TECHNICAL RESEARCH REPORT

Low Connectivity Network Design on Series-Parallel Graphs

by S. Raghavan

TR 2001-39

ISR develops, applies and teaches advanced methodologies of design and analysis to solve complex, hierarchical, heterogeneous and dynamic problems of engineering technology and systems for industry and government.

ISR is a permanent institute of the University of Maryland, within the Glenn L. Martin Institute of Technology/A. James Clark School of Engineering. It is a National Science Foundation Engineering Research Center.

Web site http://www.isr.umd.edu

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, MD 20742. e-mail: sr141@umail.umd.edu

Abstract

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 $o_i \in (0, 1, 2)$ for each node, design a network at minimum cost that contains at least $o_{st} = \min(o_s, o_t)$ 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.

Key words: Network design, series-parallel graphs, connectivity.

1 Introduction

With the advent of modern fiber-optic telecommunication networks, and the multimedia broadband services that they carry, network survivability has become a critical issue to telecommunications companies. Fiber-optic cables are expensive and have the capacity to carry large volumes of traffic. Consequently, minimum cost networks that simply satisfy the network demands are treelike in structure. However this type of design is undesirable, since the loss of a single cable (edge) or switching center (node) can result in a disconnected network. As a result telecommunications companies topologically design their networks so that there are two physically diverse paths between pairs of important node locations. This provides a measure of reliability because the loss of a single link or node does not result in a disconnection between the important nodes in the network. Furthermore, the probability of having two

failures in the network simultaneously is considered to be so low that having two physically diverse paths provides adequate reliability.

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 [5]). In this problem, given an undirected network $G = (N, E)$, with node set N, edge set E, a connectivity requirement $o_i \in \{0, 1, 2\}$ for each node $i \in N$, edge costs c_e for each edge $e \in E$, we wish to design a minimum cost network that has at least $o_{st} = \min\{o_s, o_t\}$ disjoint paths between each pair of nodes s and t. We consider both edge- and node-connectivity versions of the problem [1]. To distinguish between edge-connectivity and node-connectivity we use the superscript N (for example, 2^N) to denote node-connectivity requirements and the superscript 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, Monma and Stoer [9] describe a cutting plane approach for the problem on general graphs, and have been able to use this approach to solve some problems arising in local telephone companies. Magnanti and Raghavan [12] describe 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 have studied the problem on general graphs. See Grötschel, Monma and Stoer [10], and Raghavan and Magnanti [15] for recent surveys on this problem.

Telecommunication networks are typically sparse and planar. 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 [16] describe a linear time algorithm for the Steiner tree problem ($o_s \in \{0, 1\}$) on a series parallel graph. Winter [17] describes linear time algorithms for the LCND on series-parallel graphs when $o_s \in \{0, 2^N\}$ and $o_s \in \{0, 2^E\}$ (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 [13] provides a complete description [2] of the 2-edge-connected spanning subgraph polytope on series-parallel graphs (i.e., the case where $o_s = 2^E$ for all nodes). Baïou and Mahjoub [2] consider the case where $o_s \in \{0, 2^E\}$ and provide a complete description of the Steiner 2-

[1] There are two edge-disjoint paths (resp. node-disjoint paths) between a pair of nodes if the deletion of a single edge (resp. single node) in the network does not disconnect them. A network is 2-edge connected (resp. 2-node connected) if there are two edge-disjoint paths (resp. node-disjoint paths) between every pair of nodes.
[2] A complete description is a linear inequality description of the convex hull of integer feasible solutions to the problem.

edge-connected polytope. Coullard et al. [6,7] consider the node-connectivity version. In [6] they give a complete description of the 2-node-connected spanning subgraph polytope on series-parallel graphs (i.e., the case where $o_s = 2^N$ for all nodes), and in [7] they give a complete description of the Steiner 2-node-connected polytope (i.e., the case with $o_s \in \{0, 2^N\}$).

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 non-trivial, 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 [9]). 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, Monma and Stoer [9] 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 [4]. For clarity, we elaborate on the following terminology where we differ from [4]. A *trail* is a path that does not repeat edges. A *simple path* is a trail that does not repeat nodes. Let $G = (N_1, E_1)$ 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 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 4 nodes). An alternate, constructive, definition is as follows. A graph is series-parallel if it can be constructed, starting from a forest, 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 3 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 [16] characterize those networks, partial 2-trees, that can be completed to 2-trees by adding new edges. They show that a network is a partial 2-tree if and only if it has no subgraph homeomorphic to K_4. Thus partial 2-trees are exactly series-parallel graphs.

We now briefly sketch a linear time procedure developed by Wald and Colbourn [16] and adapted by Winter [17] 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 nodes with connectivity requirements, i.e. nodes with $o_s \geq 1$, 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 [8]).

The procedure to complete a partial 2-tree to a 2-tree 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 depth first search (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 as follows. 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 K_3 or the stack is empty.

(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 G.
(3) If there is no edge (i, k) add it.
(4) Delete node j. $G = G \backslash j$.
(5) If the degree of i or k in G 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 come 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, 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). During the contraction process we keep track of information (states) for the subgraph G_{ij} represented by each edge (i, j). The state information describes solutions to certain problems (that can be different from the original problem) restricted to the subgraph G_{ij}.

In order to use the contraction process to devise linear time algorithms, there are three requirements. First, the number of states we associate with each subgraph is *finite* and *independent* of the number of nodes. Second, suppose 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}, 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.

Although it is easy to state the algorithm design philosophy, and this has been formalized in several different ways (see [1,3]), it is a non-trivial task to determine the states 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 $G'_{ik} = G_{ik} \cup G_{ij} \cup G_{jk}$.
(4) Delete node j. $G' = G \backslash j$.
(5) If the degree of i or k in G is 2, place them on the stack. $G = G'$.

3 Node Connectivity Requirements

In the case of node connectivity requirements let Z_1 be the set of nodes with $o_s = 1$ and let Z_2^N be the set of nodes with $o_s = 2^N$. We now motivate the graphical structures (states) that we compute in the course of the algorithm. At the end of the contraction process an edge, say (i, k), represents the graph. The minimum cost solution to the problem either (i) includes both nodes i and k, (ii) includes node i but not k, (iii) includes node k but not i, or (iv) excludes both nodes i and k. Thus, at the minimum, we must keep track of graph 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 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 require-

ments on G_{ik} and must include i and must exclude k. If k is required (i.e., $o_k >= 1$) 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 graph structures are necessary for the computations.

P_{ik} = minimum cost network on G_{ik} containing i and k, and such that for every Z_2^N node there is a simple path (i.e., a path that does not repeat a node) from i to k through it, and every Z_1 node is connected to i and to k.

Q_{ik} = minimum cost network on G_{ik} comprising of two disjoint trees. One part includes i and the other part includes k, and all nodes in G_{ik} with non-zero connectivity requirements must be in one of the two components. If $G_{ik}\backslash\{i,k\}$ has an Z_2^N node then by convention $q_{ik} = \infty$.

R_{ik} = minimum cost network on G_{ik} comprising of two disjoint networks. One part includes i and the other part part includes k, and all nodes within G_{ik} with non-zero connectivity requirements must be in one of the two components. Further, all the Z_2^N nodes belong to only one of the disjoint parts, and there must be two node disjoint paths between every pair of Z_2^N nodes. If this is not the case, for example if i and $k \in Z_2^N$, then $r_{ik} = \infty$.

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 Z_2^N node. If $G_{ik}\backslash\{i\}$ contains at least one Z_2^N 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 graph structures on each edge $(i,k) \in E$ are initialized as follows:

$$s_{ik} = \begin{cases} \infty \text{ if i} \in Z_2^N \text{ and } k \in Z_2^N \\ c_{ik} \text{ otherwise} \end{cases}$$

$$t_{ik} = \begin{cases} \infty \text{ if } k \in Z_1 \cup Z_2^N \\ 0 \quad \text{otherwise} \end{cases}$$

$$u_{ik} = \begin{cases} \infty \text{ if either } i \text{ or } k \in Z_1 \cup Z_2^N \\ 0 \quad \text{otherwise} \end{cases}$$

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

$$q_{ik} = 0$$

$$r_{ik} = \begin{cases} \infty \text{ if } i \in Z_2^N \text{ and } k \in Z_2^N \\ 0 \quad \text{otherwise} \end{cases}$$

$$a_{ik} = \begin{cases} \infty \text{ if } k \in Z_2^N \\ 0 \quad \text{otherwise} \end{cases}$$

$$m_{ik} = \begin{cases} \infty \text{ if } i \text{ or } k \in Z_1 \cup Z_2^N \\ 0 \quad \text{otherwise} \end{cases}$$

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 $G'_{ik} = G_{ik} \cup G_{ij} \cup G_{jk}$ is updated as described in the following recursive equations.

