FROM THE PROCEEDINGS OF THE 6TH INT'L NETWORK PLANNING SYMPOSIUM, BUDAPEST, HUNGARY

NETWORKS' 94 (September, 1994)
"Planning for a customer-responsive network"

Shortest Pair of Physically-Disjoint Paths in Telecommunication Fiber Networks

Ramesh Bhandari

AT&T Bell Laboratories, Rm. 4L-423
Crawfords Corner Road, Holmdel, NJ 07733, USA
(908) 949-0693

Abstract

In this paper, we construct an algorithm for the shorterst pair of physically-disjoint paths between a given pair of nodes in telecommunication fiber networks. Such networks are complicated by the fact that a given span can be shared by more than one link. A link indicates connections between two nodes (central offices) in the network, while spans are the physical connections that comprise the facility (physical) network. Because of span-sharing possibility, new physical-disjointness algorithms are required. The algorithms we give for shortest physically-disjoint paths consider two commonly occurring span-sharing topologies in telecommunication fiber networks. Physically-disjoint (node as well as span-disjoint) paths improve reliability of a given network, while optimality implies reduced network costs.

1. Introduction

As fiber is increasingly deployed in telecommunication networks, reliability of a network is being called into question more than ever before. This is due to the fact that as more traffic is transported over the high bandwidth fiber network, any span (facility link) cut results in the loss of a large volume of traffic. One way to increase reliability of services provided to customers is via physical-diversity, i.e., by providing two physically-disjoint paths (node-disjoint as well as span-disjoint) to customers so that if one of the paths fails due to a span cut or node failure, the affected traffic can be routed on the other available physically-disjoint path. Clearly, when such pairs of disjoint paths are available, it is preferable to choose the shortest pair (sum of the two paths is a minimum), since an optimal pair for routing implies least cost, and subsequent benefit to the customers if part of the savings is passed to them.

Algorithms for optimal disjoint paths were first given by Suurballe [1,2]. However, they apply only to traditional graph-theoretic networks described by nodes and (logical) links, indicated by dashed lines in Fig. 1. A

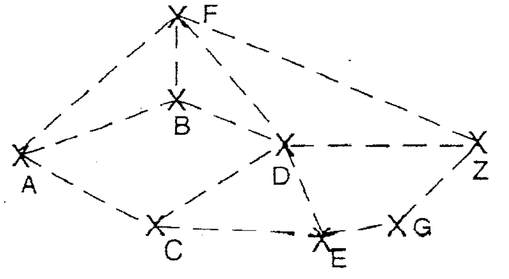

Fig. 1 A network of nodes and links

link such as AB in Fig. 1 simply indicates that a connection exists between nodes A and B; the actual path between nodes A and B may consist of a single span or a set of spans from A to B. Due to economic and practical considerations, telecommunication fiber networks may be so constructed that the physical paths of two different links in the network may share a span or a set of spans. Thus, the traditional algorithms [1,2] applicable only at the link level are inadequate because the node-disjoint paths found by them in such networks could consist of span-sharing links.

In this paper, we consider facility networks with two common types of deviations from the traditional graph-theoretic networks of nodes and links (Section 2), and provide an algorithm for the shortest pair of physically-disjoint paths between a given pair of nodes in the network (Section 3). To our knowledge, an algorithm for such type of a network has never been given before.

2. Network description

We consider the following network topologies different from the traditional network:

1) Fork Configuration : Refer to Fig. 2(a). Nodes A and B are connected physically (i.e., by fiber) via spans AO and OB; point O being a junction. The connection or link

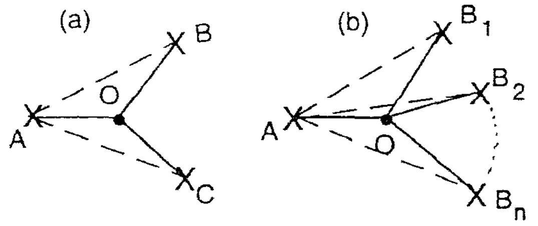

Fig.2(a) Fork configuration with two prongs (also called Y configuration in the text), and (b) fork configuration with n prongs

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 can 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. 2(a). Fig. 2(b) is a generalization of Fig. 2(a). It is a fork 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.

