Issues in the Design and Analysis of Survivable Networks

by

Kalyan T. Talluri

Bachelor of Engineering
College of Engineering, Osmania University, Hyderabad
June 1985

Master of Science in Industrial Engineering
Purdue University
December 1987

Submitted to the Operations Research Center
in Partial Fulfillment of
the Requirements for the Degree of
Doctor of Philosophy in Operations Research

at the

Massachusetts Institute of Technology

September 1991

Signature of Author [illegible]
Operations Research Center
30 August 1991

Certified by [illegible]
Thomas L. Magnanti
George Eastman Professor of Management Science
Sloan School of Management
Thesis Supervisor

Accepted by [illegible]
Thomas L. Magnanti
George Eastman Professor of Management Science
Codirector, Operations Research Center

0.1 Acknowledgments

I am very grateful to my advisor Prof.Tom Magnanti for his support and advice during my stay at M.I.T. The way this thesis has turned out owes much to him. I have learned a lot and benefited very much from my interactions with Prof.Jim Orlin. I thank him for agreeing to serve on my committee and for advice and help throughout. Prof.Anant Balakrishnan supported me at a crucial time and I am very much indebted to him. Doing research with him was a learning experience for me and hopefully I am more appreciative of being practical. Although he was not involved with the research in this thesis, I owe a lot to Prof.Dimitri Bertesekas. He was an exemplary teacher, and I have benefited tremendously from being a teaching assistant for him. His faith in me and his friendship means more to me than he will ever know. I was very fortunate in having Michel Goemans as my office-mate. His infectious enthusiasm for the subject often was a big boost to my morale. Specifically, Chapter 3 of this thesis was joint work with Michel and I enjoyed working with him very much. I would like to acknowledge my gratitude to two of my former teachers from Purdue, Don Wagner and Prof.Vijaya Chandru, for showing great interest in me, and for encouraging me to pursue research in Combinatorial Optimization.

I enjoyed my stay at the O.R.C immensely. I have met a remarkable group of people during my stay here. They have made these four years a rich and unforgettable experience and I consider myself fortunate to have known them.

Thanks also to Laura and Michele for taking care of many of my administrative problems.

My cousins T.L.N and Padma up north and Sarath and Neeru down south have been a great source of support to me during my stay here. I thank them for their warmth and hospitality.

Finally, to my mother, in whose drive and will to see me succed, lies the genesis of this thesis and to my late father, who would have been plenty proud, I would like to dedicate my thesis.

Research partially supported by NSF Grant number DDM-8996120.

Issues in the Design and Analysis of Survivable Networks

by

Kalyan T.Talluri

Submitted to the Operations Research Center
on 30 August 1991
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

In this thesis we study structural properties, techniques, and solution algorithms for the survivable network design problem (SNDP). The problem is to design a network of minimum cost that satisfies given connectivity requirements. We study this problem when the objective is to design a minimum-cost k edge-connected network that spans a set of terminals located in a "convex-hull" fashion on the plane, assuming we use straight lines as edges to connect the points. We derive a structural property relating paths in a planar graph to its boundary and show that the optimal solution on the plane is $k/2$ copies of the boundary when k is even and $k - 1/2$ copies of the boundary plus one copy of the Steiner tree when k is odd; this leads to a fully polynomial approximation to this problem. In the third chapter, we derive a "no-crossing" property of a minimum-cost k edge connected network for terminals on the plane when new points are not allowed; this property essentially says that in the minimum-cost network, two edges do not cross each other. This result is useful in designing local improvement heuristics. In the fourth chapter we study the network synthesis problem, which can be thought of as a linear programming relaxation of SNDP on a complete graph. We give a new algorithm to solve this problem that has the property that it is guaranteed to use fewer or an equal number of edges in the optimal network design than either Gomory and Hu's algorithm or a more recent algorithm due to Gusfield. We also give a lower bound on the number of edges needed in any solution of the network synthesis problem. Finally, in chapter five we study a problem that has applications in the computation of network reliablity: finding k disjoint $[s,t]$ cuts. We give a path formulation of a polytope whose vertices are the incidence vectors of subgraphs that contain exactly k disjoint $[s,t]$ dicuts. This formulation is similar in spirit to the path formulation of the max-flow problem. Our results also give a path formulation of a recent generalization of the max-flow problem due to Nishihara and Inoue and Orlin and Wagner.