The Recursive Equations (Node-Connectivity Case) **(1)**

$$\begin{aligned}
s'_{ik} = \min\{&p_{ij} + p_{jk} + p_{ik}, \\
&t_{ij} + t_{kj} + \min\{s_{ik} + a_{ij} + a_{kj}, p_{ik} + a_{ik} + a_{jk}, p_{ik} + a_{ki} + a_{ji}\}, \\
&s_{ik} + p_{ij} + q_{jk} + a_{ij} + a_{kj}, p_{ik} + s_{ij} + q_{jk} + a_{ik} + a_{jk}, \\
&p_{ik} + p_{ij} + r_{jk} + a_{ji} + a_{ki}, \\
&s_{ik} + p_{kj} + q_{ij} + a_{ij} + a_{kj}, p_{ik} + p_{kj} + r_{ij} + a_{ik} + a_{jk}, \\
&p_{ik} + s_{kj} + q_{ij} + a_{ji} + a_{ki}, \\
&r_{ik} + p_{ij} + p_{jk} + a_{ij} + a_{kj}, q_{ik} + s_{ij} + p_{jk} + a_{ik} + a_{jk}, \\
&q_{ik} + p_{ij} + s_{jk} + a_{ji} + a_{ki}\} \\
t'_{ik} = \min\{&t_{ik} + t_{jk} + \min\{p_{ij} + a_{ij} + a_{kj}, s_{ij} + a_{ik} + a_{jk}, p_{ij} + a_{ji} + a_{ki}\}, \\
&m_{jk} + t_{ik} + t_{ij} + \min\{a_{ij}, a_{ik}\}\} \\
u'_{ik} = \min\{&m_{ij} + m_{jk} + u_{ik}, m_{ik} + m_{jk} + u_{ij}, m_{ij} + m_{ik} + u_{jk}, \\
&t_{ji} + t_{jk} + m_{ik} + \min\{a_{jk}, a_{ji}\}\} \\
p'_{ik} = \min\{&p_{ij} + p_{jk} + p_{ik}, p_{ik} + t_{ij} + t_{kj} + a_{ij} + a_{kj}, p_{ij} + p_{jk} + q_{ik}, \\
&p_{ik} + p_{ij} + q_{jk} + a_{ij}, p_{ik} + p_{jk} + q_{ij} + a_{kj}\} \\
q'_{ik} = \min\{&q_{ik} + a_{ij} + a_{kj} + \min\{t_{ij} + t_{kj}, p_{ij} + q_{jk}, p_{jk} + q_{ij}\}\} \\
r'_{ik} = \min\{&t_{ij} + t_{kj} + \min\{r_{ik} + a_{ij} + a_{kj}, q_{ik} + a_{ik} + a_{jk}, q_{ik} + a_{ki} + a_{ji}\},
\end{aligned}$$

$$
\begin{aligned}
& r_{ik} + p_{ij} + q_{jk} + a_{ij} + a_{kj}, q_{ik} + s_{ij} + q_{jk} + a_{ik} + a_{jk}, \\
& q_{ik} + p_{ij} + r_{jk} + a_{ji} + a_{ki}, r_{ik} + p_{jk} + q_{ij} + a_{ij} + a_{kj}, \\
& q_{ik} + p_{jk} + r_{ij} + a_{jk} + a_{ik}, q_{ik} + s_{jk} + q_{ij} + a_{ji} + a_{ki}\} \\
a'_{ik} = & a_{ij} + a_{ik} + a_{jk} \\
m'_{ik} = & m_{ij} + m_{ik} + m_{jk}
\end{aligned}
$$

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.*

PROOF. The proof requires enumeration of all possible cases. We describe the cases for structure 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 S'_{ik}. In doing so we will also motivate the need for the structures P_{ik}, Q_{ik}, R_{ik}. The following result, that follows immediately from the well-known Mengers' theorem [14], is useful in our discussion.

Lemma 2 *Let $\{i, j\}$ be a 2-separator, separating two nodes s and $t \in Z_2^N$. 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 $G'_{ik} = G_{ij} \cup G_{jk} \cup G_{ik}$. Suppose all the Z_2^N 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 Z_2^N that have one of $\{i, j\}, \{j, k\}, \{i, k\}$ as 2-separators separating them; or all three nodes $i, j, k \in Z_2^N$. In both cases, using Lemma 2, it follows that each of the graphs induced by 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 S'_{ik} at minimum cost. Notice that in this case, $s'_{ik} = p_{ij} + p_{jk} + p_{ik}$.

Now suppose all the Z_2^N nodes are contained entirely in one of the subgraphs G_{ij}, G_{jk}, and G_{ik}. Then there are twelve possible cases.

(1) j is not in S'_{ik}.

(a) All Z_2^N nodes are contained in G_{ik}. Then the union of S_{ik}, T_{ij} and T_{kj} gives S'_{ik}. In addition a_{ij} and a_{kj} should be zero, otherwise all Z_2^N nodes are not contained in G_{ik}. In this case $s'_{ik} = t_{ij} + t_{kj} + s_{ik} + a_{ij} + a_{kj}$.

(b) All Z_2^N nodes are contained in G_{ij}. Then the union of P_{ik}, T_{ij} and T_{kj} gives S'_{ik}. Note that if G_{ik} contains no Z_2^N node then P_{ik} gives the optimal Steiner tree on the node set $Z_1 \cup \{i, k\}$ on G_{ik}. In addition a_{ik} and a_{jk} must be zero, otherwise all Z_2^N nodes are not contained in G_{ij}. In this case $s'_{ik} = t_{ij} + t_{kj} + p_{ik} + a_{ik} + a_{jk}$.

(c) All Z_2^N nodes are contained in G_{jk}. Then the union of P_{ik}, T_{ij} and T_{kj} gives S'_{ik}. In addition a_{ki} and a_{ji} should be zero, otherwise all Z_2^N nodes are not contained in G_{jk}. In this case $s'_{ik} = t_{ij} + t_{kj} + p_{ik} + a_{ki} + a_{ji}$.

(2) j is in S'_{ik} and the graph induced on G_{jk} is disjoint. The disjoint graph on G_{jk} induces two structures R_{jk} and Q_{jk} (defined earlier) based on whether Z_2^N nodes are contained in G_{jk}.

(a) All Z_2^N nodes are contained in G_{ik}. Then the union of S_{ik}, P_{ij} and Q_{jk} gives S'_{ik}. To ensure feasibility a_{ij} and a_{kj} must be zero. Thus, $s'_{ik} = s_{ik} + p_{ij} + q_{jk} + a_{ij} + a_{kj}$.

(b) All Z_2^N nodes are contained in G_{ij}. Then the union of P_{ik}, S_{ij} and Q_{jk} gives S'_{ik}. To ensure feasibility a_{ik} and a_{jk} must be zero. Thus, $s'_{ik} = p_{ik} + s_{ij} + q_{jk} + a_{ik} + a_{jk}$.

(c) All Z_2^N nodes are contained in G_{jk}. Then the union of P_{ik}, P_{ij}, R_{jk} gives S'_{ik}. To ensure feasibility a_{ji} and a_{ki} must be zero. Thus, $s'_{ik} = p_{ik} + p_{ij} + r_{jk} + a_{ji} + a_{ki}$.

(3) j is in S'_{ik} and the graph induced on G_{ij} is disjoint.

(a) All Z_2^N nodes are contained in G_{ik}. Then the union of S_{ik}, P_{kj} and Q_{ij} gives S'_{ik}. To ensure feasibility a_{ij} and a_{kj} must be zero. Thus, $s'_{ik} = s_{ik} + p_{kj} + q_{ij} + a_{ij} + a_{kj}$.

(b) All Z_2^N nodes are contained in G_{ij}. Then the union of P_{ik}, P_{kj} and R_{ij} gives S'_{ik}. To ensure feasibility a_{ik} and a_{jk} must be zero. Thus, $s'_{ik} = p_{ik} + p_{kj} + r_{ij} + a_{ik} + a_{jk}$.

(c) All Z_2^N nodes are contained in G_{jk}. Then the union of P_{ik}, S_{kj} and Q_{ij} gives S'_{ik}. To ensure feasibility a_{ji} and a_{ki} should be zero. Thus, $s'_{ik} = p_{ik} + s_{kj} + q_{ij} + a_{ji} + a_{ki}$.

(4) j is in S'_{ik} and the graph induced on G_{ik} is disjoint.

(a) All Z_2^N nodes are contained in G_{ik}. Then the union of R_{ik}, P_{ij} and P_{jk} gives S'_{ik}. To ensure feasibility a_{ij} and a_{kj} need to be zero. Thus, $s'_{ik} = r_{ik} + p_{ij} + p_{jk} + a_{ij} + a_{kj}$.

(b) All Z_2^N nodes are contained in G_{ij}. Then the union of Q_{ik}, S_{ij} and P_{jk} gives S'_{ik}. To ensure feasibilty a_{ik} and a_{jk} need be zero. Thus, $s'_{ik} = q_{ik} + s_{ij} + p_{jk} + a_{ik} + a_{jk}$.

(c) All Z_2^N nodes are contained in G_{jk}. Then the union of Q_{ik}, P_{ij} and S_{jk} gives S'_{ik}. To ensure feasibility a_{ji} and a_{ki} need be zero. Thus, $s'_{ik} = q_{ik} + p_{ij} + s_{jk} + a_{ji} + a_{ki}$.

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 s'_{ik} showing that s'_{ik} is computed correctly.

$t'_{ik}, u'_{ik}, p'_{ik}, q'_{ik}$, and r'_{ik} are computed correctly. (see appendix).

a'_{ik} is computed correctly. a'_{ik} is ∞ if any node in $G'_{ik}\backslash i$ is a Z_2^N node. Thus it is ∞ if either any node in $G_{ik}\backslash i$, $G_{ij}\backslash i$, or G_{jk} is a Z_2^N node. The equation $a'_{ik} = a_{ik} + a_{ij} + a_{jk}$ expresses this condition.

m'_{ik} is computed correctly. m'_{ik} is ∞ if any node in 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 $m'_{ik} = m_{ij} + m_{ik} + m_{jk}$ 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.