2) *Express Link:* In Fig. 3, the AB, BC, and CD connections (dashed lines) are via the physical spans (continuous line)AB, BC, and CD, respectively. In

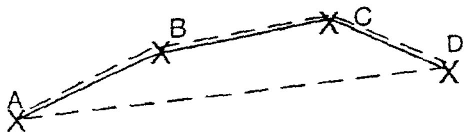

Fig. 3 Express link AD traversing three links, AB, BC, and CD.

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. In general, an express link may traverse n links, where n is an integer greater than 1; furthermore, the set of n links could include the links of fork configuration.

In what follows, we assume that the network under consideration is interspersed with network configurations depicted in Figs. 2 and 3. Fig. 4 is an example of such a

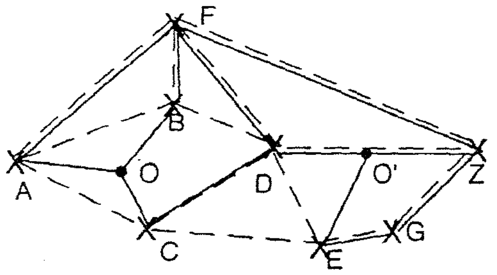

Fig. 4 A network with two fork configurations (junctions: O, O') and express link CE (span compostion: CD, DO', and O'E).

network. This network is constructed so that at the link level (dashed lines), it is identical to the traditional graph-theoretic network of Fig. 1. At the physical level, the network is different. There are 2 fork configurations with junction points: O, O', and an express link, CE. The problem is to find an optimal pair of physically-disjoint paths between a given pair of nodes in a network such as Fig. 4. Optimality refers to minimum total span (or fiber) miles for the pair of paths.

Clearly, the available shortest pair of node-disjoint paths algorithm (SPNP) algorithm [1,2], valid for the standard graph-theoretic network of Fig. 1, cannot be applied to the network of Fig. 4 at the span (physical) level, because the junction nodes O, O' are not true nodes (see Fig. 2 and discussion). On the other hand, the network at the li level (dashed lines) consists of true nodes, and the SPN algorithm can be applied to the network of Fig. 4 at t link-level. The links can be weighted by the sum of t length of the spans comprising the link. The SPN algorithm when applied at the link level in Fig. 4 giv the shortest pair of paths (with respect to span mile each path in the pair defined by a sequence of lin However, because the links of the two paths can ha overlapping spans, the paths found are not necessari physically-disjoint. For example, in Fig. 4, if the SPN algorithm finds the optimal pair of node-disjoint pat between nodes A and Z to be ABDZ and ACEGZ, spa AO and DO' would be common to the two paths four To avoid such span commonness, a new algorithm needed. In the next section, we not only give an algorit for finding physically-disjoint paths (node-disjoint as w as span-disjoint), but also ensure that the pair of pat obtained is shortest with respect to span mileage.

3. Algorithm for the shortest pair physically-disjoint paths

Our strategy in developing an algorithm for the optin pair of physically-disjoint paths in a network such as F 4 is to perform network transformations such that t SPNP algorithm can eventually be used at the link-lev without violating the requirement of physic disjointness. Therefore, as a first step, we describe t SPNP algorithm. The SPNP algorithm we give below [is an improved version of the Suurballe algorithm [1,2] that it does not require a general shortest path algorit like that of Ford's [4] or, alternatively a special canon transformation in order to facilitate the use of the popu Dijkstra algorithm [5,6]. Rather, the Dijkstra algorit here is altered slightly to circumvent the need for t usual canonic transformation. This modified Dijks algorithm is given in Appendix A.