Thesis Supervisor: Thomas L. Magnanti

Title: George Eastman Professor of Management Science
Sloan School of Management

Contents

List of Figures

Chapter 1

Introduction

This thesis is concerned with the design and analysis of telecommunication networks. Telecommunication networks are vulnerable to failure. Here failure means that in the network, one or more customers are unable to communicate with other customers. Failures usually occur because accidents or natural causes destroy the cables or transmission nodes. In order to maintain communication despite failures, we would like to build the network with alternate routes for communication that we could use when some equipment fails. Such redundancy in the system reduces the chances of failure, but at a greater overall network cost. Thus, we are led to the problem of designing a minimum-cost network that meets certain connectivity requirements.

The problem has taken on even more importance in the recent past. New fiber optic technology has begun to replace traditional copper cables for transmission links. This new technology is reliable and cost-effective, and has an enoromously high transmission capacity. For example, a strand of copper cable can carry around twenty four conversations at a time, while a fiber optic cable can carry up to twenty thousand conversations at a time. While the low capacity of copper cables has led to dense networks, in which the failure of a few links has an insignificant impact, fiber optic networks are sparse and each cable carries a large amount of traffic, so failures are catastrophic (see [49], [50], [51], [52], [53]). Consequently, planners need to trade off the potential for lost revenues and customer goodwill against the extra costs required to increase the network's survivability.

In this thesis we study structural properties, techniques, and solution algorithms for the network survivability problem. In this Chapter, in Section 1.1 we first state some basic definitions that we use throughout the thesis. In Section 1.2 we state the Survivable Network Design Problem and survey previous research on this problem and its special cases. In Section 1.3 we state a mathematical programming formulation that provides a framework for studying this class of problems. In Section 1.4 we formulate the network reliability problem that is more concerned with analyzing an existing network, rather than designing a survivable network. Finally, in Section 1.5 we outline the rest of the thesis in some detail.

1.1 Basic definitions and notation

In this section we state some basic notation and definitions that we will need in the rest of the thesis. A set V of given *nodes* represents the locations of the switches (offices) that must be connected by a network that provides the desired services. A collection E of *edges* represents the possible pairs of nodes between which we can place a direct transmission link. Therefore, $G = (V, E)$ is an undirected graph of possible direct link connections. We assume that G can have multiple edges, but no loops. Each edge $e \in E$ has a nonnegative *fixed cost* c_e that we incur if we use this edge to establish a direct link connection in the network. The cost of establishing a network consisting of a subset $F \subseteq E$ of edges is $c(F) = \sum_{e \in F} c_e$. This cost for establishing the network topology for the communication network includes the installation of conduits in which to lay the fiber optic cables, the placement of the cables into service, and other related costs. These fixed costs do not include the costs incurred while operating the network, such as routing, multiplexing, and repeater costs.

For a pair of distinct nodes $s, t \in V$, an $[s,t]$-*path* P is a sequence of nodes and edges $(v_0, e_1, v_1, e_2, \ldots, v_{l-1}, e_l, v_l)$, with each edge e_i incident to the nodes v_{i-1} and v_i $(i = 1, \ldots, l)$, $v_0 = s$ and $v_l = t$, and no node or edge appears more than once in P. When there is no room for confusion, we specify the path just by its nodes. A collection $P_1, P_2, \ldots, P_k$ of $[s,t]$-paths is *edge-disjoint* if no edge appears in more than one path, and is *node-disjoint* if no node (except for s and t) appears in more than one path. A graph with at least two nodes is *k-edge* (*k-node*) *connected* if the network contains k edge-disjoint (node-disjoint) paths between every pairs of distinct nodes.

Let $[W, W']$ denote the subset of E with one endpoint in W and the other in W'. An *edge cut* of G is a subset of E of the form $[W, \bar{W}]$ formed by a nonempty proper subset W of V and $\bar{W} = V \setminus W$. A *k-edge cut* is an edge cut of cardinality k. By a classical theorem of Menger (see [5]), G is k-(edge) connected if and only if every edge cut has at least k edges. An edge cut is *minimal* if the graph created by the removal of the edges of the cut has exactly two components.

For a given set of special nodes Z, known as *terminals*, a *Steiner cut* or *Z-cut* is a cut $[S, \bar{S}]$ with the property that both $S \cap Z$ and $\bar{S} \cap Z$ are nonempty.