4 Edge Connectivity Requirements

In the case of edge connectivity requirements let Z_1 be the set of nodes with $o_s = 1$ and let Z_2^E be the set of nodes with $o_s = 2^E$. The methodology is similar to that for the node connectivity case except that the graphical structures we need to keep track of are more complicated. At the end of the contraction process let the edge remaining, that represents the graph, be (i, k). Then, the minimum cost solution to the problem either (i) includes both nodes i and k, (ii) includes node i but not node k, (iii) includes node k but not node i, or (iv) 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 $\{i, j\}$ is a 2-separator separating two nodes $s, t \in Z_2^E$, 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. This drastically increases the number of structures we need to consider. Unlike the node-connectivity case, we need to consider structures where the endpoints have 2^E 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 2^E requirement is imposed on nodes i and k.

S_{ik}^{i} = minimum cost network on G_{ik} that satisfies the requirements of all nodes on G_{ik} and furthermore a 2^E requirement is imposed on node i and node k is required to be connected.

S_{ik}^{k} = minimum cost network on G_{ik} that satisfies the requirements of all nodes on G_{ik} and furthermore a 2^E requirement is imposed on node k and node i is required to be connected.

T_{ik}^{i} = minimum cost network on G_{ik} that satisfies the connectivity requirements on G_{ik}, excludes node k and imposes a 2^E requirement on node i. If k is required to be connected then $t_{ik}^{i} = \infty$.

In addition the following structures are necessary for the computations.

P_{ik} = minimum cost network on G_{ik} such that for every Z_2^E 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 Z_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}. One part includes i and the other part includes k, both of which are have 2^E requirements imposed on them. Further, the connectivity requirements within each disjoint network are satisfied.

Q_{ik}^{i} = minimum cost network on G_{ik} comprising of two disjoint networks on

G_{ik}. One part includes i and the other part includes k. All nodes in G_{ik} with non-zero connectivity requirements must be in one of the two components. Node i has a 2^E requirement imposed on it, and all Z_2^E nodes in $G_{ik}\backslash\{i,k\}$ are on the i part. Further, the connectivity requirements within each disjoint network are satisfied. Note, if the k part of Q_{ik}^i has a node with a 2^E requirement other than k then $q_{ik}^i = \infty$. This implies that the k part of Q_{ik}^i is a tree or just node k.

Q_{ik}^k = minimum cost network on G_{ik} comprising of two disjoint networks on G_{ik}. One part includes i and the other part includes k. All nodes in G_{ik} with non-zero connectivity requirements must be in one of the two components. Node k has a 2^E requirement imposed on it, and all Z_2^E nodes in $G_{ik}\backslash\{i,k\}$ are on the k part. Further, the connectivity requirements within each disjoint network are satisfied.

Q_{ik} = minimum cost network on G_{ik} comprising of two disjoint trees. One part includes i and the other part includes k, and all nodes in G_{ik} with non-zero connectivity requirements must be in one of the two components. If $G_{ik}\backslash\{i,k\}$ has an Z_2^E node then by convention $q_{ik} = \infty$.

R_{ik} = minimum cost network on G_{ik} comprising of two disjoint networks. One part includes i and the other part part includes k, and all nodes within G_{ik} with non-zero connectivity requirements must be in one of the two components. Further, all the Z_2^E nodes belong to only one of the disjoint parts, and there must be two edge disjoint paths between every pair of Z_2^E nodes. If this is not the case, for example if i and $k \in Z_2^E$, then $r_{ik} = \infty$.

Finally, the following variables provide information on feasibility.

b_{ik} indicates whether $G_{ik}\backslash\{i\}$ contains at least one Z_2^E node. If $G_{ik}\backslash\{i\}$ contains at least one Z_2^E 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.

y_k indicates whether node $k \in Z_2^E$. y_k is ∞ if $k \in Z_2^E$, and is zero otherwise.

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

$$p_{ik} = c_{ik}$$
$$s_{ik}^{ik} = \infty$$

$$s^i_{ik} = \begin{cases} \infty \text{ if } k \in Z_2^E \\ c_{ik} \text{ otherwise} \end{cases}$$

$$s^k_{ik} = \begin{cases} \infty \text{ if } i \in Z_2^E \\ c_{ik} \text{ otherwise} \end{cases}$$

$$s_{ik} = \begin{cases} \infty \text{ if } i \in Z_2^E \text{ and } j \in Z_2^E \\ c_{ik} \text{ otherwise} \end{cases}$$

$$u_{ik} = \begin{cases} \infty \text{ if either } i \text{ or } k \in Z_1 \cup Z_2^E \\ 0 \quad \text{otherwise} \end{cases}$$

$$t^i_{ik} = \begin{cases} \infty \text{ if } k \in Z_1 \cup Z_2^E \\ 0 \quad \text{otherwise} \end{cases}$$

$$t_{ik} = \begin{cases} \infty \text{ if } k \in Z_1 \cup Z_2^E \\ 0 \quad \text{otherwise} \end{cases}$$

$$q^{ik}_{ik} = 0$$
$$q^i_{ik} = 0$$
$$q^k_{ik} = 0$$
$$q_{ik} = 0$$

$$r_{ik} = \begin{cases} \infty \text{ if } i \in Z_2^E \text{ and } k \in Z_2^E \\ 0 \quad \text{otherwise} \end{cases}$$

$$b_{ik} = \begin{cases} \infty \text{ if } k \in Z_2^E \\ 0 \quad \text{otherwise} \end{cases}$$

$$m_{ik} = \begin{cases} \infty \text{ if } i \text{ or } k \in Z_1 \cup Z_2^E \\ 0 \quad \text{otherwise} \end{cases}$$

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

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

$$\begin{aligned} s'_{ik} = \min\{&p_{ij} + p_{jk} + p_{ik}, \\ &t_{ij} + t_{kj} + \min\{s_{ik} + b_{ij} + b_{kj}, p_{ik} + b_{jk} + b_{ik}, p_{ik} + b_{ji} + b_{ki}\}, \\ &s^i_{ik} + t^i_{ij} + b_{kj} + t_{kj}, s^k_{ik} + t^k_{kj} + b_{ij} + t_{ij}, s^{ik}_{ik} + t^i_{ij} + t^k_{kj}, \end{aligned}$$

$$
\begin{aligned}
& 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^j_{ij}+s^{jk}_{jk}+q^k_{ik}, s^{ij}_{ij}+s^{jk}_{jk}+q^{ik}_{ik}\}\\
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}\}\\
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}\}\\
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}\}\\
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}\}\\
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}\}\\
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},
\end{aligned}
$$

$$
\begin{aligned}
& p_{ik} + s^{k}_{kj} + q^{i}_{ij}, p_{ik} + s^{kj}_{kj} + q^{ij}_{ij}, \\
& p_{ij} + p_{jk} + q^{ik}_{ik}\} \\
q'_{ik} =& \min\{q_{ik} + b_{ij} + b_{kj} + \min\{t_{ij} + t_{kj}, p_{ij} + q_{jk}, p_{kj} + q_{ij}\}\} \\
q'^{i}_{ik} =& \min\{q^{i}_{ik} + t_{kj} + b_{kj} + t^{i}_{ij}, q^{i}_{ik} + p_{kj} + b_{kj} + q^{i}_{ij}, \\
& q^{i}_{ik} + s^{i}_{ij} + q_{jk}, q^{i}_{ik} + s^{ij}_{ij} + q^{j}_{jk}\} \\
q'^{ik}_{ik} =& \min\{q^{ik}_{ik} + t^{i}_{ij} + t^{k}_{kj}, \\
& q^{ik}_{ik} + s^{i}_{ij} + q^{k}_{jk}, q^{ik}_{ik} + s^{ij}_{ij} + q^{jk}_{jk}, \\
& q^{ik}_{ik} + s^{k}_{kj} + q^{i}_{ji}, q^{ik}_{ik} + s^{kj}_{kj} + q^{ji}_{ji}\} \\
r'_{ik} =& \min\{t_{ij} + t_{kj} + \min\{r_{ik} + b_{ij} + b_{kj}, q_{ik} + b_{ik} + b_{jk}, q_{ik} + b_{ji} + b_{ki}\}, \\
& q^{i}_{ik} + t^{i}_{ij} + t_{kj} + b_{jk}, q^{k}_{ik} + t^{k}_{kj} + t_{ij} + b_{ji}, \\
& r_{ik} + q_{ij} + p_{kj} + b_{ij} + b_{kj}, q_{ik} + r_{ij} + p_{kj} + b_{ik} + b_{jk}, \\
& q_{ik} + q_{ij} + s_{kj} + b_{ji} + b_{ki}, \\
& q^{i}_{ik} + q^{i}_{ij} + p_{jk} + b_{kj} + b_{jk}, q^{k}_{ik} + q_{ij} + s^{k}_{kj} + b_{ji}, \\
& q_{ik} + q^{j}_{ij} + s^{j}_{kj} + b_{ki}, q^{k}_{ik} + q^{j}_{ij} + s^{kj}_{kj} + y_i, \\
& r_{ik} + p_{ij} + q_{jk} + b_{ij} + b_{kj}, q_{ik} + s_{ij} + q_{jk} + b_{ik} + b_{jk}, \\
& q_{ik} + p_{ij} + r_{jk} + b_{ji} + b_{ki}, \\
& q^{k}_{ik} + p_{ij} + q^{k}_{jk} + b_{ij} + b_{ji}, q^{i}_{ik} + s^{i}_{ij} + q_{jk} + b_{jk}, \\
& q_{ik} + s^{j}_{ij} + q^{j}_{jk} + b_{ik}, q^{i}_{ik} + s^{ij}_{ij} + q^{j}_{jk} + y_k\} \\
b'_{ik} =& b_{ij} + b_{jk} + b_{ik} \\
m'_{ik} =& m_{ik} + m_{ij} + m_{jk}
\end{aligned}
$$

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.*

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

There are 34 possible cases.