3.1 Shortest pair of node-disjoint pat algorithm [3]

1. For the given pair of nodes under consideration, fi the shortest path using the shortest path algorithm giv in Appendix A. As an example, refer to Fig. 5a. Ea link in the graph is equivalent to two oppositely direct arcs of length equal to the link length.
2. Replace each link on the shortest path by an a directed towards the originating node (see Fig. 5b). Ma the length of the arcs negative.
3. Split each node (except the endpoint nodes, A and and nodes of degree 3 or less [7]) on the shortest path in two colocated subnodes. Join these nodes by an arc length zero, and direct it towards the starting no Replace external links connected to nodes on the short path by two arcs of the same and original length, a connected to the two subnodes as shown in Fig. 5c.
4. Run the shortest path algorithm (Appendix A) again.
5. Remove the zero length arcs; coalesce the subnode into their parent nodes. Replace the single arcs of th

shortest path with original links (of positive length).

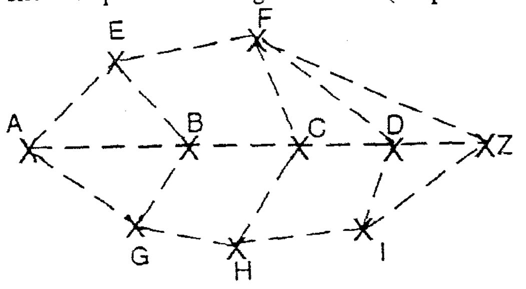

Fig. 5a Shortest path for the pair, A and Z, is assumed to be ABCDZ, with A the starting node in the shortest path algorithm.

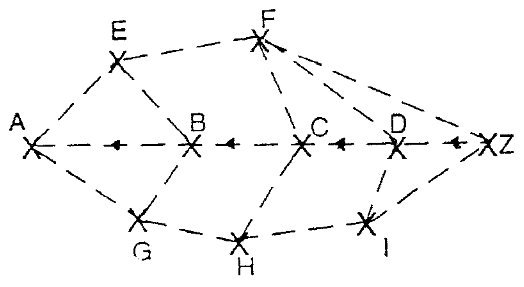

Fig. 5b Network with shortest path links replaced with negative arcs directed towards node A.

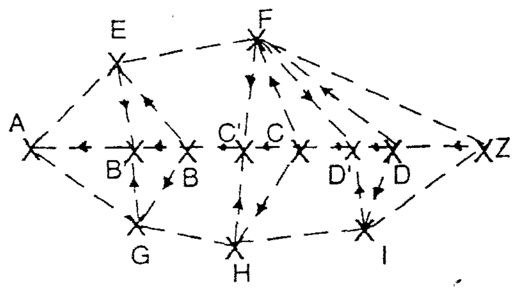

Fig. 5c Network modified by node-splitting.

Remove overlapping links of the two paths found to obtain the shortest pair of paths.

We now consider a network interspersed with express links and fork configurations, as in Fig. 4. As a first step, we will focus on the express links.

3.2 Express link

Fig. 6 shows an express link spanning n links. As Fig. 4 showed, the simultaneous presence of the express link and the links between nodes 0 and n in the network leaves open the possibility of link (and thus span) sharing by two node-disjoint paths determined by the SPNP algorithm. For example, one of the paths could enter node 0 and leave node n. The second path could enter node 1 and exit via any of the intermediate nodes 2,3,---,n-1, producing path overlaps of 1,---,n-2 links, respectively. A remedy for preventing overlapping links (and thus overlapping spans) while maintaining optimality for a pair of paths is provided through the following arguments:

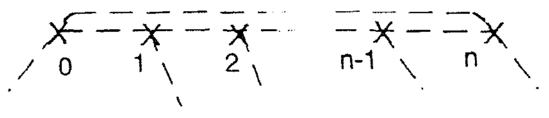

Fig. 6 Express link traversing n links.