For a given embedding of the graph G on the plane, two edges of G are said to be *crossing* if they cross in the embedding. If the embedding has no crossing edges, we say that it is a *planar embedding.* A graph is *planar* if it has a planar embedding. The *boundary* of a planar graph is the boundary of the unbounded face.

For a given embedding of a 2-connected planar graph G, the boundary of the graph itself is a 2-connected subgraph of G.

1.2 The survivable network design problem

In this section we state the Survivable Network Design Problem and survey the literature on it. To each node of the network, we associate a nonnegative integer r_i called the *survivability requirement.* Our goal is to build a minimum-cost network with

$$r_{ij} := \min(r_i, r_j)$$

edge-disjoint $[s,t]$-paths between each pair of distinct nodes $i, j \in V$. The *Survivable Network Design Problem (SNDP)* is as follows:

GIVEN: A network $G = (V, E)$ with edge costs $c_e \geq 0, e \in E$ and survivability requirements $r_i, i \in V$.

FIND: A minimum-cost subnetwork $H = (V, F)$ of G that has r_{ij} edge-disjoint paths between every pair of nodes i and j.

By applying small perturbations to the data, if necessary, we assume that $c_e > 0$ for all edges in the network. We impose this assumption for simplicity.

Note that we are studying the edge-connected version of the problem. Although we could very well have defined a node-connected version of the problem or a version that requires both node and edge-connectivities, this thesis primarily focuses on the edge-connected version. When designing a network, we have the option of designing a network with multiple copies of any edge. For designing vertex-connected networks, multiple edges are redundant; when we require only edge-disjoint paths, we have two models - one that prohibits multiple edges and one that allows multiple edges. We will work with the latter model. When we have connectivity requirements of two or three between all pairs of terminals, our results will also show how to design minimum-cost networks under the former model (in fact even for the vertex-connected version of the problem). Thus, for low connectivities our results solve all three problems - edge-connected version with and without allowing multiple edges as well as the vertex-connected version. If we use l_e copies of an edge, we assign a *multiplicity number* l_e to that edge. The cost for these copies would, of course, be $l_e c_e$. Moreover for the Euclidean version of the problem, which we define later in this section, studying the problem with duplication of edges is unavoidable because we can always replace two copies of an edge by two parallel edges that are very close to each other. In passing, we might note that typically in practice $r_i \in \{0, 1, 2\}$ for all $i \in V$.

When $r_i = 1$ for all $i \in V$, the SNDP is the *Spanning tree problem on networks.* For this case, the research community has developed very efficient algorithms for solving the problem: see Kruskal [40] or Prim [56]. When $r_i = 1$ for $i \in Z \subset V$ and $r_i = 0$ for $i \notin Z$, the problem becomes the *Network Steiner tree problem* with terminal set Z. The nodes in $V \setminus Z$ are called *Steiner points.* See Winter [62] for a recent survey of algorithms for the Steiner tree problem in networks. This problem is NP-hard, so the SNDP itself is NP-hard. When $r_i = k$ for all $i \in Z \subset V$, the problem is known as the *k-connected Steiner Network Problem (k-SNP).* Goemans and Talluri [27], Monma and Shallcross [48], Monma, Munson and Pulleyblank [47], Steiglitz, Weiner and Kleitman [59] and Goemans [25] have studied structural properties of minimum-cost k-connected networks. Figure 1.1 shows a network and optimal solutions to all these problems.

When these problems are defined on the plane, we are given a set of points on the plane designated as terminals and we want to find a minimum-cost spanning network using straight lines between the points as edges, the cost of an edge being its length. Many combinatorial problems that can be solved in polynomial time in