(1) None of the networks induced on G_{ij}, G_{ik}, G_{jk} by S'_{ik} are disjoint. Then it follows that $S'_{ik} = P_{ik} \cup P_{ij} \cup P_{jk}$. In this case $s'_{ik} = p_{ij} + p_{jk} + p_{ik}$.
(2) Node j is not in S'_{ik}. There are 6 distinct subcases.

(a) All the Z_2^E nodes are contained within G_{ik}. Then $S'_{ik} = S_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility we need b_{ij} and b_{kj} to be zero. Thus $s'_{ik} = t_{ij} + t_{kj} + s_{ik} + b_{ij} + b_{kj}$.

(b) All the Z_2^E nodes are contained within G_{ij}. Then $S'_{ik} = P_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility we need b_{ik} and b_{jk} to be zero. Thus $s'_{ik} = t_{ij} + t_{kj} + p_{ik} + b_{jk} + b_{ik}$.

(c) All the Z_2^E nodes are contained within G_{jk}. Then $S'_{ik} = P_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility we require that b_{ji} and b_{ki} to be zero. Thus $s'_{ik} = t_{ij} + t_{kj} + p_{ik} + b_{ji} + b_{ki}$.

(d) All the Z_2^E nodes are contained within $G_{ij} \cup G_{ik}$, and there is at least one Z_2^E node within G_{ij} and at least one Z_2^E node within G_{ik}; both distinct from i. This forces a 2^E requirement on i. Thus $S'_{ik} = S^i_{ik} \cup T^i_{ij} \cup T_{kj}$. In addition, for feasibility we require that b_{kj} be zero. Thus $s'_{ik} = s^i_{ik} + t^i_{ij} + t_{kj} + b_{kj}$.

(e) All the Z_2^E nodes are contained within $G_{jk} \cup G_{ik}$, and there is at least one Z_2^E node within G_{jk} and at least one Z_2^E node within G_{ik}; both distinct from k. This forces a 2^E requirement on k. Thus $S'_{ik} = S^k_{ik} \cup T_{ij} \cup T^k_{kj}$. For feasibility we require that b_{ij} be zero. Thus $s'_{ik} = s^k_{ik} + t_{ij} + t^k_{kj} + b_{ij}$.

(f) G_{ij} contains at least one Z_2^E node and G_{jk} contains at least one Z_2^E node. This forces a 2^E requirement on i and k. Thus $S'_{ik} = S^{ik}_{ik} \cup T^i_{ij} \cup T^k_{kj}$, and $s'_{ik} = s^{ik}_{ik} + t^i_{ij} + t^k_{kj}$.

(3) Node j is in S'_{ik} and the graph induced on G_{jk} is disjoint. There are 9 distinct subcases.

(a) All the Z_2^E nodes are contained within G_{ik}. Then $S'_{ik} = S_{ik} \cup P_{ij} \cup Q_{kj}$. In addition, for feasibility we require that b_{ij} and b_{kj} be zero. Thus $s'_{ik} = s_{ik} + p_{ij} + q_{kj} + b_{ij} + b_{kj}$.

(b) All the Z_2^E nodes are contained within G_{ij}. Then $S'_{ik} = P_{ik} \cup S_{ij} \cup Q_{kj}$. For feasibility we require that b_{ik} and b_{jk} be zero. Thus $s'_{ik} = p_{ik} + s_{ij} + q_{kj} + b_{ik} + b_{jk}$.

(c) All the Z_2^E nodes are contained within G_{jk}. Then $S'_{ik} = P_{ik} \cup P_{ij} \cup R_{jk}$. For feasibility we require that b_{ji} and b_{ki} be zero. Thus $s'_{ik} = p_{ik} + p_{ij} + r_{jk} + b_{ji} + b_{ki}$.

(d) All the Z_2^E nodes are contained within $G_{ij} \cup G_{ik}$, and there is at least one Z_2^E node within G_{ij} and at least one Z_2^E node within G_{ik}; both distinct from i. This forces a 2^E requirement on i. Thus $S'_{ik} = S^i_{ik} \cup S^i_{ij} \cup Q_{jk}$. Note that no additional feasibility variables are required here since $q_{jk} = \infty$ if any nodes in $G_{jk} \backslash \{j, k\}$ are in Z_2^E. Thus $s'_{ik} = s^i_{ik} + s^i_{ij} + q_{kj}$.

(e) All the Z_2^E nodes are contained within G_{ik} and the k side of the disjoint network induced on G_{jk} by S'_{ik}. This forces a 2^E requirement on k. Then $S'_{ik} = S^k_{ik} \cup P_{ij} \cup Q^k_{kj}$. For feasibility we require that b_{ij} be zero. Thus $s'_{ik} = s^k_{ik} + p_{ij} + q^k_{kj} + b_{ij}$.

(f) All the Z_2^E nodes are contained within G_{ij} and the j side of the disjoint network induced on G_{jk} by S'_{ik}. This forces a 2^E requirement

on j. Then $S'_{ik} = P_{ik} \cup S^j_{ij} \cup Q^j_{jk}$. For feasibility we require b_{ik} be zero. Thus $s'_{ik} = p_{ik} + s^j_{ij} + q^j_{jk} + b_{ik}$.

(g) G_{ik} contains at least one Z_2^E node, the j side of the disjoint network induced on G_{jk} by S'_{ik} contains at least one Z_2^E node, and the k side does not contain any Z_2^E node. This forces a 2^E requirement on i and j. Then $S'_{ik} = S^i_{ik} \cup S^{ij}_{ij} \cup Q^j_{jk}$. Thus $s'_{ik} = s^i_{ik} + s^{ij}_{ij} + q^j_{jk}$.

(h) G_{ij} contains at least one Z_2^E node, the k side of the disjoint network induced on G_{jk} by S'_{ik} contains at least one Z_2^E node, and the j side does not contain any Z_2^E node. This forces a 2^E requirement on i and k. Then $S'_{ik} = S^{ik}_{ik} \cup S^i_{ij} \cup Q^k_{kj}$. Thus $s'_{ik} = s^{ik}_{ik} + s^i_{ij} + q^k_{kj}$.

(i) Both the j side and k side of the disjoint network induced on G_{jk} by S'_{ik} contain at least one Z_2^E node. This forces a 2^E requirement on j, k and i. Then $S'_{ik} = S^{ik}_{ik} \cup S^{ij}_{ij} \cup Q^{jk}_{jk}$. Thus $s'_{ik} = s^{ik}_{ik} + s^{ij}_{ij} + q^{jk}_{jk}$.

(4) Node j is in S'_{ik} and the graph induced on G_{ij} is disjoint. There are 9 distinct subcases.

(a) All the Z_2^E nodes are contained within G_{ik}. Then $S'_{ik} = S_{ik} \cup P_{kj} \cup Q_{ij}$. In addition, for feasibility b_{ij} and b_{kj} are zero. Thus $s'_{ik} = s_{ik} + p_{kj} + q_{ij} + b_{ij} + b_{kj}$.

(b) All the Z_2^E nodes are contained within G_{jk}. Then $S'_{ik} = P_{ik} \cup S_{jk} \cup Q_{ij}$. For feasibility b_{ki} and b_{ji} are zero. Thus $s'_{ik} = p_{ik} + s_{jk} + q_{ij} + b_{ji} + b_{ki}$.

(c) All the Z_2^E nodes are contained within G_{ij}. Then $S'_{ik} = P_{ik} \cup P_{jk} \cup R_{ij}$. For feasibility b_{jk} and b_{ik}. Thus $s'_{ik} = p_{ik} + p_{jk} + r_{ij} + b_{ik} + b_{jk}$.

(d) All the Z_2^E nodes are contained within $G_{ik} \cup G_{kj}$, and there is at least one Z_2^E node within G_{ik} and at least one Z_2^E node within G_{kj}; both distinct from k. This forces a 2^E requirement on k. Then $S'_{ik} = S^k_{ik} \cup S^k_{kj} \cup Q_{ij}$. Thus $s'_{ik} = s^k_{ik} + s^k_{kj} + q_{ij}$.

(e) All the Z_2^E nodes are contained within G_{ik} and the i side of the disjoint network induced on G_{ij} by S'_{ik}. This forces a 2^E requirement on i. Then $S'_{ik} = S^i_{ik} \cup P_{kj} \cup Q^i_{ij}$. For feasibility we require that b_{kj} be zero. Thus $s'_{ik} = s^i_{ik} + p_{kj} + q^i_{ij} + b_{kj}$.

(f) All the Z_2^E nodes are contained within G_{jk} and the j side of the disjoint network induced on G_{ij} by S'_{ik}. This forces a 2^E requirement on j. Then $S'_{ik} = P_{ik} \cup S^j_{kj} \cup Q^j_{ij}$. For feasibility we require b_{ki} be zero. Thus $s'_{ik} = s^j_{jk} + p_{ik} + b_{ki} + q^j_{ji}$.