Suppose path I enters node 0. Then, it can exit from any of the nodes 1,2,---,n. There are n possibilities in all, which can be divided into two types:

a) If the exit of path I takes place from any of the intermediate nodes 1,2,--n-1, then the presence of express link is redundant.

b) If exit of path I is to take place from node n, there are two choices :

1. Path I goes directly to node n in a single hop via the express link.

2. Path I goes through the intermediate nodes 2,3,---n-1, reaching node n in the final hop.

Both choices involve the same number of span miles and are therefore equivalent as far as span mileage is concerned. Choice 1 is preferable from the viewpoint of number of hops, since the number of hops is only 1. But its selection by the shortest path algorithm would leave open the possibility of overlapping links when the second path is determined. Consequently, any algorithm to be used should be constrained to select choice 2 in the above situation. This constraint can be easily enforced by simply deleting the express link from the network.

In view of a) and choice 2 in b) above, the following important rule emerges: *Eliminate all express links before running the SPNP algorithm.*

3.3 Fork Configuration

Assumptions:

1) The fork configuration consists of two prongs (this assumption is only for the sake of simplicity in discussion). See Fig. 2a. We call this configuration the Y configuration (as we shall see later, the results for the Y configuration also hold for the fork configuration with more than 2 prongs).

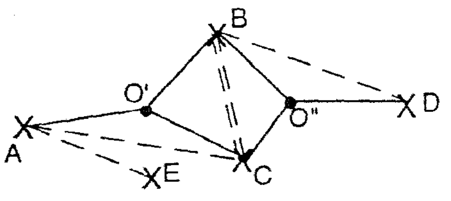

Fig. 7a Two adjacent Y configurations giving rise to multiple links between B and C.

2) There are no multiple links. If they do occur as in Fig. 7a, they can be eliminated by introducing a dummy node of degree 2 (see Fig. 7b),

As a first step, we show that the constraint of node-disjointness at the link-level ensures absence of span-sharing in the region of the network away from the common endpoints of the two paths.

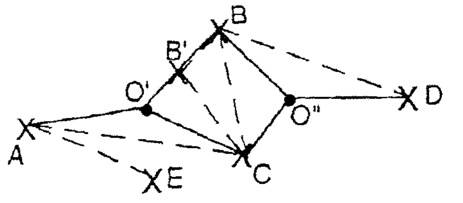

Fig. 7b Introduction of dummy node B' results in the disappearance of multiple links; link BC via junction O' is divided into two links , B'B and B'C.

Refer to Fig.8 which shows two paths encountering a Y configuration. The three different orientations of the Y configuration are considered [8]. In each of these cases,

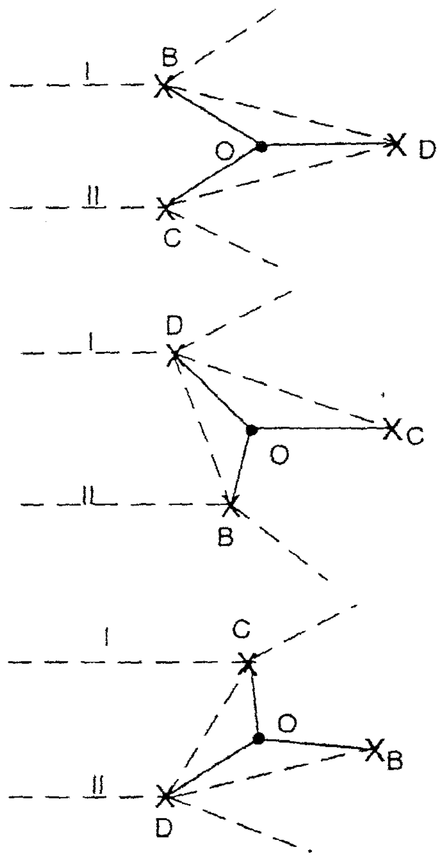

Fig. 8 Paths I and II encounter a Y configuration in a different orientation in each of the three figures.