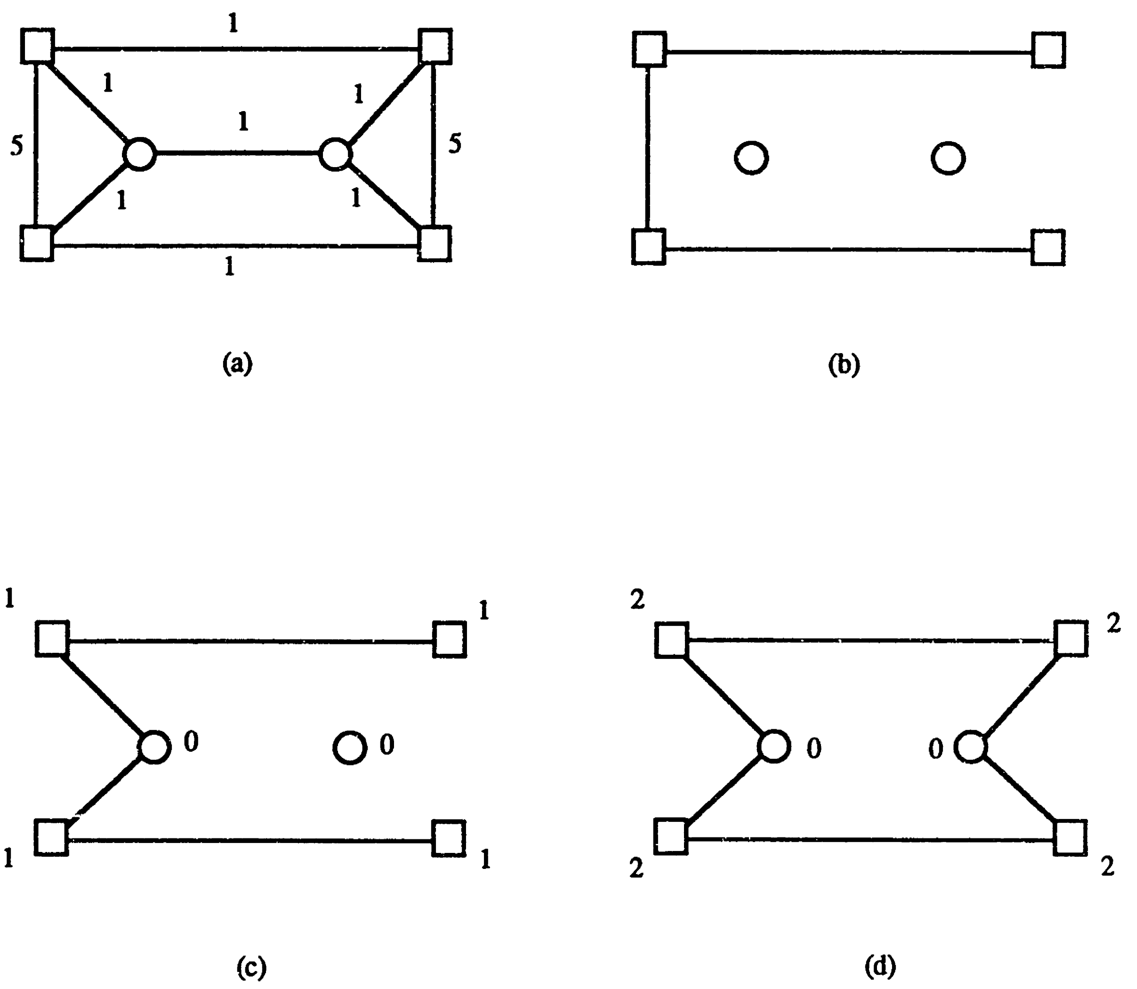


Figure 1.1: (a) The network with costs on edges (b) Minimum-cost spanning tree on the terminal nodes (c) Steiner Tree (d) Minimum-cost 2-connected Steiner tree.

networks become computationally easier to solve when posed on the plane with Euclidean distances. For example, specialized algorithms for both the spanning tree and the matching problem on the plane have better running times than general purpose versions of these algorithms. Problems that are NP-hard, however, tend to remain equally hard or become harder when defined on the plane. The Travelling Salesman Problem and the Steiner tree problem on the plane are examples. The added complexity of these problems, apart from their combinatorial difficulties, arises because of irrational numbers. For example, for the Euclidean Steiner tree problem, the choice of Steiner points is now uncountable and their location unknown. Thus, even formulating this problem as an integer linear program is impossible and we cannot apply traditional integer programming techniques like Lagrangian relaxation or polyhedral combinatorics. Some structural properties for this problem are known. Some, like the property that every Steiner point has degree three with the edges meeting at an angle of 120 degrees (Gilbert and Pollack [24]), are local; some decomposition results (as in Cockayne [8], Hwang et. al. [37]) help in reducing the problem size. Apart from trivial cases, and for problems with fewer than five terminals, for no interesting special cases can this problem can be solved efficiently. Figure 1.2 shows Euclidean Steiner trees for three, four, five, and six nodes. We denote the Euclidean version of k-SNP by k-SNPP (k-Steiner network problem on the plane).

