Low-Connectivity Network Design on Series-Parallel Graphs

S. Raghavan
The Robert H. Smith School of Business, Van Munching Hall, University of Maryland, College Park, Maryland 20742

Network survivability is a critical issue for modern fiber-optic telecommunication networks. Networks with alternate routes between pairs of nodes permit users to communicate in the face of equipment failure. In this paper, we consider the following low-connectivity network design (LCND) problem: Given a graph $G = (N, E)$ and a connectivity requirement $d_i \in \{0, 1, 2\}$ for each node and edge costs c_e for each edge $e \in E$, design a minimum-cost network that contains at least $d_{st} = \min\{d_s, d_t\}$ disjoint paths between nodes s and t. We present linear-time algorithms for both node- and edge-connectivity versions of the problem on series-parallel graphs. Due to the sparsity of telecommunications networks, this algorithm can be applied to obtain partial solutions and decompositions that may be embedded in a heuristic solution procedure as well as exact solution algorithms for the problem on general graphs. © 2004 Wiley Periodicals, Inc.

Keywords: network design; series-parallel graphs; connectivity

1. INTRODUCTION

In this paper, we consider the low-connectivity network design (LCND) problem that arises as a fundamental problem in the practical design of telecommunication networks (see [7]). In this problem, given an undirected network $G = (N, E)$, with node set N, edge set E, a connectivity requirement $d_i \in \{0, 1, 2\}$ for each node $i \in N$, and edge costs c_e for each edge $e \in E$, we wish to design a minimum-cost network that has at least $d_{st} = \min\{d_s, d_t\}$ disjoint paths between each pair of nodes s and t. We consider both edge- and node-connectivity versions of the problem.* To distinguish between edge-connectivity and node-connectivity, we append N (e.g., $2N$) to denote node-connectivity requirements and append E to denote edge-connectivity requirements.

The LCND is NP-hard on general graphs, since it generalizes the Steiner tree problem, and has raised considerable interest among researchers. Grötschel et al. [12] described a cutting-plane approach for the problem on general graphs and were able to use this approach to solve some problems arising in local telephone companies. Magnanti and Raghavan [15] described a dual-ascent algorithm for the edge-connectivity version of the problem. This algorithm generates both a heuristic solution, as well as a lower bound on the optimal solution value for the problem. Several other researchers studied the problem on general graphs. See Grötschel et al. [13] and Raghavan and Magnanti [20] for recent surveys on this problem.

Telecommunication networks are typically sparse and planar [18]. Further, in many cases, parts of the network are series-parallel graphs. Consequently, researchers have studied the restriction of some versions of the LCND to series-parallel graphs. Wald and Colbourn [25] described a linear-time algorithm for the Steiner tree problem ($d_s \in \{0, 1\}$) on a series-parallel graph. Winter [26] described linear-time algorithms for the LCND on series-parallel graphs when $d_s \in \{0, 2N\}$ and $d_s \in \{0, 2E\}$ (i.e., no node has a connectivity requirement of 1). Since network design problems are often modeled as integer programs, researchers have investigated the polyhedral structure of the LCND problem on series-parallel graphs. Mahjoub [16] provided a complete description† of the 2-edge-connected spanning subgraph polytope on series-parallel graphs (i.e., the case where $d_s = 2E$ for all nodes). Baïou and Mahjoub [3] considered the case where $d_s \in \{0, 2E\}$ and provided a complete description of the Steiner 2-edge-connected polytope. Coullard et al. [8, 9] considered the node-connectivity version. In [8], they gave a complete description of the 2-node-connected spanning subgraph polytope on series-

* There are two edge-disjoint paths (respectively, node-disjoint paths) between a pair of nodes if the deletion of a single edge (respectively, single node) in the network does not disconnect them. A network is 2-edge-connected (respectively, 2-node-connected) if there are two edge-disjoint paths (respectively, node-disjoint paths) between every pair of nodes.

Received July 2001; accepted November 2003

Correspondence to: S. Raghavan; e-mail: raghavan@umd.edu

Contract grant sponsor: Bell Laboratories Martin Farber Internship

† A complete description is a linear inequality description of the convex hull of integer feasible solutions to the problem.

parallel graphs (i.e., the case where $d_s = 2N$ for all nodes), and in [9], they gave a complete description of the Steiner 2-node-connected polytope (i.e., the case with $d_s \in \{0, 2N\}$).

Series-parallel graphs are defined by a recursive construction process. Consequently, because of this structure, many NP-hard problems are polynomially solvable on them. However, the derivation of the polynomial-time algorithms are nontrivial and, indeed, very problem-specific. As we will show, the edge- and node-connectivity version of the LCND admit linear-time algorithms. The identification of a linear-time algorithm is important for several reasons: First, due to the sparsity of telecommunication networks, large portions of them are series-parallel. As a result, these algorithms may be used to obtain partial solutions on the series-parallel portions that could be used in a heuristic procedure for the problem. Second, decomposition using 2-separators is used to solve the problem on general graphs (see [12]). Using a decomposition procedure makes it computationally viable to solve large-scale problems via an exact procedure. In the decomposition procedure, a 2-separator (a pair of nodes whose deletion separates the graph into two or more components) is identified. Then, a series of smaller problems is solved on the components (in a particular order and with some minor modifications to these components), whose solutions can be pieced together to obtain a solution to the original problem. The algorithm for the LCND on series-parallel graphs is also based on decomposition using 2-separators. Thus, the decomposition can easily be applied to general graphs and combined with an exact solution approach (such as a cutting-plane or branch-and-cut algorithm) could result in the exact solution of large-scale LCND problems. Interestingly, as a consequence of our algorithm for the edge-connectivity version of the LCND, we observe that a decomposition used by Grötschel et al. [12] is incorrect.

The organization of the rest of the paper is as follows: In Section 2, we describe a well-known correspondence between series-parallel graphs and partial 2-trees. In doing so, we lay the foundations of our dynamic programming algorithms. In Section 3, we describe a linear-time algorithm for the node-connectivity version of the LCND problem, while in Section 4, we describe a linear-time algorithm for the edge-connectivity version of the LCND. Finally, in Section 5, we discuss how our results may be applied as a decomposition procedure to general graphs, thus providing a way to possibly solve large LCND problems on general graphs.

Notation: We use standard graph theory terms and notation as in the text by Bondy and Murty [5]. For clarity, we elaborate on the following terminology where we differ from [5]. A *trail* is a path that does not repeat edges. A *simple path* is a trail that does not repeat nodes. Let $G_1 = (N_1, E_1)$ and $G_2 = (N_2, E_2)$ be two graphs. The subgraph *induced* by G_1 on G_2 has edges $E_1 \cap E_2$ and nodes as the end points of the edges $E_1 \cap E_2$. A *node cut* is a subset N' of N such that $G - N'$ is disconnected. A *k-node cut* is a node cut of k elements. A *2-separator* is a 2-node cut.

2. SERIES-PARALLEL GRAPHS AND 2-TREES: ALGORITHM DESIGN OUTLINE

A graph $G = (N, E)$ is said to be series-parallel if and only if it contains no subgraph homeomorphic to K_4 (the complete graph on four nodes). An alternate, constructive, definition is as follows: A graph is series-parallel if each of its 2-node connected components[‡] can be constructed, starting with an edge, by the repeated application of the following two operations:

- Series construction: Replace an edge $e = (s, t)$ by a pair of edges (s, u) and (u, t).
- Parallel construction: Replace an edge $e = (s, t)$ by two parallel edges $e' = (s, t)$ and $\bar{e} = (s, t)$.

A closely related graph to a series-parallel graph is a 2-tree. It is defined via the following recursive construction procedure:

- K_3, the complete graph on three nodes is a 2-tree.
- Given a 2-tree and any edge (i, k) on the 2-tree, the graph obtained by adding a new node j and connecting it to nodes i and k via new edges (i, j) and (j, k) is a 2-tree.

Wald and Colbourn [25] characterized those networks, partial 2-trees, that can be completed to 2-trees by adding new edges. They showed that a network is a partial 2-tree if and only if it has no subgraph homeomorphic to K_4. Robertson and Seymour [22, 23] introduced a closely connected concept of treewidth for graphs. Van Leeuwen [24] showed that partial 2-trees are identical to graphs with treewidth at most 2. (This statement remains true when 2 is replaced by a general integer parameter $k > 0$, i.e., partial k-trees are identical to graphs with treewidth at most k, as proved in [24].) Thus, partial 2-trees are exactly series-parallel graphs, and they are identical to graphs with treewidth at most 2.

We now briefly sketch a linear-time procedure developed by Wald and Colbourn [25] and adapted by Winter [26] to complete a partial 2-tree to a 2-tree. In the process of completing a partial 2-tree to a 2-tree, we will make the cost of added edges L a sufficiently large number (setting $L = 1 + \sum_{e \in E} c_e$, where E is the set of edges in the original graph, is sufficient). By doing so, it is sufficient to focus our attention to solving the LCND on 2-trees. For example, if the cost of the solution obtained is L or greater, it implies that an edge not present in the original graph is in the solution and so the problem is infeasible.

Without loss of generality, we assume that the graph G is connected. Otherwise, the problem is either infeasible (if

[‡] A 2-node-connected component of a graph G is a maximal subgraph of G that is 2-node-connected.

nodes with connectivity requirements, i.e., nodes with $d_s \geq 1$, are in different connected components) or we can discard all components that do not contain nodes with connectivity requirements. Checking whether the graph is connected and identifying if all nodes with connectivity requirements lie in the same connected component can be done in linear time using depth first search (DFS) (see [10]).

The procedure to complete a partial 2-tree to a 2-tree (from [25, 26]) also identifies when the original graph is not a partial 2-tree. It first converts the connected graph to a 2-node connected graph. This is done using DFS in such a way that the partial 2-tree structure is maintained (if the original graph is a partial 2-tree). The procedure then converts the 2-node connected partial 2-tree to a 2-tree by adding edges to it. To determine the edges to be added, the nodes of G are scanned sequentially and those of degree 2 are placed on a stack. The following steps are then repeated upon a copy of G until the copy is reduced to K_3 or the stack is empty:

Let $H = G$ be a copy of graph G.

(1) Remove the top node j, from the stack. If the stack is empty, then G is not a partial 2-tree. STOP.
(2) Determine edges (i, j) and (j, k) incident to j in H.
(3) If there is no edge (i, k), add it to both H and to G with cost $c_{ik} = L$.
(4) Delete node j. $H = H\backslash j$. If $H = K_3$, G is a 2-tree. STOP.
(5) If the degree of node i or k in H is 2, place them on the stack.

The DFS procedure to transform a connected partial 2-tree to a 2-node connected partial 2-tree takes linear time. The procedure of converting a 2-node connected partial 2-tree to a 2-tree also takes linear time since each step deletes a node in the graph and the number of operations in each step is constant. Consequently, the procedure to complete a partial 2-tree to a 2-tree takes linear time.

The motivation for our dynamic programming algorithms comes from the recursive construction process of a 2-tree. We reverse the construction process and sequentially contract the graph. At each stage, we repeatedly eliminate nodes of degree 2 until the graph obtained is an edge. At any stage in the contraction process (including initially prior to contraction), let G_{ij} denote the graph represented by edge (i, j). In other words, G_{ij} represents the subgraph in the original 2-tree G that has been contracted onto edge (i, j) (Fig. 1 provides an example). During the contraction process, we keep track of state information for the subgraph G_{ij} represented by each edge (i, j). The state information describes solutions to certain problems (which can be different from the original problem) restricted to the subgraph G_{ij}.

To use the contraction process to devise linear-time algorithms, there are three requirements: First, the number of states that we associate with each subgraph is *finite* and *independent* of the number of nodes. Second, suppose that during the contraction process node j has degree 2 and is connected to nodes i and k via edges (i, j) and (k, j) prior to elimination of j. Then, the new state information for G_{ik} (i.e., after the elimination of j) must be computable *solely* from the state information for G_{ij}, G_{ik}, and G_{jk}. Finally, we should be able to deduce the solution to the problem from the state information of the edge that is left at the end of the contraction procedure.