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 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 now:
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 seen 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 endpoints, 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. 9.
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.

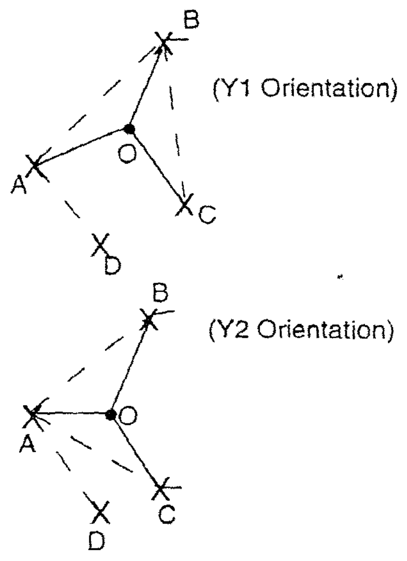

Fig. 9 Two orientations, Y1 and Y2, of the Y configuration at the end point A.

Y2 configuration: 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. For completeness, 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 the Y configuration at each end point.

Case 1:

In this case, a second path (node-disjoint from the first)

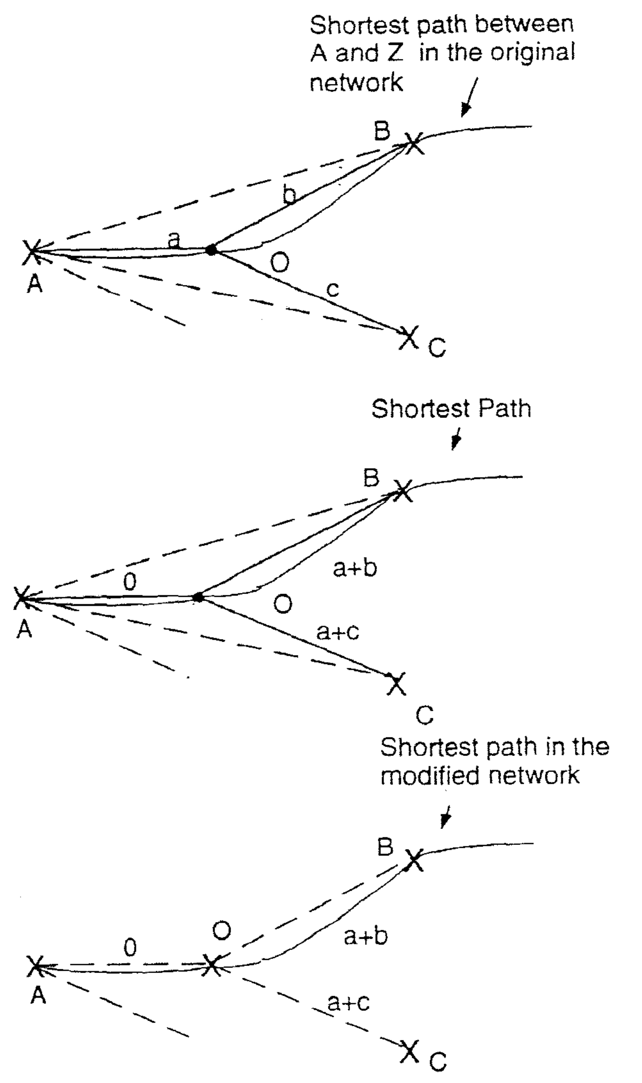
Shortest path between A and Z in the original network
B
b
a
A
O
c
C
Shortest Path
B
0
a+b
A
O
a+c
C
Shortest path in the modified network
B
0
O
a+b
A
a+c
C

Fig. 10 Shortest path originating from node A and passing through span AO of the Y configuration at end point A, and subsequent network modifications.