1.3 Mathematical programming formulation

In this section we give a mathematical programming formulation for SNDP. Although the main concern of this thesis is not polyhedral in nature, the formulation will provide a framework for visualizing all the models that we are considering.

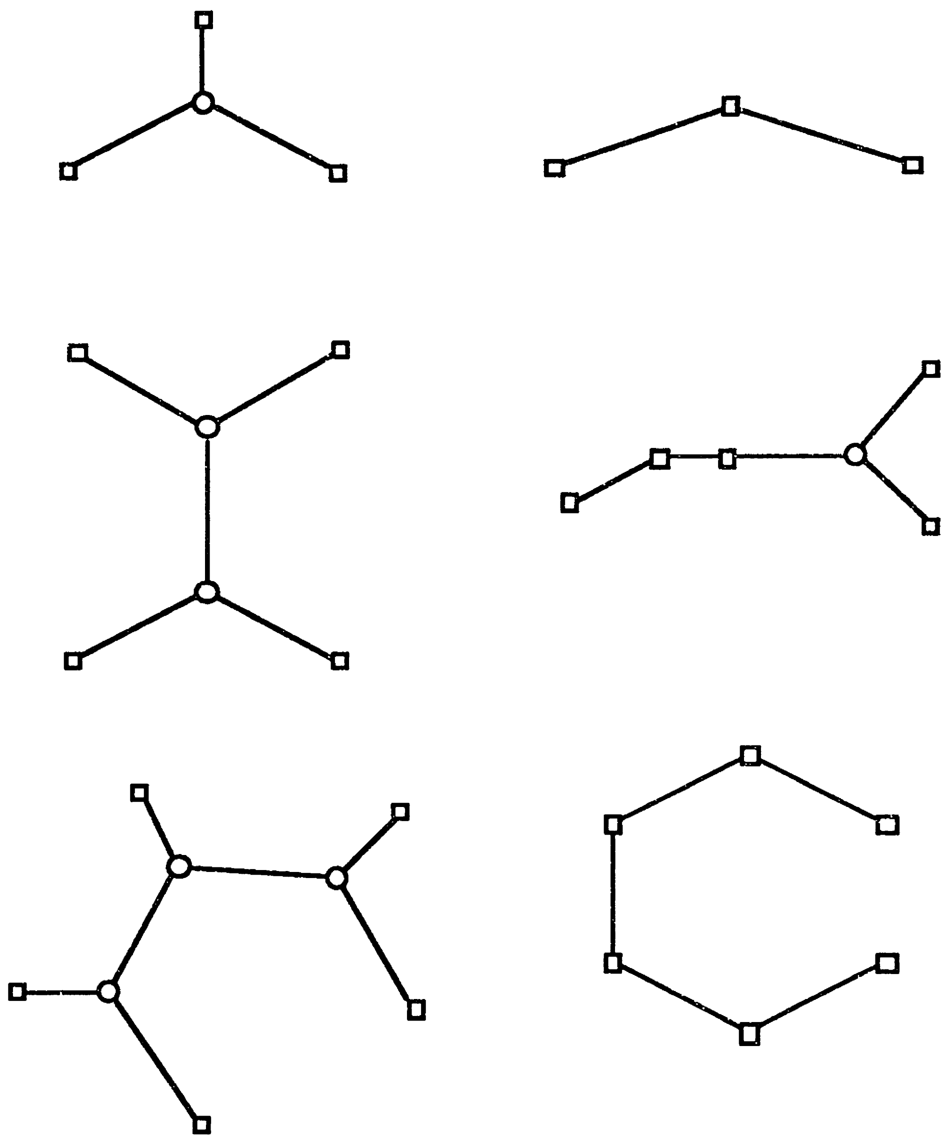


Figure 1.2: Examples of Euclidean Steiner trees: Square nodes are terminals, round nodes are Steiner points.