(g) G_{ik} contains at least one Z_2^E node, the j side of the disjoint network induced on G_{ij} by S'_{ik} contains at least one Z_2^E node, and the i side does not contain any Z_2^E node. This forces a 2^E requirement on j and k. Then $S'_{ik} = S^k_{ik} \cup S^{jk}_{jk} \cup Q^j_{ij}$. Thus $s'_{ik} = s^k_{ik} + s^{kj}_{kj} + q^j_{ij}$.

(h) G_{jk} contains at least one Z_2^E node, the i side of the disjoint network induced on G_{ij} by S'_{ik} contains at least one Z_2^E node, and the j side does not contain any Z_2^E node. This forces a 2^E requirement on i and k. Then $S'_{ik} = S^{ik}_{ik} \cup S^k_{jk} \cup Q^i_{ij}$. Thus $s'_{ik} = s^{ik}_{ik} + s^k_{kj} + q^i_{ij}$.

(i) Both the i side and j side of the disjoint network induced on G_{ij} by S'_{ik} contain at least one Z_2^E node. This forces a 2^E requirement on i,

j and k. Then $S'_{ik} = S^{ik}_{ik} \cup S^{jk}_{jk} \cup Q^{ij}_{ij}$. Thus $s'_{ik} = s^{ik}_{ik} + s^{kj}_{kj} + q^{ij}_{ij}$.

(5) Node j is in S'_{ik} and the graph induced on G_{ik} is disjoint. There are 9 distinct subcases.
   (a) All the Z_2^E nodes are contained within G_{jk}. Then $S'_{ik} = P_{ij} \cup S_{jk} \cup Q_{ik}$. In addition, for feasibility we require that that b_{ji} and b_{ki} be zero. Thus $s'_{ik} = p_{ij} + s_{jk} + q_{ik} + b_{ji} + b_{ki}$.
   (b) All the Z_2^E nodes are contained in G_{ij}. Then $S'_{ik} = S_{ij} \cup P_{jk} \cup Q_{ik}$. For feasibility we require that b_{ik} and b_{jk} be zero. Thus $s'_{ik} = s_{ij} + p_{jk} + q_{ik} + b_{ik} + b_{jk}$.
   (c) All the Z_2^E nodes are contained within G_{ik}. Then $S'_{ik} = P_{ij} \cup P_{jk} \cup R_{ik}$. For feasibility we require that b_{kj} and b_{ij} be zero. Thus $s'_{ik} = p_{ij} + p_{jk} + r_{ik} + b_{ij} + b_{kj}$.
   (d) All the Z_2^E nodes are contained within G_{jk} and the k side of the disjoint network induced on G_{ik} by S'_{ik}. This forces a 2^E requirement on k. Thus $S'_{ik} = S^k_{kj} \cup P_{ij} \cup Q^k_{ki}$. For feasibility we require that b_{ji} be zero. Thus $s'_{ik} = s^k_{kj} + p_{ji} + q^k_{ki} + b_{ji}$.
   (e) All the Z_2^E nodes are contained within G_{ij} and the i side of the disjoint network induced on G_{ik} by S'_{ik}. This forces a 2^E requirement on i. Then $S'_{ik} = S^i_{ij} \cup P_{jk} \cup Q^i_{ik}$. For feasibility we require that b_{jk} be zero. Thus $s'_{ik} = s^i_{ij} + p_{jk} + b_{jk} + q^i_{ik}$.
   (f) All the Z_2^E nodes are contained within $G_{ij} \cup G_{jk}$, and there is at least one Z_2^E node within G_{ij} and at least one Z_2^E node within G_{jk}; both distinct from j. This forces a 2^E requirement on j. Then $S'_{ik} = S^j_{ij} \cup S^j_{jk} \cup Q_{ik}$. Thus $s'_{ik} = s^j_{ij} + s^j_{jk} + q_{ik}$.
   (g) G_{jk} contains at least one Z_2^E node, the i side of the disjoint network on G_{ik} induced by S'_{ik} contains at least one Z_2^E node, and the k side contains no Z_2^E node. This forces a 2^E requirement on i and j. Then $S'_{ik} = S^{ij}_{ij} \cup S^j_{jk} \cup Q^i_{ik}$. Thus $s'_{ik} = s^{ij}_{ij} + s^j_{jk} + q^i_{ik}$.
   (h) G_{ij} contains at least one Z_2^E node, the k side of the disjoint network on G_{ik} induced by S'_{ik} contains at least one Z_2^E node, and the i side contains no Z_2^E node. This forces a 2^E requirement on j and k. Then $S'_{ik} = S^j_{ij} \cup S^{jk}_{jk} \cup Q^k_{ik}$. Thus $s'_{ik} = s^j_{ij} + s^{jk}_{jk} + q^k_{ik}$.
   (i) Both the i and k side of the disjoint network induced on G_{ik} by S'_{ik} contain at least one Z_2^E node. This forces a 2^E requirement on i, j and k. Then $S'_{ik} = S^{ij}_{ij} \cup S^{jk}_{jk} \cup Q^{ik}_{ik}$. Thus $s'_{ik} = s^{ij}_{ij} + s^{jk}_{jk} + q^{ik}_{ik}$.

s'^{ik}_{ik}, s'^{i}_{ik}, t'_{ik}, t'^{i}_{ik}, u'_{ik}, p'_{ik}, q'^{ik}_{ik}, q'^{i}_{ik}, q'_{ik}, and r'_{ik} are computed correctly (see appendix).

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

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, that separates two nodes in Z_2^N, then we may decompose the original problem into two smaller problems as follows. Consider the two graphs $G_1 = (N_1, E_1)$ and $G_2 = (N_2, E_2)$, each containing at least one node in Z_2^N distinct from i and j, and with $E_1 \cap E_2 = \phi$, 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 $G_1^* \cup G_2^* \backslash (i, j)$. By repeating this procedure, it is possible to decompose the original LCND problem into a series of smaller LCND problems. Grötschel, Monma and Stoer [9] take this approach (with a slight refinement that considers multiple subproblems, instead of two at a time) to solve to optimality several problems that arise in telecommunications practice. Note, 2-separators in a graph may be found in linear time using an algorithm based on depth first search (see [11]).

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.[3] 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 Z_2^E, and $G_1 = (N_1, E_1)$ and $G_2 = (N_2, E_2)$ the two graphs, each containing at least one node in Z_2^E distinct from i and j, with $E_1 \cap E_2 = \phi$, 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

[3] Similar decompositions to the node-connectivity case may be applied by finding two *edges* whose removal disconnects two nodes in Z_2^E.

the solution is ${P_{ij}}^1 \cup {P_{ij}}^2$.

(2) The network induced on G_2 is disjoint, and the network induced on G_1 is connected. There are three subcases.
   (a) Both the i and j side of the disjoint network contain Z_2^E nodes. Then the solution is $S_{ij}^{ij\,1} \cup Q_{ij}^{ij\,2}$.
   (b) The j side of the disjoint network does not contain Z_2^E nodes. Then the solution is ${S_{ij}^i}^1 \cup {Q_{ij}^i}^2$.
   (c) The i side of the disjoint network does not contain Z_2^E nodes. Then the solution is ${S_{ij}^j}^1 \cup {Q_{ij}^j}^2$.

(3) The network induced on G_1 is disjoint, and the network induced on G_2 is connected. There are three subcases.
   (a) Both the i and j side of the disjoint network contain Z_2^E nodes. Then the solution is $S_{ij}^{ij\,2} \cup Q_{ij}^{ij\,1}$.
   (b) The j side of the disjoint network does not contain Z_2^E nodes. Then the solution is ${S_{ij}^i}^2 \cup {Q_{ij}^i}^1$.
   (c) The i side of the disjoint network does not contain Z_2^E nodes. Then the solution is ${S_{ij}^j}^2 \cup {Q_{ij}^j}^1$.

(4) Node j is not in the solution. Then the solution is ${T_{ij}^i}^1 \cup {T_{ij}^i}^2$.

(5) Node i is not in the solution. Then the solution is ${T_{ij}^j}^1 \cup {T_{ij}^j}^2$.

In [9], Grötschel, Monma and Stoer describe a 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 Z_1 from a node in Z_2^E, but does not separate two nodes in Z_2^E. Let $G_1 = (N_1, E_1)$ be the two graphs with $E_1 \cap E_2 = \phi$, and $N_1 \cap N_2 = \{i, j\}$, with all Z_2^E nodes in N_2, and with $N_1 \backslash \{i, j\}$ containing at least one node in Z_1 and no Z_2^E nodes. Grötschel, Monma and Stoer claim 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 part includes i and one part 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 the graph induced on G_2 is disjoint, forcing both edge disjoint paths between nodes in Z_2^E to go through G_1. Thus the decomposition procedure discussed in Grötschel, Monma and Stoer is incorrect.

Acknowledgement: This work was supported by the Operations Research Department at AT&T Bell Laboratories with the Martin Farber Internship.

References

[1] S. Arnborg and A. Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Applied Mathematics*, 23:11–24, 1989.

[2] A. M. Baïou and A. R. Mahjoub. Steiner 2-edge connected subgraph polytope on series-parallel graphs. *SIAM Journal on Discrete Mathematics*, 10:505–514, 1997.