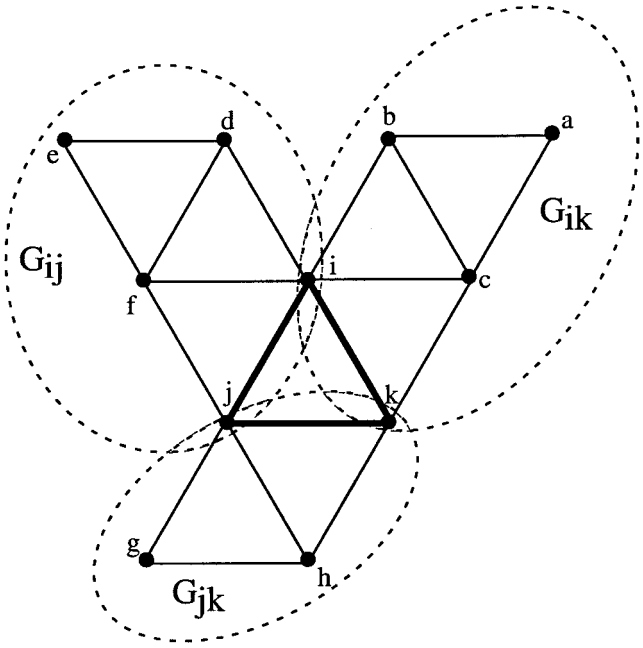

FIG. 1. Example of subgraph's of the original graph G represented by an edge in the contraction process. Suppose that the graph is contracted to (i, j), (j, k), (i, k). Then, $G_{ij} = \{(i, j), (i, f), (j, f), (d, f), (i, d), (d, e), (e, f)\}$, $G_{ik} = \{(i, k), (i, c), (k, c), (i, b), (b, c), (a, b), (a, c)\}$, and $G_{jk} = \{(j, k), (j, h), (h, k), (j, g), (g, h)\}$.

Although it is easy to state the algorithm design philosophy, and this has been formalized in several different ways (see [2, 4, 6]), it is a nontrivial task to determine the state information required for a given problem (and is very problem-specific). In the next few sections, we will develop linear-time algorithms for different versions of LCND. The basic algorithm can be described as follows: The state information is initialized for each edge of the graph G. The nodes of G are scanned sequentially and those of degree 2 are placed on a stack. The following steps are repeated until G is reduced to an edge:

(1) Remove the top node j from the stack.
(2) Determine edges (i, j) and (j, k) incident to j in G.
(3) Since G is a 2-tree, edge (i, k) exists. Compute the new state information for the graph $\tilde{G}_{ik} = G_{ik} \cup G_{ij} \cup G_{jk}$.
(4) Delete node j from G by setting $G = G\backslash j$. Update the state information associated with edge (i, k) in G (i.e., for graph G_{ik}) by setting the state information for G_{ik} equal to the state information for graph $\tilde{G}_{ik}$ computed in Step 3.
(5) If the degree of node i or k in G is 2, place them on the stack.

3. NODE-CONNECTIVITY REQUIREMENTS

In the case of node-connectivity requirements, let Y_1 be the set of nodes with $d_s = 1$ and let Y_{2N} be the set of nodes with $d_s = 2N$. We now motivate the graphical structures (states) that we compute in the course of the algorithm (i.e., the graphical structures we will compute in Step 3 of the algorithm described at the end of the previous section). At the end of the contraction process, an edge, say (i, k), represents the graph. The minimum-cost solution[§] to the problem either includes both nodes i and k, or includes node i but not k, or includes node k but not i, or excludes both nodes i and k. Thus, at the minimum, we must keep track of graphical structures corresponding to these four forms.

In our notation, the capital letters denote the graphical structure, and the small letters, their costs. For ease of exposition, we will use ∞ (instead of L) to denote the cost of edges that were added to complete the partial 2-tree to a 2-tree, as well as to denote the cost of infeasible solutions. However, in a computer implementation, as we have indicated earlier, ∞ may be replaced by a sufficiently large number L. The graphical structures, S_{ik}, T_{ik}, T_{ki}, and U_{ik}, corresponding to the four possible cases identified above are described below:

S_{ik} = minimum-cost network on G_{ik} that satisfies all the connectivity requirements on G_{ik}. It must include both nodes i and k.

T_{ik} = minimum-cost network on G_{ik} that satisfies all the connectivity requirements on G_{ik} and must include node i and must exclude node k. If k is required (i.e., $d_k >= 1$), then such a structure is infeasible, and by convention, $t_{ik} = \infty$.

U_{ik} = minimum-cost network on G_{ik} that satisfies all the connectivity requirements on G_{ik} but excludes nodes i and k. If either i or k are required, then, by convention, $u_{ik} = \infty$ (since U_{ik} is not feasible if either i or k are required).

Notice that the graphical structures are variants of the original problem restricted to the graph G_{ik}. In addition, the following graphical structures are necessary for the computations:

P_{ik} = minimum-cost network on G_{ik} containing both nodes i and k and such that for every Y_{2N} node there is a simple path (i.e., a path that does not repeat a node) from i to k through it and every Y_1 node is connected to i and to k.

Q_{ik} = minimum-cost network on G_{ik} comprising of two disjoint trees that include all nodes with nonzero connectivity requirements. One tree includes node i and the other tree includes node k. If $G_{ik}\backslash\{i, k\}$ has a Y_{2N} node, then, by convention, $q_{ik} = \infty$.

R_{ik} = minimum-cost network on G_{ik} comprising of two disjoint networks that include all nodes with nonzero connectivity requirements. One subnetwork includes node i and the other subnetwork includes node k. Further, all the Y_{2N} nodes belong to only one of the two disjoint subnetworks, and there must be two node-disjoint paths between every pair of Y_{2N} nodes. If this is not the case, for example, if i and $k \in Y_{2N}$, then $r_{ik} = \infty$.

To motivate the need for these structures, consider S_{ik}. Suppose that in Step 3 of the basic algorithm we need to compute $\tilde{S}_{ik}$ for the graph $\tilde{G}_{ik} = G_{ik} \cup G_{ij} \cup G_{jk}$. Consider the possible graphical structures induced by $\tilde{S}_{ik}$ on G_{ik}, G_{ij}, and G_{jk}, respectively. To compute $\tilde{S}_{ik}$, we need to have available the costs of these graphical structures (i.e., the possible ones $\tilde{S}_{ik}$ induces on G_{ik}, G_{ij}, and G_{jk}, respectively). $\tilde{S}_{ik}$ may induce a connected graph on G_{ik}, G_{ij}, and G_{jk}, respectively. In this case, the graphical structures $\tilde{S}_{ik}$ induces on G_{ik}, G_{ij}, and G_{jk} are P_{ik}, P_{ij}, and P_{jk}, respectively (see Lemma 2), motivating the need for P_{ik}. If $\tilde{S}_{ik}$ induces a graph that is not connected on any one of G_{ik}, G_{ij}, and G_{jk}, then, as we will show, all Y_{2N} nodes in $\tilde{G}_{ik}$ are in exactly one of G_{ik}, G_{ij}, and G_{jk}. Consequently, the disconnected subgraph induced by $\tilde{S}_{ik}$ either contains all Y_{2N} nodes, motivating R_{ik}, or it does not contain any Y_{2N} node (other than one of the two endpoints, say i or j, if the graphical structure is induced on G_{ij}), motivating Q_{ik}.

Finally, the following variables provide logical information to enable us determine whether a particular network configuration is feasible:

a_{ik} indicates whether $G_{ik}\backslash\{i\}$ contains at least one Y_{2N} node. If $G_{ik}\backslash\{i\}$ contains at least one Y_{2N} node, then $a_{ik} = \infty$ and is zero otherwise.

m_{ik} indicates whether any of the nodes in G_{ik} have a connectivity requirement. m_{ik} is ∞ if one of the nodes on G_{ik} has a connectivity requirement and is zero otherwise.

Notice that, except for T_{ik} and a_{ik}, all graphical structures and variables are symmetrical in the sense that $P_{ik} = P_{ki}$.

Initially, before the first contraction, the costs for the graphical structures on each edge $(i, k) \in E$ are initialized as follows:

$$s_{ik} = \begin{cases} \infty & \text{if } i \in Y_{2N} \text{ and } k \in Y_{2N} \\ c_{ik} & \text{otherwise} \end{cases}$$

$$t_{ik} = \begin{cases} \infty & \text{if } k \in Y_1 \cup Y_{2N} \\ 0 & \text{otherwise} \end{cases}$$

$$u_{ik} = \begin{cases} \infty & \text{if either } i \text{ or } k \in Y_1 \cup Y_{2N} \\ 0 & \text{otherwise} \end{cases}$$

$$p_{ik} = c_{ik}$$

$$q_{ik} = 0$$

[§] We consider the solution to be defined by the edges in the network. Thus, the solution does not contain nodes with degree 0.

$$r_{ik} = \begin{cases} \infty & \text{if } i \in Y_{2N} \text{ and } k \in Y_{2N} \\ 0 & \text{otherwise} \end{cases}$$

$$a_{ik} = \begin{cases} \infty & \text{if } k \in Y_{2N} \\ 0 & \text{otherwise} \end{cases}$$

$$m_{ik} = \begin{cases} \infty & \text{if } i \text{ or } k \in Y_1 \cup Y_{2N} \\ 0 & \text{otherwise.} \end{cases}$$

As the algorithm proceeds, let node j be the node of degree two being eliminated and let nodes i and k be the nodes adjacent to it. Then, the state information on $\tilde{G}_{ik} = G_{ik} \cup G_{ij} \cup G_{jk}$ is updated as described in the following recursive equations:

The Recursive Equations (Node-connectivity Case) (1)

$$\begin{aligned}\tilde{s}_{ik} = \min\{&p_{ij} + p_{jk} + p_{ik}, t_{ij} + t_{kj} + \min\{s_{ik} + a_{ij} + a_{kj}, p_{ik} \\ &+ a_{ik} + a_{jk}, p_{ik} + a_{ki} + a_{ji}\}, s_{ik} + p_{ij} + q_{jk} + a_{ij} + a_{kj}, p_{ik} \\ &+ s_{ij} + q_{jk} + a_{ik} + a_{jk}, p_{ik} + p_{ij} + r_{jk} + a_{ji} + a_{ki}, s_{ik} + p_{kj} \\ &+ q_{ij} + a_{ij} + a_{kj}, p_{ik} + p_{kj} + r_{ij} + a_{ik} + a_{jk}, p_{ik} + s_{kj} + q_{ij} \\ &+ a_{ji} + a_{ki}, r_{ik} + p_{ij} + p_{jk} + a_{ij} + a_{kj}, q_{ik} + s_{ij} + p_{jk} + a_{ik} \\ &+ a_{jk}, q_{ik} + p_{ij} + s_{jk} + a_{ji} + a_{ki}\}\end{aligned}$$

$$\begin{aligned}\tilde{t}_{ik} = \min\{&t_{ik} + t_{jk} + \min\{p_{ij} + a_{ij} + a_{kj}, s_{ij} + a_{ik} + a_{jk}, p_{ij} \\ &+ a_{ji} + a_{ki}\}, m_{jk} + t_{ik} + t_{ij} + \min\{a_{ij}, a_{ik}\}\}\end{aligned}$$

$$\begin{aligned}\tilde{u}_{ik} = \min\{&m_{ij} + m_{jk} + u_{ik}, m_{ik} + m_{jk} + u_{ij}, m_{ij} + m_{ik} \\ &+ u_{jk}, t_{ji} + t_{jk} + m_{ik} + \min\{a_{jk}, a_{ji}\}\}\end{aligned}$$

$$\begin{aligned}\tilde{p}_{ik} = \min\{&p_{ij} + p_{jk} + p_{ik}, p_{ik} + t_{ij} + t_{kj} + a_{ij} + a_{kj}, p_{ij} + p_{jk} \\ &+ q_{ik}, p_{ik} + p_{ij} + q_{jk} + a_{ij}, p_{ik} + p_{jk} + q_{ij} + a_{kj}\}\end{aligned}$$

$$\tilde{q}_{ik} = \min\{q_{ik} + a_{ij} + a_{kj} + \min\{t_{ij} + t_{kj}, p_{ij} + q_{jk}, p_{jk} + q_{ij}\}\}$$

$$\begin{aligned}\tilde{r}_{ik} = \min\{&t_{ij} + t_{kj} + \min\{r_{ik} + a_{ij} + a_{kj}, q_{ik} + a_{ik} + a_{jk}, q_{ik} \\ &+ a_{ki} + a_{ji}\}, r_{ik} + p_{ij} + q_{jk} + a_{ij} + a_{kj}, q_{ik} + s_{ij} + q_{jk} + a_{ik} \\ &+ a_{jk}, q_{ik} + p_{ij} + r_{jk} + a_{ji} + a_{ki}, r_{ik} + p_{jk} + q_{ij} + a_{ij} \\ &+ a_{kj}, q_{ik} + p_{jk} + r_{ij} + a_{jk} + a_{ik}, q_{ik} + s_{jk} + q_{ij} + a_{ji} + a_{ki}\}\end{aligned}$$

$$\tilde{a}_{ik} = a_{ij} + a_{ik} + a_{jk}$$

$$\tilde{m}_{ik} = m_{ij} + m_{ik} + m_{jk}.$$

Once we have reduced the graph to an edge [say (i, k)], the cost of the solution is $\min\{s_{ik}, t_{ik}, t_{ki}, u_{ik}\}$.