Let us define two connectivity terms for subsets of nodes:

$$\begin{aligned} r(W) &:= \max\{r_i \mid i \in W\} \quad \text{for all } W \subseteq V, \text{ and} \\ con(W) &:= \max\{r_{ij} \mid i \in W, j \in V \setminus W\} \\ &= \min\{r(W), r(V \setminus W)\} \quad \text{for all } W \subseteq V, \emptyset \neq W \neq V. \end{aligned}$$

For each edge $e \in E$, define a variable x_e and consider the vector space $\Re^E$. Every subset $F \subseteq E$ induces an *incidence vector* $x^F = (x_e^F)_{e \in F} \in \Re^E$ by setting $x_e^F := 1$ if $e \in F$, and $x_e^F := 0$ otherwise. In the other direction, each 0/1-vector $x \in \Re^E$ induces a subset $F^x := \{e \in E | x_e = 1\}$ of the edge set E of G. Using this notation, we formulate SNDP as the following integer program:

$$\begin{array}{llll} & Z = & Min \sum_{e \in E} c_e x_e & \\ & & \text{subject to} & \\ (P) & & \sum_{e \in [W,\bar{W}]} x_e \geq con(W) & \forall\, W \subset V,\ W \neq \emptyset \\ & & 0 \leq x_e,\ x_e \text{ integer} & \forall\, e \in E. \end{array}$$

By Menger's theorem, any feasible solution of this program represents the incidence vector of a subgraph with r_{ij} edge-disjoint paths between nodes $i, j \in V$. If we add the constraint

$$x_e \leq 1 \quad \text{for all } e \in E$$

to this formulation we obtain SNDP without duplication of edges. When G is a complete graph, the linear programming relaxation of this integer program (that is, the problem created by removing the restriction that x_e be integer for all $e \in E$) is called the *network synthesis problem*; Gomory and Hu [29] first studied this problem - we will be studying it in Chapter 4.

In closing this section, we mention the approach taken in [31], [32], [33]. The idea is to devise good valid inequalities for the integer polytope and solve the problem by branch and bound techniques. This approach has been found to be very succesful for solving a number of combinatorial optimization problems such as the Traveling Salesman problem. In [32], Grōtschel, Monma and Stoer use this cutting plane approach to solve a number of real-world connectivity problems. Another succesful computational approach has been taken in Wong [65] and Balakrishnan, Magnanti and Wong [1]. These authors apply an ascent procedure to the dual of the linear programming relaxations of various integer programming formulations of the network design problem. This approach generates solutions that are guaranteed to be within one to five percent of the optimal value for a number of test problems.

1.4 The network reliability problem

So far we have been concerned with building a network from scratch. Next, we study the problem of analyzing an existing network to determine its reliability. In this model we assume that each edge of the network fails with a certain probability. Moreover, we assume that all of the probabilities specified are statistically independent. Network reliability is concerned with the probability that the network is able to carry out some desired operation. The following list describes some common network operations.

- For two specified nodes $s, t \in V$, the *two-terminal reliability* denoted by $Rel_2(G)$ is the probability that the network G contains a $[s, t]$-path.
- The *all-terminal reliability* $Rel_A(G)$ is the probability that for every pair v_1, v_2 of nodes, the network contains a path from v_1 to v_2; equivalently, the all-terminal reliability is the probability that the graph contains at least one spanning tree.
- The final network operation generalizes both of the previous two and involves pairwise communication of k specified nodes, $2 \leq k \leq n$. The *k-terminal*

reliability $Rel_k(G)$ is the probability that for k specified *target* nodes, the graph contains paths between each pair of the k nodes. This is equivalent to asking for the probability that the graph contains a Steiner tree.

In addition to these three measures, we might define a number of other reasonable reliability measures. Network Reliability theory studies methods to calculate these probabilities. For general graphs, almost all these measures are too difficult to calculate exactly. Calculation of most of the interesting reliabilities has been proven to be #P-hard, a complexity result roughly equivalent to being NP-hard for enumeration problems. Thus, much of the research in this area is geared towards obtaining good, quickly computable bounds for the measures. For a thorough survey of all the literature and techniques, see Colbourn's recent book on the combinatorics of network reliability [9]. Chapter 5 of this thesis studies a problem that arose from an attempt to obtain bounds for $Rel_2(G)$.

1.5 Thesis outline

The contribution of this thesis are new structural results that help in designing algorithms for the survivable network design problem. For example, two immediate consequences of these structural properties are a fully polynomial approximation scheme for k-SNDP on the plane and theoretical justification for the use of 2-opt heuristics for the k-connected network design problem on the plane without the use of Steiner points. We have also investigated some theoretical properties that lead to a better understanding of the problem of finding k disjoint $[s,t]$ cuts in a network.