[3] W. Bern, E. L. Lawler, and L. Wong. Linear time computation of optimal subgraphs of decomposable graphs. *Journal of Algorithms*, 8:216–235, 1987.

[4] J. Bondy and U. Murty. *Graph Theory with Applications*. North-Holland, 1976.

[5] R. H. Cardwell, C. L. Monma, and T. H. Wu. Computer-aided design procedures for survivable fiber optic networks. *IEEE Journal on Selected Areas in Communications*, 7:1188–1197, 1989.

[6] 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.

[7] 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.

[8] S. Even. *Graph Algorithms*. Computer Science Press, Rockville, MD, 1979.

[9] M. Grötschel, C. L. Monma, and M. Stoer. Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. *Operations Research*, 40:309–330, 1992.

[10] M. Grötschel, C. L. Monma, and M. Stoer. Design of survivable networks. In M. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 617–672. North-Holland, 1995.

[11] J. Hopcroft and R. Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2:135–158, 1973.

[12] 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, 2001.

[13] A. R. Mahjoub. Two-edge connected spanning subgraphs and polyhedra. *Mathematical Programming*, 64:199–208, 1994.

[14] K. Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10:96–115, 1927.

[15] S. Raghavan and T. L. Magnanti. Network connectivity. In M. Dell'Amico, F. Maffioli, and S. Martello, editors, *Annotated Bibliographies in Combinatorial Optimization*, pages 335–354. John Wiley & Sons, 1997.

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

[17] P. Winter. Generalized Steiner problems in series-parallel networks. *Journal of Algorithms*, 7:549–566, 1987.

A Appendix

Listing of cases for Theorem 1

Cases for T'_{ik}: There are 5 cases for T'_{ik}.

(1) j is included in T'_{ik}.
   (a) All Z_2^N nodes are in G_{ik}. Then $T'_{ik} = T_{ik} \cup P_{ij} \cup T_{jk}$. In addition, for feasibility we require that a_{ij} and a_{kj} be zero.
   (b) All Z_2^N nodes are in G_{ij}. Then $T'_{ik} = T_{ik} \cup S_{ij} \cup T_{jk}$. For feasibility a_{ik} and a_{jk} must be zero.
   (c) All Z_2^N nodes are in G_{jk}. Then $T'_{ik} = T_{ik} \cup P_{ij} \cup T_{jk}$. For feasibility a_{ji} and a_{ki} must be zero.

(2) j is not in T'_{ik}.
   (a) All Z_2^N nodes are in G_{ik}. Then $T'_{ik} = T_{ik} \cup T_{ij}$. For feasibility m_{jk} and a_{ij} must be zero.
   (b) All Z_2^N nodes are in G_{ij}. Then $T'_{ik} = T_{ik} \cup T_{ij}$. For feasibility m_{jk} and a_{ik} must be zero.

Cases for U'_{ik}: There are 5 cases for U'_{ik}.

(1) j is not in U'_{ik}.
   (a) All Z_2^N nodes are in G_{ik}. Then $U'_{ik} = U_{ik}$. For feasibility m_{ij} and m_{jk} must be zero.
   (b) All Z_2^N nodes are in G_{ij}. Then $U'_{ik} = U_{ij}$. For feasibility m_{ik} and m_{jk} must be zero.
   (c) All Z_2^N nodes are in G_{jk}. Then $U'_{ik} = U_{jk}$. For feasibility m_{ij} and m_{ik} must be zero.

(2) j is included in U'_{ik}.
   (a) All Z_2^N nodes are in G_{ij}. Then $U'_{ik} = T_{ji} \cup T_{jk}$. For feasibility m_{ik} and a_{jk} must be zero.
   (b) All Z_2^N nodes are in G_{jk}. Then $U'_{ik} = T_{ji} \cup T_{jk}$. For feasibility m_{ik} and a_{ji} must be zero.

Cases for P'_{ik}: There are 5 cases for P'_{ik}.

(1) None of the graphs induced on G_{ij}, G_{ik} and G_{jk} is disjoint. Then $P'_{ik} = P_{ik} \cup P_{ij} \cup P_{jk}$.

(2) j is not included in P'_{ik}. Then $P'_{ik} = P_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility a_{ij} and a_{kj} must be zero.

(3) j is included and the graph induced on G_{ik} by P'_{ik} is disjoint. Then $P'_{ik} = P_{ij} \cup P_{jk} \cup Q_{ik}$.

(4) j is included and the graph induced on G_{jk} by P'_{ik} is disjoint. Then $P'_{ik} = P_{ik} \cup P_{ij} \cup Q_{jk}$. For feasibility a_{ij} must be zero.
(5) j is included and the graph induced on G_{ij} by P'_{ik} is disjoint. Then $P'_{ik} = P_{ik} \cup P_{jk} \cup Q_{ij}$. For feasibility a_{kj} must be zero.

Cases for Q'_{ik}: There are 3 cases for Q'_{ik}.
(1) j is not included in Q'_{ik}. Then $Q'_{ik} = Q_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility a_{ij} and a_{kj} must be zero.
(2) j is included in Q'_{ik} and the graph induced on G_{jk} is disjoint. Then $Q'_{ik} = Q_{ik} \cup P_{ij} \cup Q_{jk}$. For feasibility a_{ij} and a_{kj} must be zero.
(3) j is included in Q'_{ik} and the graph induced on G_{ij} is disjoint. Then $Q'_{ik} = Q_{ik} \cup Q_{ij} \cup P_{jk}$. For feasibility a_{ij} and a_{kj} must be zero.

Cases for R'_{ik}: There are 9 cases for R'_{ik}.
(1) j is not in R'_{ik}.
  (a) All Z_2^N nodes are in G_{ik}. Then $R'_{ik} = R_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility a_{ij} and a_{kj} must be zero.
  (b) All Z_2^N nodes are in G_{ij}. Then $R'_{ik} = Q_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility a_{ik} and a_{jk} must be zero.
  (c) All Z_2^N nodes are in G_{jk}. Then $R'_{ik} = Q_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility a_{ji} and a_{ki} must be zero.
(2) j is in R'_{ik} and the graph induced on G_{jk} is disjoint.
  (a) All Z_2^N nodes are in G_{ik}. Then $R'_{ik} = R_{ik} \cup P_{ij} \cup Q_{jk}$. For feasibility a_{ij} and a_{kj} must be zero.
  (b) All Z_2^N nodes are in G_{ij}. Then $R'_{ik} = Q_{ik} \cup S_{ij} \cup Q_{jk}$. For feasibility a_{ik} and a_{jk} must be zero.
  (c) All Z_2^N nodes are in G_{jk}. Then $R'_{ik} = Q_{ik} \cup P_{ij} \cup R_{jk}$. For feasibility a_{ji} and a_{ki} must be zero.
(3) j is in R'_{ik} and the graph induced on G_{ij} is disjoint.
  (a) All Z_2^N nodes are in G_{ik}. Then $R'_{ik} = R_{ik} \cup Q_{ij} \cup P_{jk}$. For feasibility a_{ij} and a_{kj} must be zero.
  (b) All Z_2^N nodes are in G_{ij}. Then $R'_{ik} = Q_{ik} \cup R_{ij} \cup P_{jk}$. For feasibility a_{ik} and a_{jk} must be zero.
  (c) All Z_2^N nodes are in G_{jk}. Then $R'_{ik} = Q_{ik} \cup Q_{ij} \cup S_{jk}$. For feasibility a_{ki} and a_{ji} must be zero.

Listing of cases for Theorem 4

Cases for S'^{ik}_{ik}: There are 7 cases for S'^{ik}_{ik}.
(1) None of the graphs induced on G_{ij}, G_{ik}, G_{jk} by S'^{ik}_{ik} are disjoint. Then $S'^{ik}_{ik} = P_{ik} \cup P_{ij} \cup P_{jk}$.
(2) Node j is not in S'^{ik}_{ik}. Then $S'^{ik}_{ik} = S^{ik}_{ik} \cup T^{i}_{ij} \cup T^{k}_{kj}$.
(3) j is in S'^{ik}_{ik} and the graph induced on G_{jk} is disjoint.
  (a) The j side of the disjoint network induced on G_{jk} by S'^{ik}_{ik} does not contain Z_2^E nodes. Then $S'_{ik} = S^{ik}_{ik} \cup S^{i}_{ij} \cup Q^{k}_{kj}$.
  (b) The j side of the disjoint network induced on G_{jk} by S'^{ik}_{ik} contains

Z_2^E nodes. Then $S'_{ik} = S_{ik}^{ik} \cup S_{ij}^{ij} \cup Q_{jk}^{jk}$.

(4) j is in ${S'}_{ik}^{ik}$ and the graph induced on G_{ij} is disjoint.
   (a) The j side of the disjoint network on G_{ij} induced by ${S'}_{ik}^{ik}$ does not contain Z_2^E nodes. Then ${S'}_{ik}^{ik} = S_{ik}^{ik} \cup S_{jk}^{k} \cup Q_{ij}^{i}$.
   (b) The j side of the disjoint network induced on G_{ij} by ${S'}_{ik}^{ik}$ contains Z_2^E nodes. Then ${S'}_{ik}^{ik} = S_{ik}^{ik} \cup S_{jk}^{jk} \cup Q_{ij}^{ij}$.