We now establish the correctness of the above equations:

Theorem 1. *The recursive equations* (1) *correctly compute the costs of the graphical structures for the node-connectivity case.*

Proof. The proof requires enumeration of all possible cases. We describe the cases for structure $\tilde{S}_{ik}$ in detail. The other cases are identified in the Appendix. Once we have identified all possible cases, it is easy to obtain the recursive equations.

We will consider the graphical structure $\tilde{S}_{ik}$. In doing so, we will also motivate the need for the structures P_{ik}, Q_{ik}, and R_{ik}. The following result, which follows immediately from the well-known Mengers' theorem [17], is useful in our discussion:

Lemma 2. *Let $\{i, j\}$ be a 2-separator, separating two nodes s and $t \in Y_{2N}$. If there are two node-disjoint paths between s and t, then one of the paths must include node i and the other path node j.*

We restrict our attention to the graph $\tilde{G}_{ik} = G_{ij} \cup G_{jk} \cup G_{ik}$. Suppose that all the Y_{2N} nodes are not contained entirely in one of the subgraphs G_{ij}, G_{jk}, and G_{ik}. In that case, there either exist two nodes s and t in Y_{2N} that have one of $\{i, j\}$, $\{j, k\}$, $\{i, k\}$ as 2-separators separating them or i, j, k are the only nodes in G_{ik} that belong to Y_{2N}. In both cases, using Lemma 2, it follows that each of the graphs induced by $\tilde{S}_{ik}$ on G_{ik}, G_{jk}, and G_{ij} is a connected graph that includes nodes i and k, nodes j and k, and nodes i and j, respectively. From this, it follows that the union of P_{ij}, P_{jk}, and P_{ik} provides us with a solution that satisfies the requirements of $\tilde{S}_{ik}$ at minimum cost. Notice that, in this case, $\tilde{s}_{ik} = p_{ij} + p_{jk} + p_{ik}$.

Now suppose that all the Y_{2N} nodes are contained entirely in one of the subgraphs G_{ij}, G_{jk}, and G_{ik}. Then, either each of the graphs induced by $\tilde{S}_{ik}$ on G_{ik}, G_{jk}, and G_{ij} is a connected graph that includes nodes i and k, nodes j and k, and nodes i and j, respectively, in which case $P_{ij} \cup P_{jk} \cup P_{ik}$ provides us with a solution that satisfies the requirements of $\tilde{S}_{ik}$ at minimum cost. *Or* there are 12 possible cases:

(1) j is not in $\tilde{S}_{ik}$.
   (a) All Y_{2N} nodes are contained in G_{ik}. Then, the union of S_{ik}, T_{ij}, and T_{kj} gives $\tilde{S}_{ik}$. In addition, a_{ij} and a_{kj} should be zero; otherwise, all Y_{2N} nodes are not contained in G_{ik}. In this case, $\tilde{s}_{ik} = t_{ij} + t_{kj} + s_{ik} + a_{ij} + a_{kj}$.
   (b) All Y_{2N} nodes are contained in G_{ij}. Then, the union of P_{ik}, T_{ij}, and T_{kj} gives $\tilde{S}_{ik}$. Note that if G_{ik} contains no Y_{2N} node then P_{ik} gives the optimal Steiner tree on the node set $Y_1 \cup \{i, k\}$ on G_{ik}. In addition, a_{ik} and a_{jk} must be zero; otherwise, all Y_{2N} nodes are not contained in G_{ij}. In this case, $\tilde{s}_{ik} = t_{ij} + t_{kj} + p_{ik} + a_{ik} + a_{jk}$.
   (c) All Y_{2N} nodes are contained in G_{jk}. Then, the union of P_{ik}, T_{ij}, and T_{kj} gives $\tilde{S}_{ik}$. In addition, a_{ki} and a_{ji} should be zero; otherwise, all Y_{2N} nodes

are not contained in G_{jk}. In this case, $\tilde{s}_{ik} = t_{ij} + t_{kj} + p_{ik} + a_{ki} + a_{ji}$.

(2) j is in $\tilde{S}_{ik}$, and j and k are not connected in the graph induced on G_{jk}.[#] The graph induced on G_{jk} is either R_{jk} or Q_{jk} (defined earlier) based on whether Y_{2N} nodes are contained in G_{jk}.
   (a) All Y_{2N} nodes are contained in G_{ik}. Then, the union of S_{ik}, P_{ij}, and Q_{jk} gives $\tilde{S}_{ik}$. To ensure feasibility, a_{ij} and a_{kj} must be zero. Thus, $\tilde{s}_{ik} = s_{ik} + p_{ij} + q_{jk} + a_{ij} + a_{kj}$.
   (b) All Y_{2N} nodes are contained in G_{ij}. Then, the union of P_{ik}, S_{ij}, and Q_{jk} gives $\tilde{S}_{ik}$. To ensure feasibility, a_{ik} and a_{jk} must be zero. Thus, $\tilde{s}_{ik} = p_{ik} + s_{ij} + q_{jk} + a_{ik} + a_{jk}$.
   (c) All Y_{2N} nodes are contained in G_{jk}. Then, the union of P_{ik}, P_{ij}, and R_{jk} gives $\tilde{S}_{ik}$. To ensure feasibility, a_{ji} and a_{ki} must be zero. Thus, $\tilde{s}_{ik} = p_{ik} + p_{ij} + r_{jk} + a_{ji} + a_{ki}$.

(3) j is in $\tilde{S}_{ik}$, and i and j are not connected in the graph induced on G_{ij}. These cases are symmetrical to cases 2(a)–2(c).

(4) j is in $\tilde{S}_{ik}$, and i and k are not connected in the graph induced on G_{ik}. These cases are symmetrical to cases 2(a)–2(c).

$\tilde{S}_{ik}$ is obtained by computing the structure with the lowest cost out of these 13 cases. Substituting the equations for these 13 cases gives the equation for $\tilde{s}_{ik}$ showing that $\tilde{s}_{ik}$ is computed correctly.

$\tilde{t}_{ik}$, $\tilde{u}_{ik}$, $\tilde{p}_{ik}$, $\tilde{q}_{ik}$, and $\tilde{r}_{ik}$ are computed correctly as shown by the cases in the Appendix.

$\tilde{a}_{ik}$ is computed correctly. $\tilde{a}_{ik}$ is ∞ if any node in $\tilde{G}_{ik}\backslash i$ is a Y_{2N} node. Thus, it is ∞ if any node in $G_{ik}\backslash i$, $G_{ij}\backslash i$, or G_{jk} is a Y_{2N} node. The equation $\tilde{a}_{ik} = a_{ik} + a_{ij} + a_{jk}$ expresses this condition.

$\tilde{m}_{ik}$ is computed correctly. $\tilde{m}_{ik}$ is ∞ if any node in $\tilde{G}_{ik}$ has a connectivity requirement. Thus, it is ∞ if any node in G_{ij}, G_{ik}, or G_{jk} has a connectivity requirement. The equation $\tilde{m}_{ik} = m_{ij} + m_{ik} + m_{jk}$ expresses this condition. ∎

Theorem 3. *The node-connectivity version of the LCND problem on a series-parallel graph can be solved in linear time.*

Proof. At each step, the algorithm performs a fixed number of operations. The number of steps is linear in the number of nodes. Thus, based on our preceding discussion, it follows that the solution to the node-connectivity version of the LCND problem can be computed in linear time on a series-parallel graph.

In our discussion, we restricted ourselves to obtaining the cost of the solution. It should be clear that by keeping track of the associated graphs for each structure the optimal network can also be obtained in linear time. ∎

[#] In the rest of this paper, especially in the Appendix, we will interchangeably use the term "the graph induced on G_{jk} is disconnected" to refer to the situation where nodes j and k are not connected in the graph induced on G_{jk}.

4. EDGE-CONNECTIVITY REQUIREMENTS

In the case of edge-connectivity requirements, let Y_1 be the set of nodes with $d_s = 1$ and let Y_{2E} be the set of nodes with $d_s = 2E$. The methodology is similar to that for the node-connectivity case, except that the graphical structures that we need to keep track of are more complicated. At the end of the contraction process, let the edge remaining, which represents the graph, be (i, k). Then, the minimum-cost solution to the problem either includes both nodes i and k, or includes node i but not node k, or includes node k but not node i, or excludes both node i and k. These possible structures are denoted as follows:

S_{ik} = minimum-cost network on G_{ik} that satisfies the requirements of all nodes on G_{ik} and must include nodes i and k.
T_{ik} = minimum-cost network on G_{ik} that satisfies the connectivity requirements on G_{ik} and must include i and exclude k. If k is required, then $t_{ik} = \infty$.
U_{ik} = minimum-cost network on G_{ik} that satisfies the connectivity requirements on G_{ik} and excludes nodes i and k. If either i or k have connectivity requirements, then, by convention, $u_{ik} = \infty$.

One of the most important differences between the two problems lies in the fact that the edge-connectivity version of Lemma 2 is not true. In other words, suppose that $\{i, j\}$ is a 2-separator separating two nodes s, $t \in Y_{2E}$ and there are two (or more) edge-disjoint paths between nodes s and t. Then, it is possible that all the edge-disjoint paths pass through node i but not node j or pass through node j but not node i. In the former case, node i must have at least two edge-disjoint paths to node s and at least two edge-disjoint paths to node t. Similarly, in the latter case, node j must have at least two edge-disjoint paths to node s and at least two edge-disjoint paths to node t. Consequently, we may observe the following: If all paths between two nodes s, $t \in Y_{2E}$ in a feasible network satisfying all the connectivity requirements pass through a node u, node u must have two edge-disjoint paths to node s, two edge-disjoint paths to node t, and, thus, two edge-disjoint paths to all nodes in Y_{2E}. It is as if node u has a $2E$ connectivity requirement imposed on it. This situation drastically increases the number of structures that we need to consider. Unlike the node-connectivity case, we need to consider structures where the endpoints have $2E$ connectivity requirements imposed on them (this corresponds to cases where two edge-disjoint paths pass through the same node). Consequently, the following S and T structures are also necessary:

S_{ik}^{ik} = minimum-cost network on G_{ik} that satisfies the requirements of all nodes on G_{ik} and, furthermore, a $2E$ requirement is imposed on nodes i and k.

S^i_{ik} = minimum-cost network on G_{ik} that satisfies the requirements of all nodes on G_{ik} and, furthermore, a $2E$ requirement is imposed on node i and node k is required to be connected.
S^k_{ik} = minimum-cost network on G_{ik} that satisfies the requirements of all nodes on G_{ik} and, furthermore, a $2E$ requirement is imposed on node k and node i is required to be connected.
T^i_{ik} = minimum-cost network on G_{ik} that satisfies the connectivity requirements on G_{ik}, excludes node k, and imposes a $2E$ requirement on node i. If k is required to be connected, then $t^i_{ik} = \infty$.