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:

Here the second path, even though node-disjoint from the first, can share a span (the stem of the Y configuration at the end point A). Fig. 10 shows the shortest path between A and Z, passing through link AB. Clearly, the second path determined by the SPNP algorithm could traverse link AC, thus sharing span AO in the process. We now state that this situation can be remedied by modifying the network in the following way:

a) 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. 10).

b) Replace the span node of the Y configuration (through which the shortest path passes) with a network node so that spans of Y configuration become links of the network (see Fig. 10).

Network modification b) ensures the absence of span commonness (in the original network) since the only span that can be common to the two paths has been transformed into a link. Furthermore, because the length of this link is 0 via modification a), the shortest path between nodes A and Z remains invariant in the modified network. This is a crucial requirement for convergence and ensuring the optimality of the disjoint paths when the SPNP algorithm is applied [9].

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 above network modification is performed at both the end points.

3.3.3 Algorithms for Y Configuration The results of Sec. 3.3.1 and Sec. 3.3.2 can now be combined into the following algorithm:

Algorithm 1: *In a network interspersed with Y configurations (express links absent), the 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 in the network of nodes and links.
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. 10.
3. Modify the network (at the link-level) as in the SPNP algorithm routine (see Sec. 3.1).
4. Run the shortest path algorithm again, using the algorithm in Appendix A.
5. Erase overlapping parts and coalesce split nodes, as in the SPNP algorithm, to obtain the shortest pair of node-disjoint paths. The pair obtained is also span-disjoint.
6. Transform back to the original network 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 the original network.

Note that 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, is also applicable to networks that contain fork configurations with more than two prongs (see Fig. 2(b)). This is due to the fact that the basic topology of the fork configuration is independent of the number of prongs. Thus, algorithm 1 generalizes 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.

3.4 Overall Algorithm

An algorithm for a general network like Fig. 4 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 2: *When a network contains express links, and fork configurations, the shortest pair of physically-disjoint pair of paths between a given pair of end points is obtained from the following steps:*

1 Remove the express links in the network.
2. Perform steps of Algorithm 1.
3. Piece together links on the two paths 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. Summary

In this paper, we have considered networks which are described by links and physical connections called spans. Two different links may, however, share the same span. We have considered two types of span-sharing links that seem to commonly occur in telecommunication networks, and have provided an algorithm for the shortest pair of physically-disjoint paths for a given pair of nodes. The developed algorithm, while requiring network modifications, uses two runs of a shortest path algorithm such as the one given in Appendix A, and is therefore computationally fast. Disjoint paths are useful in diverse provisioning of business services, and when computationally fast can be employed in real-time diverse provisioning of such services in a switched service environment. Additionally, they can 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.

5. Acknowledgement

The problem originated in a project brought to the author's attention by Clayton Lockhart, and the author thanks him for his encouragement.

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 Link-/Node-disjoint Shortest Pair Algorithms", to be published.
4. L.R. Ford and D. R. Fulkerson, *Flows in Networks* , Princeton, University Press (1962).
5. E.W. Dijkstra, "A Note on Two Problems in Connexion with Graphs", Numer. Math. 1(1959)269-271.
6. M. Gondran and M. Minoux, *Graphs and Algorithms* , John Wiley (1984)
7. Splitting nodes of degree 3 or less is redundant, a fact not recognized in [1,2]; clearly, not splitting such nodes reduces significantly the amount of network modification in sparse networks such as the telecommunication fiber networks.
8. Although the three possible orientations of a Y configuration are displayed in Fig. 8, only two are fundamentally different, the first and the second; the second and the third configurations are basically the same
9. Formal proofs are not being given here due to manuscript-length constraints.

Appendix A

Modified Dijkstra Algorithm for Shortest Path from A to Z

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 $i \in G_A$,
$= \infty$, otherwise
($G_i$ =set of first neighbor nodes of node i, $l(i,j)$=length of arc from node i to node j).
$P(i)=A \;\forall\, i \in G_A$.
Set $\overline{S} = G_A$.

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. $\forall i \in G_j$, 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.