjan, "A Quick Method for Finding Shortest Pairs of Disjoint Paths", Networks, 14 (1984) 325-336.
3. R. Bhandari, "Simpler Edge/Vertex-disjoint Shortest Pair Algorithms ", to be published.
4. For general information on NP-complete algorithms, see M.R. Garey and D.S. Johnson, *Computers and Intractability -A Guide to the Theory of NP-completeness*, W.H. Freeman, 1979.
5. L.R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton, University Press (1962).
6. E.W. Dijkstra, "A Note on Two Problems in Connexion with Networks," Numer. Math. 1(1959)269-271.
7. M. Gondran and M. Minoux, *Graphs and Algorithms*, John Wiley (1984).
8. Although the three possible orientations of a Y configuration are displayed in Fig. 9, only two are fundamentally different, the first and the second; the second and the third configurations are basically the same.

Appendix A

Modified Dijkstra Algorithm for Shortest Path from A to Z

The algorithm given below is a slight variant of the original Dijkstra algorithm [6,7]. It is different (in Step 3 below) in that it scans all the neighbors of the node selected (or "permanently " labeled) in Step 2.

Let d(i) denote the distance of node i from starting node A. Let P(i) denote its predecessor.

1. Start with
$d(A)=0$, $d(i)=l(A,i)$,if $i \in \Gamma_A$,
$= \infty$, otherwise
(Γ_i =set of first neighbor nodes of node i, $l(i,j)$=length of arc from node i to node j).
$P(i)=A \ \forall \ i \in \Gamma_A$.
Set $\overline{S} = \Gamma_A$.

2. Find $j \in \overline{S}$
such that $d(j)=\min d(i), i \in \overline{S}$.
Set $\overline{S} = \overline{S} - \{j\}$.
If j = Z (the terminal node), END; otherwise, go to 3.

3. $\forall i \in \Gamma_j$, if $d(j)+l(j,i) < d(i)$, set $d(i)=d(j)+l(j,i)$, $P(i)=j$ and $\overline{S}= \overline{S} \cup \{i\}$;
go to 2.

The algorithm, after initialized in step 1, alternates between steps 2 and 3. In each iteration, a node with least pathlength is selected from the set: $\overline{S}$ The algorithm searches by making one move at a time, and terminates when the node selected from the set $\overline{S}$ is Z.

In the original Dijkstra algorithm, when a node with the least path length is selected from the list of tentatively labeled nodes, the selected node is said to have been labeled "permanently", i.e, the shortest path length to that selected node from the given origin (the starting node A) has been found. No further scanning from any other node in the network can update the label of this node. In our application, because of the presence of negative arcs in the modified network (see Fig. 6b), rescanning can update the label of the previously selected (or "permanently" labeled) node. The algorithm given above permits such rescanning.