(5) j is in ${S'}_{ik}^{ik}$ and the graph induced on G_{ik} is disjoint. The only way for i and k to be two-edge connected is through node j. Then ${S'}_{ik}^{ik} = S_{ij}^{ij} \cup S_{jk}^{jk} \cup Q_{ik}^{ik}$.

Cases for ${S'}_{ik}^{i}$: There are 13 cases for ${S'}_{ik}^{i}$.

(1) None of the graphs induced on G_{ij}, G_{ik}, G_{jk} by ${S'}_{ik}^{i}$ are disjoint. Then ${S'}_{ik}^{i} = P_{ik} \cup P_{ij} \cup P_{jk}$.

(2) Node j is not in ${S'}_{ik}^{i}$.
   (a) G_{jk} does not contain Z_2^E nodes. Then ${S'}_{ik}^{i} = S_{ik}^{i} \cup T_{ij}^{i} \cup T_{kj}$. In addition, for feasibility we require that b_{kj} be zero.
   (b) G_{jk} contains Z_2^E nodes. Then ${S'}_{ik}^{i} = S_{ik}^{ik} \cup T_{ij}^{i} \cup T_{kj}^{k}$.

(3) j is in ${S'}_{ik}^{i}$ and the graph induced on G_{jk} is disjoint.
   (a) G_{jk} does not contain Z_2^E nodes. Then ${S'}_{ik}^{i} = S_{ik}^{i} \cup S_{ij}^{i} \cup Q_{jk}$.
   (b) The j side but not the k side of the disjoint network induced on G_{jk} by ${S'}_{ik}^{i}$ contains Z_2^E nodes. Then ${S'}_{ik}^{i} = S_{ik}^{i} \cup S_{ij}^{ij} \cup Q_{jk}^{j}$.
   (c) The k side but not the j side of the disjoint network induced on G_{jk} by ${S'}_{ik}^{i}$ contains Z_2^E nodes. Then ${S'}_{ik}^{i} = S_{ik}^{ik} \cup S_{ij}^{i} \cup Q_{kj}^{k}$.
   (d) Both the j side and the k side of the disjoint network induced on G_{jk} by ${S'}_{ik}^{i}$ contain Z_2^E nodes. Then ${S'}_{ik}^{i} = S_{ik}^{ik} \cup S_{ij}^{ij} \cup Q_{jk}^{jk}$.

(4) j is in ${S'}_{ik}^{i}$ and the graph induced on G_{ij} is disjoint.
   (a) G_{jk} and the j side of the disjoint network induced on G_{ij} by ${S'}_{ik}^{i}$ do not contain contain Z_2^E nodes. Then ${S'}_{ik}^{i} = S_{ik}^{i} \cup P_{kj} \cup Q_{ij}^{i}$. For feasibility we require that b_{kj} be zero.
   (b) G_{jk} contains Z_2^E nodes, and the j side of the disjoint network on G_{ij} induced by ${S'}_{ik}^{i}$ does not contain contain Z_2^E nodes. Then ${S'}_{ik}^{i} = S_{ik}^{ik} \cup S_{jk}^{k} \cup Q_{ij}^{i}$.
   (c) The j side of the disjoint network induced on G_{ij} by ${S'}_{ik}^{i}$ contains Z_2^E nodes. Then ${S'}_{ik}^{i} = S_{ik}^{ik} \cup S_{jk}^{jk} \cup Q_{ij}^{ij}$.

(5) j is in ${S'}_{ik}^{i}$ and the graph induced on G_{ik} is disjoint.
   (a) G_{jk} and the k side of the disjoint network induced on G_{ik} by ${S'}_{ik}^{i}$ do not contain contain Z_2^E nodes. Then ${S'}_{ik}^{i} = S_{ij}^{i} \cup P_{jk} \cup Q_{ik}^{i}$. For feasibility b_{jk} must be zero.
   (b) G_{jk} contains Z_2^E nodes, and the k side of the disjoint network on G_{ik} induced by ${S'}_{ik}^{i}$ does not contain contain Z_2^E nodes. Then ${S'}_{ik}^{i} = S_{ij}^{ij} \cup S_{jk}^{j} \cup Q_{ik}^{i}$.
   (c) The k side of the disjoint network induced on G_{ik} by ${S'}_{ik}^{i}$ contains Z_2^E nodes. Then ${S'}_{ik}^{i} = S_{ij}^{ij} \cup S_{jk}^{jk} \cup Q_{ik}^{ik}$.

Cases for T'_{ik}: There are 9 cases for T'_{ik}.

(1) j is not included in T'_{ik}.
   (a) All the Z_2^E nodes are contained within G_{ik}. Then $T'_{ik} = T_{ik} \cup T_{ij}$. For feasibility m_{jk} and b_{ij} must be zero.
   (b) All the Z_2^E nodes are contained within G_{ij}. Then $T'_{ik} = T_{ik} \cup T_{ij}$. For feasibility m_{jk} and b_{ik} must be zero.
   (c) All the Z_2^E nodes are contained within $G_{ij} \cup G_{ik}$, and there is at least one Z_2^E node within G_{ij} and at least one Z_2^E node within G_{ik}; both distinct from i. Then $T'_{ik} = T^i_{ik} \cup T^i_{ij}$. For feasibility m_{jk} must be zero.

(2) j is included in T'_{ik}.
   (a) All the Z_2^E nodes are contained within G_{ik}. Then $T'_{ik} = T_{ik} \cup P_{ij} \cup T_{jk}$. For feasibility b_{ij} and b_{kj} must be zero.
   (b) All the Z_2^E nodes are contained within G_{ij}. Then $T'_{ik} = T_{ik} \cup S_{ij} \cup T_{jk}$. For feasibility b_{ik} and b_{jk} must be zero.
   (c) All the Z_2^E nodes are contained within G_{jk}. Then $T'_{ik} = T_{ik} \cup P_{ij} \cup T_{jk}$. For feasibility b_{ji} and b_{ki} must be zero.
   (d) All the Z_2^E nodes are contained within $G_{ij} \cup G_{ik}$, and there is at least one Z_2^E node within G_{ij} and at least one Z_2^E node within G_{ik}; both distinct from i. Then $T'_{ik} = T^i_{ik} \cup S^i_{ij} \cup T_{jk}$. For feasibility b_{jk} must be zero.
   (e) All the Z_2^E nodes are contained within $G_{ij} \cup G_{jk}$, and there is at least one Z_2^E node within G_{ij} and at least one Z_2^E node within G_{jk}; both distinct from j. Then $T'_{ik} = T_{ik} \cup S^j_{ij} \cup T^j_{jk}$. For feasibility b_{ik} must be zero.
   (f) Both G_{ik} and G_{jk} contain Z_2^E nodes. Then $T'_{ik} = T^i_{ik} \cup S^{ij}_{ij} \cup T^j_{jk}$.

Cases for T'^i_{ik}: There are 3 cases for T'^i_{ik}.

(1) j is not included in T'^i_{ik}. Then $T'^i_{ik} = T^i_{ik} \cup T^i_{ij}$. For feasibility m_{jk} must be zero.

(2) j is included in T'^i_{ik}.
   (a) G_{jk} does not contain Z_2^E nodes. Then $T'^i_{ik} = T^i_{ik} \cup S^i_{ij} \cup T_{jk}$. For feasibility b_{jk} must be zero.
   (b) G_{jk} contains Z_2^E nodes. Then $T'^i_{ik} = T^i_{ik} \cup S^{ij}_{ij} \cup T^j_{jk}$.

Cases for U'_{ik}: There are 6 cases for U'_{ik}.

(1) j is not in U'_{ik}.
   (a) All Z_2^E nodes are contained within G_{ik}. Then $U'_{ik} = U_{ik}$. For feasibility m_{ij} and m_{kj} must be zero.
   (b) All Z_2^E nodes are contained within G_{ij}. Then $U'_{ik} = U_{ij}$. For feasibility m_{ik} and m_{jk} must be zero.
   (c) All Z_2^E nodes are contained within G_{jk}. Then $U'_{ik} = U_{jk}$. For feasibility m_{ji} and m_{ki} must be zero.

(2) j is in U'_{ik}.
   (a) All Z_2^E nodes are contained within G_{ij}. Then $U'_{ik} = T_{ji} \cup T_{jk}$. For feasibility m_{ik} and b_{jk} must be zero.
   (b) All Z_2^E nodes are contained within G_{jk}. Then $U'_{ik} = T_{ji} \cup T_{jk}$. For

feasibility m_{ik} and b_{ji} must be zero.

(c) All Z_2^E nodes are contained within $G_{ij} \cup G_{jk}$, and there is at least one Z_2^E node within G_{ij} and at least one Z_2^E node within G_{jk}; both distinct from j. Then $U'_{ik} = T^j_{ji} \cup T^j_{jk}$. For feasibility m_{ik} must be zero.

Cases for P'_{ik}: There are 7 cases for P'_{ik}.

(1) None of the networks induced by P'_{ik} on G_{ik}, G_{ij} and G_{jk} is disjoint. Then $P'_{ik} = P_{ik} \cup P_{ij} \cup P_{jk}$.

(2) j is not in P'_{ik}. Then $P'_{ik} = P_{ik} \cup T^i_{ij} \cup T^k_{kj}$.

(3) j is in P'_{ik} and the network induced on G_{jk} is disjoint.

(a) No Z_2^E nodes are present on the j side of the disjoint network induced on G_{jk}. Then $P'_{ik} = P_{ik} \cup S^i_{ij} \cup Q^k_{kj}$.