In addition, the following structures are necessary for the computations:

P_{ik} = minimum-cost network on G_{ik} such that, for every Y_{2E} node in G_{ik}, there is a trail (i.e., a path that does not repeat an edge) from i to k through it and every Y_1 node is connected to i and k.
Q^{ik}_{ik} = minimum-cost network on G_{ik} comprising of two disjoint networks on G_{ik} that include all nodes with nonzero connectivity requirements. One subnetwork includes node i and the other subnetwork includes node k, both of which have $2E$ requirements imposed on them. Further, the connectivity requirements within each disjoint network are satisfied.
Q^i_{ik} = minimum-cost network on G_{ik} comprising of two disjoint networks on G_{ik} that include all nodes with nonzero connectivity requirements. One subnetwork includes node i and the other subnetwork includes node k. Node i has a $2E$ requirement imposed on it, and all Y_{2E} nodes in $G_{ik}\backslash\{i, k\}$ are in the subnetwork containing node i. Further, the connectivity requirements within each disjoint network are satisfied. Note that if the subnetwork of Q^i_{ik} containing node k has a node with a $2E$ requirement other than node k then $q^i_{ik} = \infty$. This implies that the subnetwork of Q^i_{ik} containing node k is a tree or just node k.
Q^k_{ik} = minimum-cost network on G_{ik} comprising of two disjoint networks on G_{ik} that include all nodes with nonzero connectivity requirements. One subnetwork includes node i and the other subnetwork includes node k. Node k has a $2E$ requirement imposed on it, and all Y_{2E} nodes in $G_{ik}\backslash\{i, k\}$ are in the subnetwork containing node k. Further, the connectivity requirements within each disjoint subnetwork are satisfied.
Q_{ik} = minimum-cost network on G_{ik} comprising of two disjoint trees that include all nodes with nonzero connectivity requirements. One tree includes node i and the other tree includes node k. If $G_{ik}\backslash\{i, k\}$ has an Y_{2E} node, then, by convention, $q_{ik} = \infty$.
R_{ik} = minimum-cost network on G_{ik} comprising of two disjoint networks that include all nodes with nonzero connectivity requirements. One subnetwork includes node i and the other subnetwork includes node k. Further, all the Y_{2E} nodes belong to only one of the disjoint subnetworks, and there must be two edge-disjoint paths between every pair of Y_{2E} nodes. If this is not the case, for example, if i and $k \in Y_{2E}$, then $r_{ik} = \infty$.

Finally, the following variables provide information on feasibility:

b_{ik} indicates whether $G_{ik}\backslash\{i\}$ contains at least one Y_{2E} node. If $G_{ik}\backslash\{i\}$ contains at least one Y_{2E} node, then $b_{ik} = \infty$ and is zero otherwise.
m_{ik} indicates whether any of the nodes in G_{ik} have a connectivity requirement. m_{ik} is ∞ if any node on G_{ik} has a connectivity requirement and is zero otherwise.
w_k indicates whether node $k \in Y_{2E}$. w_k is ∞ if $k \in Y_{2E}$ and is zero otherwise.

Notice that except for T_{ik}, T^i_{ik}, and b_{ik} all graphical structures and variables are symmetrical in the sense that $P_{ik} = P_{ki}$ ($S^i_{ik} = S^i_{ki}$, $Q^i_{ik} = Q^i_{ki}$, and so on). We initialize the costs as follows:

$$p_{ik} = c_{ik}$$

$$s^{ik}_{ik} = \infty$$

$$s^i_{ik} = \begin{cases} \infty & \text{if } k \in Y_{2E} \\ c_{ik} & \text{otherwise} \end{cases}$$

$$s^k_{ik} = \begin{cases} \infty & \text{if } i \in Y_{2E} \\ c_{ik} & \text{otherwise} \end{cases}$$

$$s_{ik} = \begin{cases} \infty & \text{if } i \in Y_{2E} \text{ and } j \in Y_{2E} \\ c_{ik} & \text{otherwise} \end{cases}$$

$$u_{ik} = \begin{cases} \infty & \text{if either } i \text{ or } k \in Y_1 \cup Y_{2E} \\ 0 & \text{otherwise} \end{cases}$$

$$t^i_{ik} = \begin{cases} \infty & \text{if } k \in Y_1 \cup Y_{2E} \\ 0 & \text{otherwise} \end{cases}$$

$$t_{ik} = \begin{cases} \infty & \text{if } k \in Y_1 \cup Y_{2E} \\ 0 & \text{otherwise} \end{cases}$$

$$q^{ik}_{ik} = 0$$

$$q^i_{ik} = 0$$

$$q^k_{ik} = 0$$

$$q_{ik} = 0$$

$$r_{ik} = \begin{cases} \infty & \text{if } i \in Y_{2E} \text{ and } k \in Y_{2E} \\ 0 & \text{otherwise} \end{cases}$$

$$b_{ik} = \begin{cases} \infty & \text{if } k \in Y_{2E} \\ 0 & \text{otherwise} \end{cases}$$

$$m_{ik} = \begin{cases} \infty & \text{if } i \text{ or } k \in Y_1 \cup Y_{2E} \\ 0 & \text{otherwise.} \end{cases}$$

As the algorithm proceeds, the state information on $\tilde{G}_{ik} = G_{ik} \cup G_{ij} \cup G_{jk}$ is updated as described in the following recursive equations:

The Recursive Equations (Edge-connectivity Case) (2)

$$\begin{aligned}
\tilde{s}_{ik} = \min\{&p_{ij} + p_{jk} + p_{ik}, t_{ij} + t_{kj} + \min\{s_{ik} + b_{ij} + b_{kj}, p_{ik} \\
&+ b_{jk} + b_{ik}, p_{ik} + b_{ji} + b_{ki}\}, s^i_{ik} + t^i_{ij} + b_{kj} + t_{kj}, s^k_{ik} + t^k_{kj} \\
&+ b_{ij} + t_{ij}, s^{ik}_{ik} + t^i_{ij} + t^k_{kj}, s_{ik} + p_{ij} + q_{kj} + b_{ij} + b_{kj}, p_{ik} + s_{ij} \\
&+ q_{kj} + b_{ik} + b_{jk}, p_{ik} + p_{ij} + r_{jk} + b_{ji} + b_{ki}, s^i_{ik} + s^i_{ij} \\
&+ q_{kj}, s^k_{ik} + p_{ij} + b_{ij} + q^k_{kj}, s^j_{ij} + p_{ik} + q^j_{jk} + b_{ik}, s^i_{ik} + s^{ij}_{ij} \\
&+ q^j_{jk}, s^{ik}_{ik} + s^i_{ij} + q^k_{kj}, s^{ik}_{ik} + s^{ij}_{ij} + q^{jk}_{jk}, s_{ik} + p_{kj} + q_{ij} + b_{ij} \\
&+ b_{kj}, p_{ik} + s_{jk} + q_{ij} + b_{ji} + b_{ki}, p_{ik} + p_{jk} + r_{ij} + b_{ik} \\
&+ b_{jk}, s^k_{ik} + s^k_{kj} + q_{ij}, s^i_{ik} + p_{kj} + b_{kj} + q^i_{ij}, s^j_{jk} + p_{ki} + b_{ki} \\
&+ q^j_{ji}, s^k_{ik} + s^{kj}_{kj} + q^j_{ij}, s^{ik}_{ik} + s^k_{kj} + q^i_{ij}, s^{ik}_{ik} + s^{kj}_{kj} + q^{ij}_{ij}, p_{ij} + s_{jk} \\
&+ q_{ik} + b_{ji} + b_{ki}, s_{ij} + p_{jk} + q_{ik} + b_{ik} + b_{jk}, p_{ij} + p_{jk} + r_{ik} \\
&+ b_{ij} + b_{kj}, s^k_{kj} + p_{ji} + b_{ji} + q^k_{ki}, s^i_{ij} + p_{jk} + b_{jk} + q^i_{ik}, s^j_{ij} \\
&+ s^j_{jk} + q_{ik}, s^{ij}_{ij} + s^j_{jk} + q^i_{ik}, s^i_{ij} + s^{jk}_{jk} + q^k_{ik}, s^{ij}_{ij} + s^{jk}_{jk} + q^{ik}_{ik}\}
\end{aligned}$$

$$\begin{aligned}
\tilde{s}^{ik}_{ik} = \min\{&p_{ik} + p_{ij} + p_{jk}, s^{ik}_{ik} + t^i_{ij} + t^k_{kj}, s^{ik}_{ik} + s^i_{ij} + q^k_{kj}, s^{ik}_{ik} \\
&+ s^{ij}_{ij} + q^{jk}_{jk}, s^{ik}_{ik} + s^k_{jk} + q^i_{ij}, s^{ik}_{ik} + s^{jk}_{jk} + q^{ij}_{ij}, s^{ij}_{ij} + s^{jk}_{jk} + q^{ik}_{ik}\}
\end{aligned}$$

$$\begin{aligned}
\tilde{s}^i_{ik} = \min\{&p_{ij} + p_{jk} + p_{ik}, s^i_{ik} + b_{kj} + t_{kj} + t^i_{ij}, s^{ik}_{ik} + t^k_{kj} \\
&+ t^i_{ij}, s^i_{ik} + s^i_{ij} + q_{kj}, s^i_{ik} + s^{ij}_{ij} + q^j_{jk}, s^{ik}_{ik} + s^i_{ij} + q^k_{kj}, s^{ik}_{ik} + s^{ij}_{ij} \\
&+ q^{jk}_{jk}, s^i_{ik} + p_{kj} + b_{kj} + q^i_{ij}, s^{ik}_{ik} + s^k_{jk} + q^i_{ij}, s^{ik}_{ik} + s^{jk}_{jk} + q^{ij}_{ij}, s^i_{ij} \\
&+ p_{jk} + b_{jk} + q^i_{ik}, s^{ij}_{ij} + s^j_{jk} + q^i_{ik}, s^{ij}_{ij} + s^{jk}_{jk} + q^{ik}_{ik}\}
\end{aligned}$$

$$\begin{aligned}
\tilde{t}_{ik} = \min\{&t_{ik} + t_{ij} + m_{jk} + \min\{b_{ij}, b_{ik}\}, t^i_{ik} + t^i_{ij} + m_{jk}, t_{ik} \\
&+ t_{jk} + \min\{p_{ij} + b_{ij} + b_{kj}, s_{ij} + b_{ik} + b_{jk}, p_{ij} + b_{ji} + b_{ki}\}, t^i_{ik} \\
&+ s^i_{ij} + t_{jk} + b_{jk}, t_{ik} + s^j_{ij} + t^j_{jk} + b_{ik}, t^i_{ik} + s^{ij}_{ij} + t^j_{jk}\}
\end{aligned}$$

$$\tilde{t}^i_{ik} = \min\{t^i_{ik} + t^i_{ij} + m_{jk}, t^i_{ik} + s^i_{ij} + t_{jk} + b_{jk}, t^i_{ik} + s^{ij}_{ij} + t^j_{jk}\}$$

$$\begin{aligned}
\tilde{u}_{ik} = \min\{&u_{ik} + m_{ij} + m_{kj}, u_{ij} + m_{ik} + m_{jk}, u_{jk} + m_{ji} \\
&+ m_{ki}, t_{ji} + t_{jk} + m_{ik} + \min\{b_{jk}, b_{ji}\}, t^j_{ji} + t^j_{jk} + m_{ik}\}
\end{aligned}$$

$$\begin{aligned}
\tilde{p}_{ik} = \min\{&p_{ij} + p_{ik} + p_{jk}, p_{ik} + t^i_{ij} + t^k_{kj}, p_{ik} + s^i_{ij} + q^k_{kj}, p_{ik} \\
&+ s^{ij}_{ij} + q^{kj}_{kj}, p_{ik} + s^k_{kj} + q^i_{ij}, p_{ik} + s^{kj}_{kj} + q^{ij}_{ij}, p_{ij} + p_{jk} + q^{ik}_{ik}\}
\end{aligned}$$

$$\tilde{q}_{ik} = \min\{q_{ik} + b_{ij} + b_{kj} + \min\{t_{ij} + t_{kj}, p_{ij} + q_{jk}, p_{kj} + q_{ij}\}\}$$

$$\begin{aligned}
\tilde{q}^i_{ik} = \min\{&q^i_{ik} + t_{kj} + b_{kj} + t^i_{ij}, q^i_{ik} + p_{kj} + b_{kj} + q^i_{ij}, \\
&q^i_{ik} + s^i_{ij} + q_{jk}, q^i_{ik} + s^{ij}_{ij} + q^j_{jk}\}
\end{aligned}$$

$$\begin{aligned}
\tilde{q}^{ik}_{ik} = \min\{&q^{ik}_{ik} + t^i_{ij} + t^k_{kj}, q^{ik}_{ik} + s^i_{ij} + q^k_{jk}, q^{ik}_{ik} + s^{ij}_{ij} + q^{jk}_{jk}, \\
&q^{ik}_{ik} + s^k_{kj} + q^i_{ji}, q^{ik}_{ik} + s^{kj}_{kj} + q^{ji}_{ji}\}
\end{aligned}$$

$$\begin{aligned}
\tilde{r}_{ik} = \min\{&t_{ij} + t_{kj} + \min\{r_{ik} + b_{ij} + b_{kj}, q_{ik} + b_{ik} + b_{jk}, q_{ik} \\
&+ b_{ji} + b_{ki}\}, q^i_{ik} + t^i_{ij} + t_{kj} + b_{jk}, q^k_{ik} + t^k_{kj} + t_{ij} + b_{ji}, r_{ik} + q_{ij} \\
&+ p_{kj} + b_{ij} + b_{kj}, q_{ik} + r_{ij} + p_{kj} + b_{ik} + b_{jk}, q_{ik} + q_{ij} + s_{kj} \\
&+ b_{ji} + b_{ki}, q^i_{ik} + q^i_{ij} + p_{jk} + b_{kj} + b_{jk}, q^k_{ik} + q_{ij} + s^k_{kj} \\
&+ b_{ji}, q_{ik} + q^i_{ij} + s^j_{kj} + b_{ki}, q^k_{ik} + q^j_{ij} + s^{kj}_{kj} + w_i, r_{ik} + p_{ij} \\
&+ q_{jk} + b_{ij} + b_{kj}, q_{ik} + s_{ij} + q_{jk} + b_{ik} + b_{jk}, q_{ik} + p_{ij} + r_{jk} \\
&+ b_{ji} + b_{ki}, q^k_{ik} + p_{ij} + q^k_{jk} + b_{ij} + b_{ji}, q^i_{ik} + s^i_{ij} + q_{jk} \\
&+ b_{jk}, q_{ik} + s^j_{ij} + q^j_{jk} + b_{ik}, q^i_{ik} + s^{ij}_{ij} + q^j_{jk} + w_k\}
\end{aligned}$$

$$\tilde{b}_{ik} = b_{ij} + b_{jk} + b_{ik}$$

$$\tilde{m}_{ik} = m_{ik} + m_{ij} + m_{jk}.$$

As in the node-connectivity case, once we have reduced the graph to an edge, the cost of the solution is $\min\{s_{ik}, t_{ik}, t_{ki}, u_{ik}\}$.

We now prove the correctness of the above equations:

Theorem 4. *The recursive equations* (2) *correctly compute the costs of the graphical structures for the edge-connectivity case.*

Proof. The proof is a process of enumerating all 108 possible cases. We describe the cases for one graphical structure $\tilde{S}_{ik}$ in detail. The others are discussed in the Appendix. As before, we restrict our attention to the graph $\tilde{G}_{ik} = G_{ij} \cup G_{jk} \cup G_{ik}$.

There are 34 possible cases:

(1) Nodes i and j are connected in the network induced by $\tilde{S}_{ik}$ on G_{ij}, nodes i and k are connected in the network induced by $\tilde{S}_{ik}$ on G_{ik}, and nodes j and k are connected in the network induced by $\tilde{S}_{ik}$ on G_{jk} (i.e., none of the networks induced on G_{ij}, G_{ik}, and G_{jk} by $\tilde{S}_{ik}$ are disconnected). Then, it follows that $\tilde{S}_{ik} = P_{ik} \cup P_{ij} \cup P_{jk}$. In this case, $\tilde{s}_{ik} = p_{ij} + p_{jk} + p_{ik}$.

(2) Node j is not in $\tilde{S}_{ik}$. There are six subcases:
   (a) All the Y_{2E} nodes are contained within G_{ik}. Then, $\tilde{S}_{ik} = S_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility, we need b_{ij} and b_{kj} to be zero. Thus, $\tilde{s}_{ik} = t_{ij} + t_{kj} + s_{ik} + b_{ij} + b_{kj}$.
   (b) All the Y_{2E} nodes are contained within G_{ij}. Then, $\tilde{S}_{ik} = P_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility, we need b_{ik} and b_{jk} to be zero. Thus, $\tilde{s}_{ik} = t_{ij} + t_{kj} + p_{ik} + b_{jk} + b_{ik}$.
   (c) All the Y_{2E} nodes are contained within G_{jk}. Then, $\tilde{S}_{ik} = P_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility, we require

that b_{ji} and b_{ki} to be zero. Thus, $\tilde{s}_{ik} = t_{ij} + t_{kj} + p_{ik} + b_{ji} + b_{ki}$.

(d) All the Y_{2E} nodes are contained within $G_{ij} \cup G_{ik}$, and there is at least one Y_{2E} node within G_{ij} and at least one Y_{2E} node within G_{ik}, both distinct from i. This forces a $2E$ requirement on i. Thus, $\tilde{S}_{ik} = S^i_{ik} \cup T^i_{ij} \cup T_{kj}$. In addition, for feasibility, we require that b_{kj} be zero. Thus, $\tilde{s}_{ik} = s^i_{ik} + t^i_{ij} + t_{kj} + b_{kj}$.

(e) All the Y_{2E} nodes are contained within $G_{jk} \cup G_{ik}$, and there is at least one Y_{2E} node within G_{jk} and at least one Y_{2E} node within G_{ik}, both distinct from k. This forces a $2E$ requirement on k. Thus, $\tilde{S}_{ik} = S^k_{ik} \cup T_{ij} \cup T^k_{kj}$. For feasibility, we require that b_{ij} be zero. Thus, $\tilde{s}_{ik} = s^k_{ik} + t_{ij} + t^k_{kj} + b_{ij}$.

(f) G_{ij} contains at least one Y_{2E} node and G_{jk} contains at least one Y_{2E} node. This forces a $2E$ requirement on i and k. Thus, $\tilde{S}_{ik} = S^{ik}_{ik} \cup T^i_{ij} \cup T^k_{kj}$, and $\tilde{s}_{ik} = s^{ik}_{ik} + t^i_{ij} + t^k_{kj}$.

(3) Node j is in $\tilde{S}_{ik}$, and nodes j and k are not connected in the graph induced on G_{jk}. There are nine subcases:

(a) All the Y_{2E} nodes are contained within G_{ik}. Then, $\tilde{S}_{ik} = S_{ik} \cup P_{ij} \cup Q_{kj}$. In addition, for feasibility, we require that b_{ij} and b_{kj} be zero. Thus, $\tilde{s}_{ik} = s_{ik} + p_{ij} + q_{kj} + b_{ij} + b_{kj}$.

(b) All the Y_{2E} nodes are contained within G_{ij}. Then, $\tilde{S}_{ik} = P_{ik} \cup S_{ij} \cup Q_{kj}$. For feasibility, we require that b_{ik} and b_{jk} be zero. Thus, $\tilde{s}_{ik} = p_{ik} + s_{ij} + q_{kj} + b_{ik} + b_{jk}$.

(c) All the Y_{2E} nodes are contained within G_{jk}. Then, $\tilde{S}_{ik} = P_{ik} \cup P_{ij} \cup R_{jk}$. For feasibility, we require that b_{ji} and b_{ki} be zero. Thus, $\tilde{s}_{ik} = p_{ik} + p_{ij} + r_{jk} + b_{ji} + b_{ki}$.

(d) All the Y_{2E} nodes are contained within $G_{ij} \cup G_{ik}$, and there is at least one Y_{2E} node within G_{ij} and at least one Y_{2E} node within G_{ik}, both distinct from i. This forces a $2E$ requirement on i. Thus, $\tilde{S}_{ik} = S^i_{ik} \cup S^i_{ij} \cup Q_{jk}$. Note that no additional feasibility variables are required here since $q_{jk} = \infty$ if any nodes in $G_{jk} \backslash \{j, k\}$ are in Y_{2E}. Thus, $\tilde{s}_{ik} = s^i_{ik} + s^i_{ij} + q_{kj}$.

(e) All the Y_{2E} nodes are contained within G_{ik} and the component containing node k in the network induced by $\tilde{S}_{ik}$ on G_{jk} and each contains at least one Y_{2E} node distinct from k. This forces a $2E$ requirement on k. Then, $\tilde{S}_{ik} = S^k_{ik} \cup P_{ij} \cup Q^k_{kj}$. For feasibility, we require that b_{ij} be zero. Thus, $\tilde{s}_{ik} = s^k_{ik} + p_{ij} + q^k_{kj} + b_{ij}$.

(f) All the Y_{2E} nodes are contained within G_{ij} and the component containing node j in the network induced by $\tilde{S}_{ik}$ on G_{jk}, and each contains at least one Y_{2E} node distinct from j. This forces a $2E$ requirement on j. Then, $\tilde{S}_{ik} = P_{ik} \cup S^j_{ij} \cup Q^j_{jk}$. For feasibility, we require that b_{ik} be zero. Thus, $\tilde{s}_{ik} = p_{ik} + s^j_{ij} + q^j_{jk} + b_{ik}$.

(g) G_{ik} contains at least one Y_{2E} node, the component containing node j in the network induced by $\tilde{S}_{ik}$ on G_{jk} contains at least one Y_{2E} node, and the component containing node k in the network induced by $\tilde{S}_{ik}$ on G_{jk} does not contain any Y_{2E} node. This forces a $2E$ requirement on i and j. Then, $\tilde{S}_{ik} = S^i_{ik} \cup S^{ij}_{ij} \cup Q^j_{jk}$. Thus, $\tilde{s}_{ik} = s^i_{ik} + s^{ij}_{ij} + q^j_{jk}$.

(h) G_{ij} contains at least one Y_{2E} node, the component containing node k in the network induced by $\tilde{S}_{ik}$ on G_{jk} contains at least one Y_{2E} node, and the component containing node j in the network induced by $\tilde{S}_{ik}$ on G_{jk} does not contain any Y_{2E} node. This forces a $2E$ requirement on i and k. Then, $\tilde{S}_{ik} = S^{ik}_{ik} \cup S^i_{ij} \cup Q^k_{kj}$. Thus, $\tilde{s}_{ik} = s^{ik}_{ik} + s^i_{ij} + q^k_{kj}$.

(i) Both the component containing node j and the component containing node k in the disjoint network induced by $\tilde{S}_{ik}$ on G_{jk} contain at least one Y_{2E} node. This forces a $2E$ requirement on j, k, and i. Then, $\tilde{S}_{ik} = S^{ik}_{ik} \cup S^{ij}_{ij} \cup Q^{jk}_{jk}$. Thus, $\tilde{s}_{ik} = s^{ik}_{ik} + s^{ij}_{ij} + q^{jk}_{jk}$.

(4) Node j is in $\tilde{S}_{ik}$, and nodes i and j are not connected in the graph induced on G_{ij}. There are nine subcases symmetrical to subcases 3(a)–3(i).

(5) Node j is in $\tilde{S}_{ik}$ and nodes i and k are not connected in the graph induced on G_{ik}. There are nine subcases symmetrical to subcases 3(a)–3(i).

$\tilde{s}^{ik}_{ik}$, $\tilde{s}^i_{ik}$, $\tilde{t}_{ik}$, $\tilde{t}^i_{ik}$, $\tilde{u}_{ik}$, $\tilde{p}_{ik}$, $\tilde{q}^{ik}_{ik}$, $\tilde{q}^i_{ik}$, $\tilde{q}_{ik}$, and $\tilde{r}_{ik}$ are computed correctly (see Appendix).

$\tilde{b}_{ik}$ is computed correctly. Proof similar to $\tilde{a}_{ik}$ in Theorem 2.

$\tilde{m}_{ik}$ is computed correctly. Proof as in Theorem 2. ∎

Theorem 5. *The edge-connectivity version of the LCND problem on a series-parallel graph can be solved in linear time.*

Proof. Similar to Theorem 3. ∎

5. DECOMPOSITION PROCEDURE FOR GENERAL GRAPHS

We now turn our attention to more general graphs and discuss how some of the ideas in this paper may be applied to them.

We first consider the node-connectivity case. Using Lemma 2, we observe that if we find a 2-separator $\{i, j\}$ in a graph, which separates two nodes in Y_{2N}, then we may decompose the original problem into two smaller problems as follows: Consider the two connected graphs $G_1 = (N_1, E_1)$ and $G_2 = (N_2, E_2)$, each containing at least one node in Y_{2N} distinct from i and j, with $E_1 \cup E_2 = E$, $E_1 \cap E_2 = \phi$, $N_1 \cup N_2 = N$, and $N_1 \cap N_2 = \{i, j\}$. [These can easily be determined by deleting nodes i and j (the nodes in the 2-separator) from the graph.] Create $G'_1 = G_1 \cup (i, j)$ and $G'_2 = G_2 \cup (i, j)$. Give nodes i and j a node-connectivity requirement of 2, and set $c_{ij} = 0$, in both G'_1 and G'_2. Solve the LCND problem on each of the smaller graphs, and denote the solutions as G^*_1 and G^*_2, respectively. The solution to the LCND problem on the original graph is obtained as $G^*_1 \cup G^*_2 \backslash (i, j)$. This procedure can be generalized as follows: Consider the connected graphs $G_t = (N_t, E_t)$, $t = 1, \ldots, q$, each containing at least one node in Y_{2N} distinct from the nodes i and j of the 2-separator, with $\cup_{t=1,\ldots,q} E_t = E$, $\cup_{t=1,\ldots,q} N_t = N$, $E_r \cap$

$E_s = \phi$, and $N_r \cap N_s = \{i, j\}$ for any $r, s \in \{1, \ldots, q\}$ and $r \neq s$. Again, these may be determined by deleting nodes i and j in the 2-separator. For $t = 1, \ldots, q$, create $G'_t = G_t \cup (i, j)$, give nodes i and j a node-connectivity requirement of 2, and set $c_{ij} = 0$. Solve the LCND problem on each G_t and denote the solution as G^*_t. The solution to the LCND problem on the original graph is obtained as $\cup_{t=1, \ldots, q} G^*_t \backslash (i, j)$.