(b) Z_2^E nodes are present on the j side of the disjoint network induced on G_{jk}. Then $P'_{ik} = P_{ik} \cup S^{ij}_{ij} \cup Q^{kj}_{kj}$.

(4) j is in P'_{ik} and the network induced on G_{ij} is disjoint.

(a) No Z_2^E nodes are present on the j side of the disjoint network induced on G_{ij}. Then $P'_{ik} = P_{ik} \cup S^k_{kj} \cup Q^i_{ij}$.

(b) Z_2^E nodes are present on the j side of the disjoint network induced on G_{ij}. Then $P'_{ik} = P_{ik} \cup S^{kj}_{kj} \cup Q^{ij}_{ij}$.

(5) j is in P'_{ik} and the network induced on G_{ik} is disjoint. Then $P'_{ik} = P_{ij} \cup P_{jk} \cup Q^{ik}_{ik}$.

Cases for Q'_{ik}: There are 3 cases for Q'_{ik}.

(1) j is not in Q'_{ik}. Then $Q'_{ik} = Q_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility b_{ij} and b_{kj} must be zero.

(2) j is in Q'_{ik}, and the graph induced on G_{jk} by Q'_{ik} is disjoint. Then $Q'_{ik} = Q_{ik} \cup P_{ij} \cup Q_{jk}$. For feasibility b_{ij} and b_{kj} must be zero.

(3) j is in Q'_{ik}, and the graph induced on G_{ij} by Q'_{ik} is disjoint. Then $Q'_{ik} = Q_{ik} \cup Q_{ij} \cup P_{kj}$. For feasibility b_{ij} and b_{kj} must be zero.

Cases for Q'^i_{ik}: There are 4 cases for Q'^i_{ik}

(1) j is not in Q'^i_{ik}. Then $Q'^i_{ik} = Q^i_{ik} \cup T^i_{ij} \cup T_{kj}$. For feasibility b_{kj} must be zero.

(2) j is in Q'^i_{ik}, and the graph induced on G_{ij} is disjoint. Then $Q'^i_{ik} = Q^i_{ik} \cup Q^i_{ij} \cup P_{kj}$. For feasibility b_{kj} must be zero.

(3) j is in Q'^i_{ik}, and the graph induced on G_{jk} is disjoint.

(a) No Z_2^E nodes are present in the j side of the disjoint network induced on G_{jk}. Then $Q'^i_{ik} = Q^i_{ik} \cup S^i_{ij} \cup Q_{jk}$.

(b) Z_2^E nodes are present in the j side of the disjoint network induced on G_{jk}. Then $Q'^i_{ik} = Q^i_{ik} \cup S^{ij}_{ij} \cup Q^j_{jk}$.

Cases for Q'^{ik}_{ik}: There are 5 cases for Q'^{ik}_{ik}.

(1) j is not in Q'^{ik}_{ik}. Then $Q'^{ik}_{ik} = Q^{ik}_{ik} \cup T^i_{ij} \cup T^k_{kj}$.

(2) j is in Q'^{ik}_{ik} and the network induced on G_{jk} is disjoint.

(a) No Z_2^E nodes are present in the j side of the disjoint network induced on G_{jk}. Then $Q'^{ik}_{ik} = Q^{ik}_{ik} \cup S^i_{ij} \cup Q^k_{kj}$.

(b) Z_2^E nodes are present in the j side of the disjoint network induced

on G_{jk}. Then $Q'^{ik}_{ik} = Q^{ik}_{ik} \cup S^{ij}_{ij} \cup Q^{kj}_{kj}$.

(3) j is in Q'^{ik}_{ik} and the network induced on G_{ij} is disjoint.

(a) No Z_2^E nodes are present in the j side of the disjoint network induced on G_{ij}. Then $Q'^{ik}_{ik} = Q^{ik}_{ik} \cup S^{k}_{kj} \cup Q^{i}_{ji}$.

(b) Z_2^E nodes are present on the j side of the disjoint network induced on G_{ij}. Then $Q'^{ik}_{ik} = Q^{ik}_{ik} \cup S^{kj}_{kj} \cup Q^{ji}_{ji}$.

Cases for R'_{ik}: There are 19 cases for R'_{ik}.

(1) j is not in R'_{ik}.

(a) All Z_2^E nodes are contained within G_{ik}. Then $R'_{ik} = R_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility b_{ij} and b_{kj} must be zero.

(b) All Z_2^E nodes are contained within G_{ij}. Then $R'_{ik} = Q_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility b_{ik} and b_{jk} must be zero.

(c) All Z_2^E nodes are contained within G_{jk}. Then $R'_{ik} = Q_{ik} \cup T_{ij} \cup T_{kj}$. For feasibility b_{ji} and b_{ki} must be zero.

(d) All the Z_2^E nodes are contained within G_{ij} and the i part of the disjoint network induced on G_{ik}; and both parts contain a Z_2^E node distinct from i. Then $R'_{ik} = Q^i_{ik} \cup T^i_{ij} \cup T_{kj}$. For feasibility b_{jk} must be zero.

(e) All the Z_2^E nodes are contained within G_{jk} and the k part of the disjoint network induced on G_{ik}; and both parts contain a Z_2^E node distinct from k. Then $R'_{ik} = Q^k_{ik} \cup T^k_{kj} \cup T_{ij}$. For feasibility b_{ji} must be zero.

(2) j is in R'_{ik} and the network induced on G_{ij} by R'_{ik} is disjoint.

(a) All Z_2^E nodes are contained within G_{ik}. Then $R'_{ik} = R_{ik} \cup Q_{ij} \cup P_{kj}$. For feasibility b_{ij} and b_{kj} must be zero.

(b) All Z_2^E nodes are contained within G_{ij}. Then $R'_{ik} = Q_{ik} \cup R_{ij} \cup P_{kj}$. For feasibility b_{ik} and b_{jk} must be zero.

(c) All Z_2^E nodes are contained within G_{jk}. Then $R'_{ik} = Q_{ik} \cup Q_{ij} \cup S_{kj}$. For feasibility b_{ji} and b_{ki} must be zero.

(d) All the Z_2^E nodes are contained within the i part of the disjoint networks induced on G_{ij} and G_{ik}; and both parts contain a Z_2^E node distinct from i. Then $R'_{ik} = Q^i_{ik} \cup Q^i_{ij} \cup P_{jk}$. 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 Z_2^E.)

(e) All the Z_2^E nodes are contained within G_{jk} and the k part of the disjoint network induced on G_{ik}; and both parts contain a Z_2^E node distinct from k. Then $R'_{ik} = Q^k_{ik} \cup Q_{ij} \cup S^k_{kj}$. For feasibility b_{ji} must be zero.

(f) All the Z_2^E nodes are contained within G_{jk} and the j part of the disjoint network induced on G_{ij}; and both parts contain a Z_2^E node distinct from j. Then $R'_{ik} = Q_{ik} \cup Q^j_{ij} \cup S^j_{kj}$. For feasibility b_{ki} must be zero.

(g) The k but not the i side of the disjoint network induced on G_{ik} contains Z_2^E nodes, and the j side but not the i side of the disjoint

network induced on G_{ij} contains Z_2^E nodes. Then $R'_{ik} = Q^k_{ik} \cup Q^j_{ij} \cup S^{kj}_{kj}$. For feasibility y_i must be zero.

(3) j is in R'_{ik} and the network induced on G_{jk} by R'_{ik} is disjoint.
   (a) All Z_2^E nodes are contained within G_{ik}. Then $R'_{ik} = R_{ik} \cup P_{ij} \cup Q_{jk}$. For feasibility b_{ij} and b_{kj} must be zero.
   (b) All Z_2^E nodes are contained within G_{ij}. Then $R'_{ik} = Q_{ik} \cup S_{ij} \cup Q_{jk}$. For feasibility b_{ik} and b_{jk} must be zero.
   (c) All Z_2^E nodes are contained within G_{jk}. Then $R'_{ik} = Q_{ik} \cup P_{ij} \cup R_{jk}$. For feasibility b_{ji} and b_{ki} must be zero.
   (d) All the Z_2^E nodes are contained within the k part of the disjoint networks induced on G_{ik} and G_{jk}; and both parts contain a Z_2^E node distinct from k. Then $R'_{ik} = Q^k_{ik} \cup P_{ij} \cup Q^k_{jk}$. For feasibility b_{ij} and b_{ji} must be zero.
   (e) All the Z_2^E nodes are contained within G_{ij} and the i side of the disjoint network induced on G_{ik}; and both parts contain a Z_2^E node distinct from i. Then $R'_{ik} = Q^i_{ik} \cup S^i_{ij} \cup Q_{jk}$. For feasibility b_{jk} must be zero.
   (f) All the Z_2^E nodes are contained within G_{ij} and the j side of the disjoint network induced on G_{jk}; and both parts contain a Z_2^E node distinct from j. Then $R'_{ik} = Q_{ik} \cup S^j_{ij} \cup Q^j_{jk}$. For feasibility b_{ik} must be zero.
   (g) The i side but not the k side of the disjoint network induced on G_{ik} contains Z_2^E nodes, and the j side but not the k side of the disjoint network induced on G_{jk} contains Z_2^E nodes. Then $R'_{ik} = Q^i_{ik} \cup S^{ij}_{ij} \cup Q^j_{jk}$. For feasibility y_k must be zero.