By repeating this decomposition procedure, either two problems at a time or multiple subproblems at a time, it is possible to decompose the original LCND problem into a series of smaller LCND problems. Grötschel et al. [12] took this approach (considering multiple subproblems at a time) to solve to optimality several problems that arise in telecommunications practice. Note that 2-separators in a graph may be found in linear time using an algorithm based on depth first search (see [14]).

We now turn our attention to the edge-connectivity case. Decompositions for this case are somewhat more involved since we do not have a counterpart to Lemma 2.[‖] Consequently, the decomposition procedures are more complex, requiring the solution of multiple smaller problems, and may not be as easy to apply (except for the simplest cases) in general graphs. For completeness, we discuss the decomposition:

As in the node-connectivity case, let $\{i, j\}$ be a 2-separator separating two nodes in Y_{2E}, and $G_1 = (N_1, E_1)$ and $G_2 = (N_2, E_2)$, the two connected graphs, each containing at least one node in Y_{2E} distinct from i and j, with $E_1 \cup E_2 = E$, $E_1 \cap E_2 = \phi$, $N_1 \cup N_2 = N$, and $N_1 \cap N_2 = \{i, j\}$. There are nine different cases to consider. We use the notation developed in Section 4, with superscripts 1 and 2, to denote the structure belongs to graphs G_1 or G_2 respectively:

(1) The networks induced on G_1 and G_2 by the solution are connected. Then, the solution is $P^1_{ij} \cup P^2_{ij}$.
(2) Nodes i and j are not connected in the network induced on G_2, and the network induced on G_1 is connected. There are three subcases:
   (a) Both the component containing node i and the component containing node j in the network induced on G_2 contain Y_{2E} nodes. Then, the solution is $S^{ij^1}_{ij} \cup Q^{ij^2}_{ij}$.
   (b) The component containing node j in the network induced on G_2 does not contain Y_{2E} nodes. Then, the solution is $S^{i^1}_{ij} \cup Q^{i^2}_{ij}$.
   (c) The component containing node i in the network induced on G_2 does not contain Y_{2E} nodes. Then, the solution is $S^{j^1}_{ij} \cup Q^{j^2}_{ij}$.
(3) Nodes i and j are not connected in the network induced on G_1, and the network induced on G_2 is connected. There are three subcases:
   (a) Both the component containing node i and the component containing node j in the network induced on G_1 contain Y_{2E} nodes. Then, the solution is $S^{ij^2}_{ij} \cup Q^{ij^1}_{ij}$.
   (b) The component containing node j in the network induced on G_1 does not contain Y_{2E} nodes. Then, the solution is $S^{i^2}_{ij} \cup Q^{i^1}_{ij}$.
   (c) The component containing node i in the network induced on G_1 does not contain Y_{2E} nodes. Then, the solution is $S^{j^2}_{ij} \cup Q^{j^1}_{ij}$.
(4) Node j is not in the solution. Then, the solution is $T^{i^1}_{ij} \cup T^{i^2}_{ij}$.
(5) Node i is not in the solution. Then, the solution is $T^{j^1}_{ij} \cup T^{j^2}_{ij}$.

We considered the decomposition procedure when 2-separators separate two nodes in Y_{2E}. In a similar fashion, it is easy to derive the decomposition procedure when a 2-separator separates a node in Y_1 from a node in Y_{2E}, but does not separate two nodes in Y_{2E}. In [12], Grötschel et al. described one such decomposition procedure that they claim can be applied to the edge-connectivity case of the LCND problem. We now show that this decomposition procedure is incorrect. In this decomposition, a 2-separator $\{i, j\}$ separates a node in Y_1 from a node in Y_{2E}, but does not separate two nodes in Y_{2E}. Let $G_1 = (N_1, E_1)$ and $G_2 = (N_2, E_2)$ be the two graphs with $E_1 \cup E_2 = E$, $E_1 \cap E_2 = \phi$, $N_1 \cup N_2 = N$, and $N_1 \cap N_2 = \{i, j\}$, with all Y_{2E} nodes in N_2 and with $N_1 \backslash \{i, j\}$ containing at least one node in Y_1 and no Y_{2E} nodes. Grötschel et al. claimed that the network induced on G_1 by the solution to the LCND problem can only take one of the four following forms:

(1) Two disjoint trees. One tree includes i and the other tree includes j.
(2) A tree that does not include node i.
(3) A tree that does not include node j.
(4) A tree that includes both nodes i and j.

They neglect the case where nodes i and j are not connected in the graph induced on G_2, forcing both edge-disjoint paths between nodes in Y_{2E} to go through G_1. Thus, the decomposition procedure discussed in Grötschel et al. is incorrect.

CONCLUDING REMARKS

In this paper, we have described linear-time algorithms for the low-connectivity network design problem on series-parallel graphs. Our research generalizes earlier work [25, 26] by considering 0, 1, and 2 connectivity requirements together (i.e., $d_s \in \{0, 1, 2\}$). Further, we show how these ideas may be applied to obtain decomposition procedures that can be applied to general graphs.

For many problems, the linear-time algorithms on partial 2-trees generalize to partial k-trees [1, 2, 4, 6]. In particular, Arnborg et al. [1] showed that all properties definable in monadic second-order logic (MSOL) with quantification over vertex and edge sets can be decided in linear time for

[‖] Similar decompositions to the node-connectivity case may be applied by finding two *edges* whose removal disconnects two nodes in Y_{2E}.

partial k-trees given the tree decomposition. They also considered problems involving counting or summing evaluations over sets definable in MSOL and showed that a large class of these problems can be solved in linear time for partial k-trees. Borie et al. [6] described a predicate calculus in which many graph problems can be expressed. They showed that any problem that can be expressed using this predicate calculus can be solved in linear time on any recursively constructed graph, once the decomposition tree needed to construct the recursively constructed graph is known.

We are not aware of a way of expressing the LCND problem in either of these two formats. Thus, in the case of the LCND problem, extending these results even to 3-trees seems particularly challenging. One of the prime difficulties in generalizing the approach (described in this paper) to 3-trees is the number of states, that is, graphical structures, and cases grow quite rapidly and substantially. Further, it is conceivable that the LCND problem itself is NP-hard on a k-tree (for some $k \geq 3$). An interesting and recent result by Nishizeki et al. [19] showed that the edge-disjoint paths problem is NP-complete on 2-trees (while it is polynomially solvable on 1-trees and outerplanar graphs). Another interesting result in an earlier paper by Richey and Parker [21] showed that a particular generalization of the Steiner tree problem that they call the multiple Steiner subgraph problem (this is actually a special case of the weighted Steiner tree packing problem introduced in [11]) is polynomially solvable on trees, but is NP-complete on series-parallel graphs.

APPENDIX

Hand-drawn figures for all cases are available from the author.

Listing of Cases for Theorem 1

Cases for $\tilde{T}_{ik}$. There are five cases for $\tilde{T}_{ik}$.

(1) j is included in $\tilde{T}_{ik}$.
   (a) All Y_{2N} nodes are in G_{ik}. Then, $\tilde{T}_{ik} = T_{ik} \cup P_{ij} \cup T_{jk}$. In addition, for feasibility, we require that a_{ij} and a_{kj} be zero.
   (b) All Y_{2N} nodes are in G_{ij}. Then, $\tilde{T}_{ik} = T_{ik} \cup S_{ij} \cup T_{jk}$. For feasibility, a_{ik} and a_{jk} must be zero.
   (c) All Y_{2N} nodes are in G_{jk}. Then, $\tilde{T}_{ik} = T_{ik} \cup P_{ij} \cup T_{jk}$. For feasibility, a_{ji} and a_{ki} must be zero.

(2) j is not in $\tilde{T}_{ik}$.
   (a) All Y_{2N} nodes are in G_{ik}. Then, $\tilde{T}_{ik} = T_{ik} \cup T_{ij}$. For feasibility, m_{jk} and a_{ij} must be zero.
   (b) All Y_{2N} nodes are in G_{ij}. Then, $\tilde{T}_{ik} = T_{ik} \cup T_{ij}$. For feasibility, m_{jk} and a_{ik} must be zero.

Cases for $\tilde{U}_{ik}$. There are five cases for $\tilde{U}_{ik}$.

(1) j is not in $\tilde{U}_{ik}$.
   (a) All Y_{2N} nodes are in G_{ik}. Then, $\tilde{U}_{ik} = U_{ik}$. For feasibility, m_{ij} and m_{jk} must be zero.
   (b) All Y_{2N} nodes are in G_{ij}. Then, $\tilde{U}_{ik} = U_{ij}$. For feasibility, m_{ik} and m_{jk} must be zero.
   (c) All Y_{2N} nodes are in G_{jk}. Then, $\tilde{U}_{ik} = U_{jk}$. For feasibility, m_{ij} and m_{ik} must be zero.

(2) j is included in $\tilde{U}_{ik}$.
   (a) All Y_{2N} nodes are in G_{ij}. Then, $\tilde{U}_{ik} = T_{ji} \cup T_{jk}$. For feasibility, m_{ik} and a_{jk} must be zero.
   (b) All Y_{2N} nodes are in G_{jk}. Then, $\tilde{U}_{ik} = T_{ji} \cup T_{jk}$. For feasibility, m_{ik} and a_{ji} must be zero.

Cases for $\tilde{P}_{ik}$. There are five cases for $\tilde{P}_{ik}$.

(1) None of the graphs induced on G_{ij}, G_{ik}, and G_{jk} is disconnected. Then, $\tilde{P}_{ik} = P_{ik} \cup P_{ij} \cup P_{jk}$.
(2) j is not included in $\tilde{P}_{ik}$. Then, $\tilde{P}_{ik} = P_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility, a_{ij} and a_{kj} must be zero.
(3) j is included and the graph induced on G_{ik} by $\tilde{P}_{ik}$ is disconnected. Then, $\tilde{P}_{ik} = P_{ij} \cup P_{jk} \cup Q_{ik}$.
(4) j is included and the graph induced on G_{jk} by $\tilde{P}_{ik}$ is disconnected. Then, $\tilde{P}_{ik} = P_{ik} \cup P_{ij} \cup Q_{jk}$. For feasibility, a_{ij} must be zero.
(5) j is included and the graph induced on G_{ij} by $\tilde{P}_{ik}$ is disconnected. Then, $\tilde{P}_{ik} = P_{ik} \cup P_{jk} \cup Q_{ij}$. For feasibility, a_{kj} must be zero.

Cases for $\tilde{Q}_{ik}$. There are three cases for $\tilde{Q}_{ik}$.

(1) j is not included in $\tilde{Q}_{ik}$. Then, $\tilde{Q}_{ik} = Q_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility, a_{ij} and a_{kj} must be zero.
(2) j is included in $\tilde{Q}_{ik}$ and the graph induced on G_{jk} is disconnected. Then, $\tilde{Q}_{ik} = Q_{ik} \cup P_{ij} \cup Q_{jk}$. For feasibility, a_{ij} and a_{kj} must be zero.
(3) j is included in $\tilde{Q}_{ik}$ and the graph induced on G_{ij} is disconnected. Then, $\tilde{Q}_{ik} = Q_{ik} \cup Q_{ij} \cup P_{jk}$. For feasibility, a_{ij} and a_{kj} must be zero.

Cases for $\tilde{R}_{ik}$. There are nine cases for $\tilde{R}_{ik}$.

(1) j is not in $\tilde{R}_{ik}$.
   (a) All Y_{2N} nodes are in G_{ik}. Then, $\tilde{R}_{ik} = R_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility, a_{ij} and a_{kj} must be zero.
   (b) All Y_{2N} nodes are in G_{ij}. Then, $\tilde{R}_{ik} = Q_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility, a_{ik} and a_{jk} must be zero.
   (c) All Y_{2N} nodes are in G_{jk}. Then, $\tilde{R}_{ik} = Q_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility, a_{ji} and a_{ki} must be zero.

(2) j is in $\tilde{R}_{ik}$ and the graph induced on G_{jk} is disconnected.
   (a) All Y_{2N} nodes are in G_{ik}. Then, $\tilde{R}_{ik} = R_{ik} \cup P_{ij} \cup Q_{jk}$. For feasibility, a_{ij} and a_{kj} must be zero.
   (b) All Y_{2N} nodes are in G_{ij}. Then, $\tilde{R}_{ik} = Q_{ik} \cup S_{ij} \cup Q_{jk}$. For feasibility, a_{ik} and a_{jk} must be zero.
   (c) All Y_{2N} nodes are in G_{jk}. Then, $\tilde{R}_{ik} = Q_{ik} \cup P_{ij} \cup R_{jk}$. For feasibility, a_{ji} and a_{ki} must be zero.

(3) j is in $\tilde{R}_{ik}$ and the graph induced on G_{ij} is disconnected. These cases are symmetrical to subcases 2(a)–2(c) for $\tilde{R}_{ik}$.

Listing of Cases for Theorem 4

Cases for $\tilde{S}_{ik}^{ik}$. There are seven cases for $\tilde{S}_{ik}^{ik}$.

(1) None of the graphs induced on G_{ij}, G_{ik}, and G_{jk} by $\tilde{S}_{ik}^{ik}$ are disconnected. Then, $\tilde{S}_{ik}^{ik} = P_{ik} \cup P_{ij} \cup P_{jk}$.
(2) Node j is not in $\tilde{S}_{ik}^{ik}$. Then, $\tilde{S}_{ik}^{ik} = S_{ik}^{ik} \cup T_{ij}^{i} \cup T_{kj}^{k}$.

(3) j is in $\tilde{S}_{ik}^{ik}$ and the graph induced on G_{jk} is disconnected.
   (a) The component containing node j in the network induced by $\tilde{S}_{ik}^{ik}$ on G_{jk} does not contain Y_{2E} nodes. Then, $\tilde{S}_{ik}^{ik} = S_{ik}^{ik} \cup S_{ij}^{i} \cup Q_{kj}^{k}$.
   (b) The component containing node j in the network induced by $\tilde{S}_{ik}^{ik}$ on G_{jk} contains Y_{2E} nodes. Then, $\tilde{S}_{ik}^{ik} = S_{ik}^{ik} \cup S_{ij}^{ij} \cup Q_{jk}^{jk}$.
(4) j is in $\tilde{S}_{ik}^{ik}$ and the graph induced on G_{ij} is disconnected. These cases are symmetrical to subcases 3(a) and 3(b) for $\tilde{S}_{ik}^{ik}$.
(5) j is in $\tilde{S}_{ik}^{ik}$ and the graph induced on G_{ik} is disconnected. The only way for i and k to be two-edge connected is through node j. Then, $\tilde{S}_{ik}^{ik} = S_{ij}^{ij} \cup S_{jk}^{jk} \cup Q_{ik}^{ik}$.

Cases for $\tilde{S}_{ik}^{i}$. There are 13 cases for $\tilde{S}_{ik}^{i}$.

(1) None of the graphs induced on G_{ij}, G_{ik}, and G_{jk} by $\tilde{S}_{ik}^{i}$ are disconnected. Then, $\tilde{S}_{ik}^{i} = P_{ik} \cup P_{ij} \cup P_{jk}$.
(2) Node j is not in $\tilde{S}_{ik}^{i}$.
   (a) G_{jk} does not contain Y_{2E} nodes. Then, $\tilde{S}_{ik}^{i} = S_{ik}^{i} \cup T_{ij}^{i} \cup T_{kj}$. In addition, for feasibility, we require that b_{kj} be zero.
   (b) G_{jk} contains Y_{2E} nodes. Then, $\tilde{S}_{ik}^{i} = S_{ik}^{ik} \cup T_{ij}^{i} \cup T_{kj}^{k}$.
(3) j is in $\tilde{S}_{ik}^{i}$ and the graph induced on G_{jk} is disconnected.
   (a) G_{jk} does not contain Y_{2E} nodes. Then, $\tilde{S}_{ik}^{i} = S_{ik}^{i} \cup S_{ij}^{i} \cup Q_{jk}$.
   (b) The component containing node j, but not the component containing node k, in the network induced by $\tilde{S}_{ik}^{i}$ on G_{jk} contains Y_{2E} nodes. Then, $\tilde{S}_{ik}^{i} = S_{ik}^{i} \cup S_{ij}^{ij} \cup Q_{jk}^{j}$.
   (c) The component containing node k, but not the component containing node j, in the network induced by $\tilde{S}_{ik}^{i}$ on G_{jk} contains Y_{2E} nodes. Then, $\tilde{S}_{ik}^{i} = S_{ik}^{ik} \cup S_{ij}^{i} \cup Q_{kj}^{k}$.
   (d) Both the component containing node j and the component containing node k in the network induced by $\tilde{S}_{ik}^{i}$ on G_{jk} contain Y_{2E} nodes. Then, $\tilde{S}_{ik}^{i} = S_{ik}^{ik} \cup S_{ij}^{ij} \cup Q_{jk}^{jk}$.
(4) j is in $\tilde{S}_{ik}^{i}$ and the graph induced on G_{ij} is disconnected.
   (a) G_{jk} and the component containing node j in the network induced by $\tilde{S}_{ik}^{i}$ on G_{ij} do not contain Y_{2E} nodes. Then, $\tilde{S}_{ik}^{i} = S_{ik}^{i} \cup P_{kj} \cup Q_{ij}^{i}$. For feasibility, we require that b_{kj} be zero.
   (b) G_{jk} contains Y_{2E} nodes, and the component containing node j in the network induced by $\tilde{S}_{ik}^{i}$ on G_{ij} does not contain Y_{2E} nodes. Then, $\tilde{S}_{ik}^{i} = S_{ik}^{ik} \cup S_{jk}^{k} \cup Q_{ij}^{i}$.
   (c) The component containing node j in the network induced by $\tilde{S}_{ik}^{i}$ on G_{ij} contains Y_{2E} nodes. Then, $\tilde{S}_{ik}^{i} = S_{ik}^{ik} \cup S_{jk}^{jk} \cup Q_{ij}^{ij}$.
(5) j is in $\tilde{S}_{ik}^{i}$ and the graph induced on G_{ik} is disconnected. These cases are symmetrical to subcases 4(a)–4(c) for $\tilde{S}_{ik}^{i}$.

Cases for $\tilde{T}_{ik}$. There are nine cases for $\tilde{T}_{ik}$.

(1) j is not included in $\tilde{T}_{ik}$.
   (a) All the Y_{2E} nodes are contained within G_{ik}. Then, $\tilde{T}_{ik} = T_{ik} \cup T_{ij}$. For feasibility, m_{jk} and b_{ij} must be zero.
   (b) All the Y_{2E} nodes are contained within G_{ij}. Then, $\tilde{T}_{ik} = T_{ik} \cup T_{ij}$. For feasibility, m_{jk} and b_{ik} must be zero.
   (c) All the Y_{2E} nodes are contained within $G_{ij} \cup G_{ik}$, and there is at least one Y_{2E} node within G_{ij} and at least one Y_{2E} node within G_{ik}, both distinct from i. Then, $\tilde{T}_{ik} = T_{ik}^{i} \cup T_{ij}^{i}$. For feasibility, m_{jk} must be zero.
(2) j is included in $\tilde{T}_{ik}$.
   (a) All the Y_{2E} nodes are contained within G_{ik}. Then, $\tilde{T}_{ik} = T_{ik} \cup P_{ij} \cup T_{jk}$. For feasibility, b_{ij} and b_{kj} must be zero.
   (b) All the Y_{2E} nodes are contained within G_{ij}. Then, $\tilde{T}_{ik} = T_{ik} \cup S_{ij} \cup T_{jk}$. For feasibility, b_{ik} and b_{jk} must be zero.
   (c) All the Y_{2E} nodes are contained within G_{jk}. Then, $\tilde{T}_{ik} = T_{ik} \cup P_{ij} \cup T_{jk}$. For feasibility, b_{ji} and b_{ki} must be zero.
   (d) All the Y_{2E} nodes are contained within $G_{ij} \cup G_{ik}$, and there is at least one Y_{2E} node within G_{ij} and at least one Y_{2E} node within G_{ik}, both distinct from i. Then, $\tilde{T}_{ik} = T_{ik}^{i} \cup S_{ij}^{i} \cup T_{jk}$. For feasibility, b_{jk} must be zero.
   (e) All the Y_{2E} nodes are contained within $G_{ij} \cup G_{jk}$, and there is at least one Y_{2E} node within G_{ij} and at least one Y_{2E} node within G_{jk}, both distinct from j. Then, $\tilde{T}_{ik} = T_{ik} \cup S_{ij}^{j} \cup T_{jk}^{j}$. For feasibility, b_{ik} must be zero.
   (f) Both G_{ik} and G_{jk} contain Y_{2E} nodes. Then, $\tilde{T}_{ik} = T_{ik}^{i} \cup S_{ij}^{ij} \cup T_{jk}^{j}$.

Cases for $\tilde{T}_{ik}^{i}$. There are three cases for $\tilde{T}_{ik}^{i}$.

(1) j is not included in $\tilde{T}_{ik}^{i}$. Then, $\tilde{T}_{ik}^{i} = T_{ik}^{i} \cup T_{ij}^{i}$. For feasibility, m_{jk} must be zero.
(2) j is included in $\tilde{T}_{ik}^{i}$.
   (a) G_{jk} does not contain Y_{2E} nodes. Then, $\tilde{T}_{ik}^{i} = T_{ik}^{i} \cup S_{ij}^{i} \cup T_{jk}$. For feasibility, b_{jk} must be zero.
   (b) G_{jk} contains Y_{2E} nodes. Then, $\tilde{T}_{ik}^{i} = T_{ik}^{i} \cup S_{ij}^{ij} \cup T_{jk}^{j}$.

Cases for $\tilde{U}_{ik}$. There are six cases for $\tilde{U}_{ik}$.

(1) j is not in $\tilde{U}_{ik}$.
   (a) All Y_{2E} nodes are contained within G_{ik}. Then, $\tilde{U}_{ik} = U_{ik}$. For feasibility, m_{ij} and m_{kj} must be zero.
   (b) All Y_{2E} nodes are contained within G_{ij}. Then, $\tilde{U}_{ik} = U_{ij}$. For feasibility, m_{ik} and m_{jk} must be zero.
   (c) All Y_{2E} nodes are contained within G_{jk}. Then, $\tilde{U}_{ik} = U_{jk}$. For feasibility, m_{ji} and m_{ki} must be zero.
(2) j is in $\tilde{U}_{ik}$.
   (a) All Y_{2E} nodes are contained within G_{ij}. Then, $\tilde{U}_{ik} = T_{ji} \cup T_{jk}$. For feasibility, m_{ik} and b_{jk} must be zero.
   (b) All Y_{2E} nodes are contained within G_{jk}. Then, $\tilde{U}_{ik} = T_{ji} \cup T_{jk}$. For feasibility, m_{ik} and b_{ji} must be zero.
   (c) All Y_{2E} nodes are contained within $G_{ij} \cup G_{jk}$, and there is at least one Y_{2E} node within G_{ij} and at least one Y_{2E} node within G_{jk}, both distinct from j. Then, $\tilde{U}_{ik} = T_{ji}^{j} \cup T_{jk}^{j}$. For feasibility, m_{ik} must be zero.

Cases for $\tilde{P}_{ik}$. There are seven cases for $\tilde{P}_{ik}$.

(1) None of the networks induced by $\tilde{P}_{ik}$ on G_{ik}, G_{ij}, and G_{jk} is disconnected. Then, $\tilde{P}_{ik} = P_{ik} \cup P_{ij} \cup P_{jk}$.
(2) j is not in $\tilde{P}_{ik}$. Then, $\tilde{P}_{ik} = P_{ik} \cup T^i_{ij} \cup T^k_{kj}$.
(3) j is in $\tilde{P}_{ik}$ and the network induced on G_{jk} is disconnected.
   (a) No Y_{2E} nodes are present on the component containing node j in the network induced by $\tilde{P}_{ik}$ on G_{jk}. Then, $\tilde{P}_{ik} = P_{ik} \cup S^i_{ij} \cup Q^k_{kj}$.
   (b) Y_{2E} nodes are present on the component containing node j in the network induced by $\tilde{P}_{ik}$ on G_{jk}. Then, $\tilde{P}_{ik} = P_{ik} \cup S^{ij}_{ij} \cup Q^{kj}_{kj}$.
(4) j is in $\tilde{P}_{ik}$ and the network induced on G_{ij} is disconnected. These cases are symmetrical to subcases 3(a) and 3(b) for $\tilde{P}_{ik}$.
(5) j is in $\tilde{P}_{ik}$ and the network induced on G_{ik} is disconnected. Then, $\tilde{P}_{ik} = P_{ij} \cup P_{jk} \cup Q^{ik}_{ik}$.

Cases for $\tilde{Q}_{ik}$. There are three cases for $\tilde{Q}_{ik}$.

(1) j is not in $\tilde{Q}_{ik}$. Then, $\tilde{Q}_{ik} = Q_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility, b_{ij} and b_{kj} must be zero.
(2) j is in $\tilde{Q}_{ik}$, and the graph induced by $\tilde{Q}_{ik}$ on G_{jk} is disconnected. Then, $\tilde{Q}_{ik} = Q_{ik} \cup P_{ij} \cup Q_{jk}$. For feasibility, b_{ij} and b_{kj} must be zero.
(3) j is in $\tilde{Q}_{ik}$, and the graph induced by $\tilde{Q}_{ik}$ on G_{ij} is disconnected. Then, $\tilde{Q}_{ik} = Q_{ik} \cup Q_{ij} \cup P_{kj}$. For feasibility, b_{ij} and b_{kj} must be zero.

Cases for $\tilde{Q}^i_{ik}$. There are four cases for $\tilde{Q}^i_{ik}$.

(1) j is not in $\tilde{Q}^i_{ik}$. Then, $\tilde{Q}^i_{ik} = Q^i_{ik} \cup T^i_{ij} \cup T_{kj}$. For feasibility, b_{kj} must be zero.
(2) j is in $\tilde{Q}^i_{ik}$, and the graph induced on G_{ij} is disconnected. Then, $\tilde{Q}^i_{ik} = Q^i_{ik} \cup Q^i_{ij} \cup P_{kj}$. For feasibility, b_{kj} must be zero.
(3) j is in $\tilde{Q}^i_{ik}$, and the graph induced on G_{jk} is disconnected.
   (a) No Y_{2E} nodes are present in the component containing node j in the network induced by $\tilde{Q}^i_{ik}$ on G_{jk}. Then, $\tilde{Q}^i_{ik} = Q^i_{ik} \cup S^i_{ij} \cup Q_{jk}$.
   (b) Y_{2E} nodes are present in the component containing node j in the network induced by $\tilde{Q}^i_{ik}$ on G_{jk}. Then, $\tilde{Q}^i_{ik} = Q^i_{ik} \cup S^{ij}_{ij} \cup Q^j_{jk}$.

Cases for $\tilde{Q}^{ik}_{ik}$. There are five cases for $\tilde{Q}^{ik}_{ik}$.

(1) j is not in $\tilde{Q}^{ik}_{ik}$. Then, $\tilde{Q}^{ik}_{ik} = Q^{ik}_{ik} \cup T^i_{ij} \cup T^k_{kj}$.
(2) j is in $\tilde{Q}^{ik}_{ik}$ and the network induced on G_{jk} is disconnected.
   (a) No Y_{2E} nodes are present in the component containing node j in the network induced by $\tilde{Q}^{ik}_{ik}$ on G_{jk}. Then, $\tilde{Q}^{ik}_{ik} = Q^{ik}_{ik} \cup S^i_{ij} \cup Q^k_{kj}$.
   (b) Y_{2E} nodes are present in the component containing node j in the network induced by $\tilde{Q}^{ik}_{ik}$ on G_{jk}. Then, $\tilde{Q}^{ik}_{ik} = Q^{ik}_{ik} \cup S^{ij}_{ij} \cup Q^{kj}_{kj}$.
(3) j is in $\tilde{Q}^{ik}_{ik}$ and the network induced on G_{ij} is disconnected. These cases are symmetrical to subcases 2(a) and 2(b) for $\tilde{Q}^{ik}_{ik}$.

Cases for $\tilde{R}_{ik}$. There are 19 cases for $\tilde{R}_{ik}$.

(1) j is not in $\tilde{R}_{ik}$.
   (a) All Y_{2E} nodes are contained within G_{ik}. Then, $\tilde{R}_{ik} = R_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility, b_{ij} and b_{kj} must be zero.
   (b) All Y_{2E} nodes are contained within G_{ij}. Then, $\tilde{R}_{ik} = Q_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility, b_{ik} and b_{jk} must be zero.
   (c) All Y_{2E} nodes are contained within G_{jk}. Then, $\tilde{R}_{ik} = Q_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility, b_{ji} and b_{ki} must be zero.
   (d) All the Y_{2E} nodes are contained within G_{ij} and the component containing node i in the network induced on G_{ik}, and both contain a Y_{2E} node distinct from i. Then, $\tilde{R}_{ik} = Q^i_{ik} \cup T^i_{ij} \cup T_{kj}$. For feasibility, b_{jk} must be zero.
   (e) All the Y_{2E} nodes are contained within G_{jk} and the component containing node k in the network induced on G_{ik}, and both contain a Y_{2E} node distinct from k. Then, $\tilde{R}_{ik} = Q^k_{ik} \cup T^k_{kj} \cup T_{ij}$. For feasibility, b_{ji} must be zero.
(2) j is in $\tilde{R}_{ik}$ and the network induced on G_{ij} by $\tilde{R}_{ik}$ is disconnected.
   (a) All Y_{2E} nodes are contained within G_{ik}. Then, $\tilde{R}_{ik} = R_{ik} \cup Q_{ij} \cup P_{kj}$. For feasibility, b_{ij} and b_{kj} must be zero.
   (b) All Y_{2E} nodes are contained within G_{ij}. Then, $\tilde{R}_{ik} = Q_{ik} \cup R_{ij} \cup P_{kj}$. For feasibility, b_{ik} and b_{jk} must be zero.
   (c) All Y_{2E} nodes are contained within G_{jk}. Then, $\tilde{R}_{ik} = Q_{ik} \cup Q_{ij} \cup S_{kj}$. For feasibility, b_{ji} and b_{ki} must be zero.
   (d) All the Y_{2E} nodes are contained within the components containing node i in the networks induced by $\tilde{R}_{ik}$ on G_{ij} and G_{ik}, and both components contain a Y_{2E} node distinct from i. Then, $\tilde{R}_{ik} = Q^i_{ik} \cup Q^i_{ij} \cup P_{jk}$. For feasibility, b_{kj} and b_{jk} must be zero. (We use both b_{kj} and b_{jk} to ensure both j and k do not belong to Y_{2E}.)
   (e) All the Y_{2E} nodes are contained within G_{jk} and the component containing node k in the network induced on G_{ik}, and both contain a Y_{2E} node distinct from k. Then, $\tilde{R}_{ik} = Q^k_{ik} \cup Q_{ij} \cup S^k_{kj}$. For feasibility, b_{ji} must be zero.
   (f) All the Y_{2E} nodes are contained within G_{jk} and the component containing node j in the network induced on G_{ij}, and both contain a Y_{2E} node distinct from j. Then, $\tilde{R}_{ik} = Q_{ik} \cup Q^j_{ij} \cup S^j_{kj}$. For feasibility, b_{ki} must be zero.
   (g) The component containing node k, but not the component containing node i, in the network induced on G_{ik} contains Y_{2E} nodes, and the component containing node j, but not the component containing node i, in the network induced on G_{ij} contains Y_{2E} nodes. Then, $\tilde{R}_{ik} = Q^k_{ik} \cup Q^j_{ij} \cup S^{kj}_{kj}$. For feasibility, w_i must be zero.
(3) j is in $\tilde{R}_{ik}$ and the network induced on G_{jk} by $\tilde{R}_{ik}$ is disconnected. These cases are symmetrical to subcases 2(a)–2(g) for $\tilde{R}_{ik}$.

REFERENCES

[1] S. Arnborg, J. Lagergren, and D. Seese, Easy problems for tree-decomposable graphs, J Alg 12 (1991), 308–340.

[2] S. Arnborg and A. Proskurowski, Linear time algorithms for NP-hard problems restricted to partial k-trees, Discr Appl Math 23 (1989), 11–24.

[3] A.M. Baïou and A.R. Mahjoub, Steiner 2-edge connected subgraph polytope on series-parallel graphs, SIAM J Discr Math 10 (1997), 505–514.

[4] W. Bern, E.L. Lawler, and L. Wong, Linear time computation of optimal subgraphs of decomposable graphs, J Alg 8 (1987), 216–235.

[5] J. Bondy and U. Murty, Graph theory with applications, North-Holland, New York, 1976.

[6] R.B. Borie, R.G. Parker, and C.A. Tovey, Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families, Algorithmica 7 (1992), 555–581.

[7] R.H. Cardwell, C.L. Monma, and T.H. Wu, Computer-aided design procedures for survivable fiber optic networks, IEEE J Select Areas Commun 7 (1989), 1188–1197.

[8] C.R. Coullard, A. Rais, R.L. Rardin, and D.K. Wagner, The 2-connected spanning subgraph polytope for series-parallel graphs, Technical report 90-12, Purdue University, West Lafayette, Indiana, 1990.

[9] C.R. Coullard, A. Rais, R.L. Rardin, and D.K. Wagner, The 2-connected Steiner subgraph polytope for series-parallel graphs, Technical report 91-32, Purdue University, West Lafayette, Indiana, 1991.

[10] S. Even, Graph algorithms, Computer Science Press, Rockville, MD, 1979.

[11] M. Grötschel, A. Martin, and R. Weismantel, Packing Steiner trees: Polyhedral investigations, Math Program Series A 72 (1996), 101–123.

[12] M. Grötschel, C.L. Monma, and M. Stoer, Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints, Oper Res 40 (1992), 309–330.

[13] M. Grötschel, C.L. Monma, and M. Stoer, "Design of survivable networks," Network models, Vol. 7, Handbooks in operations research and management science, M. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser (Editors), Elsevier, Amsterdam, 1995, pp. 617–672.

[14] J. Hopcroft and R. Tarjan, Dividing a graph into triconnected components, SIAM J Comput 2 (1973), 135–158.

[15] T.L. Magnanti and S. Raghavan, A dual-ascent algorithm for low-connectivity network design, Technical report, Robert H. Smith School of Business, University of Maryland, College Park, MD 20742, 2003.

[16] A.R. Mahjoub, Two-edge connected spanning subgraphs and polyhedra, Math Program 64 (1994), 199–208.

[17] K. Menger, Zur allgemeinen kurventheorie, Fund Math 10 (1927), 96–115.

[18] C.L. Monma and D.F. Shallcross, Methods for designing communications networks with certain two-connected survivability constraints, Oper Res 37 (1989), 531–541.

[19] T. Nishizeki, J. Vygen, and X. Zhou, The edge-disjoint paths problem is NP-complete for series-parallel graphs, Discr Appl Math 115 (2001), 177–186.

[20] S. Raghavan and T.L. Magnanti, "Network connectivity," Annotated bibliographies in combinatorial optimization, M. Dell'Amico, F. Maffioli, and S. Martello (Editors), Wiley, New York, 1997, pp. 335–354.

[21] M.B. Richey and R.G. Parker, On multiple Steiner subgraph problems, Networks 16 (1986), 423–438.

[22] N. Roberstson and P.D. Seymour, Graph width and well-quasi ordering: A survey, Progress in graph theory, J. Bondy and U. Murty (Editors), Academic, New York, 1984, pp. 399–406.

[23] N. Roberstson and P.D. Seymour, Graph minors. II. Algorithmic aspects of treewidth, J Alg 7 (1986), 309–322.

[24] J. Van Leeuwen, Graph algorithms, Handbook of theoretical computer science, Vol A, Algorithms and complexity, J. Van Leeuwen (Editor), Elsevier, Amsterdam, 1990, pp. 525–631.

[25] J.A. Wald and C.J. Colbourn, Steiner trees, partial 2-trees and minimum IFI networks, Networks 13 (1983), 159–167.

[26] P. Winter, Generalized Steiner problems in series-parallel networks, J Alg 7 (1987), 549–566.