The thesis is organized as follows:

In Chapter 2 we study the k-connected Steiner network design problem on planar graphs when all the terminals lie on the outer face of the graph, and the k-SNPP for a special "convex-hull" type of configuration of terminals on the plane. We derive a structural property relating paths in a planar graph to its boundary and show that the optimal solution on the plane is $k/2$ copies of the boundary wher. k is even and $k-1/2$ copies of the boundary plus one copy of the Steiner tree when k is odd; thus when k is even, the design of minimum-cost k-edge connected networks is very easy, and when k is odd, using a result due to Provan, there is a fully polynomial approximation scheme to solve the k-SNPP on the plane for this "convex-hull" arrangment of terminals.

In Chapter 3 we study the problem of designing minimum-cost k-connected networks using only edges *between terminals.* We show that minimum-cost k-connected networks have a "no-crossing" property that can be used in local improvement algorithms. We also show that the support of the solution to the linear programming formulation of this problem also has this property.

In Chapter 4 we consider the problem of network synthesis using as few edges as possible. Our motivation is that edges have a certain set-up cost and so we would like to use as few edges as possible. We give a new algorithm that improves upon the algorithms of Gomory and Hu and Gusfield in the sense that it produces a design that uses fewer or no more edges. We also give a lower bound on the number of edges needed to provide the required network synthesis.

In Chapter 5 we study disjoint $[s,t]$-dicuts which are of importance in computing two-terminal network reliability. We give a new path formulation to the polytope whose extreme points are the incidence vectors of subgraphs with exactly k disjoint $[s,t]$-dicuts. This leads to a characterization of all subgraphs that contain exactly k disjoint $[s,t]$-dicuts. We also discuss connections with a new generalization of the max-flow problem.

Finally, in Chapter 6 we conclude the thesis by discussing some open problems.

Chapter 2

The k-connected Steiner Network Design Problem on the Plane

In this chapter we study the k-connected Steiner network design problem on the plane.

2.1 Introduction

We are given n points on the plane (called *terminals*). Recall the definition of the k-SNPP: using straight lines as edges, design a network that spans these points and that contains k edge-disjoint paths between any pair of terminals. Our objective is to minimize the total cost of the network, which is defined as the sum of the lengths of its edges. We refer to a solution to k-SNPP as a k-Steiner network (k-SN). If the connectivity requirements all equal one and if the network cannot use any new points (Steiner points), the resulting problem is the Euclidean spanning tree problem; if we permit Steiner points, the problem is the Euclidean Steiner tree problem.

One special configuration of the terminals arises often in practice: all the terminals lie on the boundary of their convex hull. By making small perturbations, if necessary, we assume that all the terminals are vertices of their convex hull - we do so for simplicity. Cockayne [7], Du et. al. [16] and Provan [58] have studied the Euclidean Steiner tree problem on such an arrangement of terminals. Provan gives a fully polynomial approximation scheme for finding a Steiner tree for this arrangement of terminals (finding the optimal solution seems quite hard even for this special case).

In this chapter we study the problem for this "convex-hull" configuration of terminals when we require k edge-disjoint paths between every pair of terminals. We make the assumption that we can use as many copies of an edge as we want. For example, the minimum-cost network on two terminals would be k copies of the straight line joining the two terminals. The multiple-edge assumption is not a major drawback when designing networks on the plane because by using parallel edges very close to each other, instead of multiple edges, we can design a new network without multiple edges and whose cost is within an arbitrary constant ϵ of the optimal network.

We prove that for k even, the optimal solution is a design with $k/2$ copies of the boundary of the convex hull, whereas for k odd, the optimal solution is a design with $(k-1)/2$ copies of the boundary and one copy of the Steiner tree. For example, Figures 2.1(a), (b) and (c) show the minimum-cost one, two, and three connected networks on the set of terminals represented by the square nodes. The black nodes are the Steiner points. Since if a network is 2-vertex connected, it is also 2-edge connected, the solution to the 2-edge connected problem which is 2-vertex connected also solves the problem of finding the minimum-cost 2-vertex connected network. We also show that the minimum-cost 3-vertex connected network on this arrangement of terminals is one copy of the boundary of the convex hull and one copy of the full Steiner tree on the terminal set.

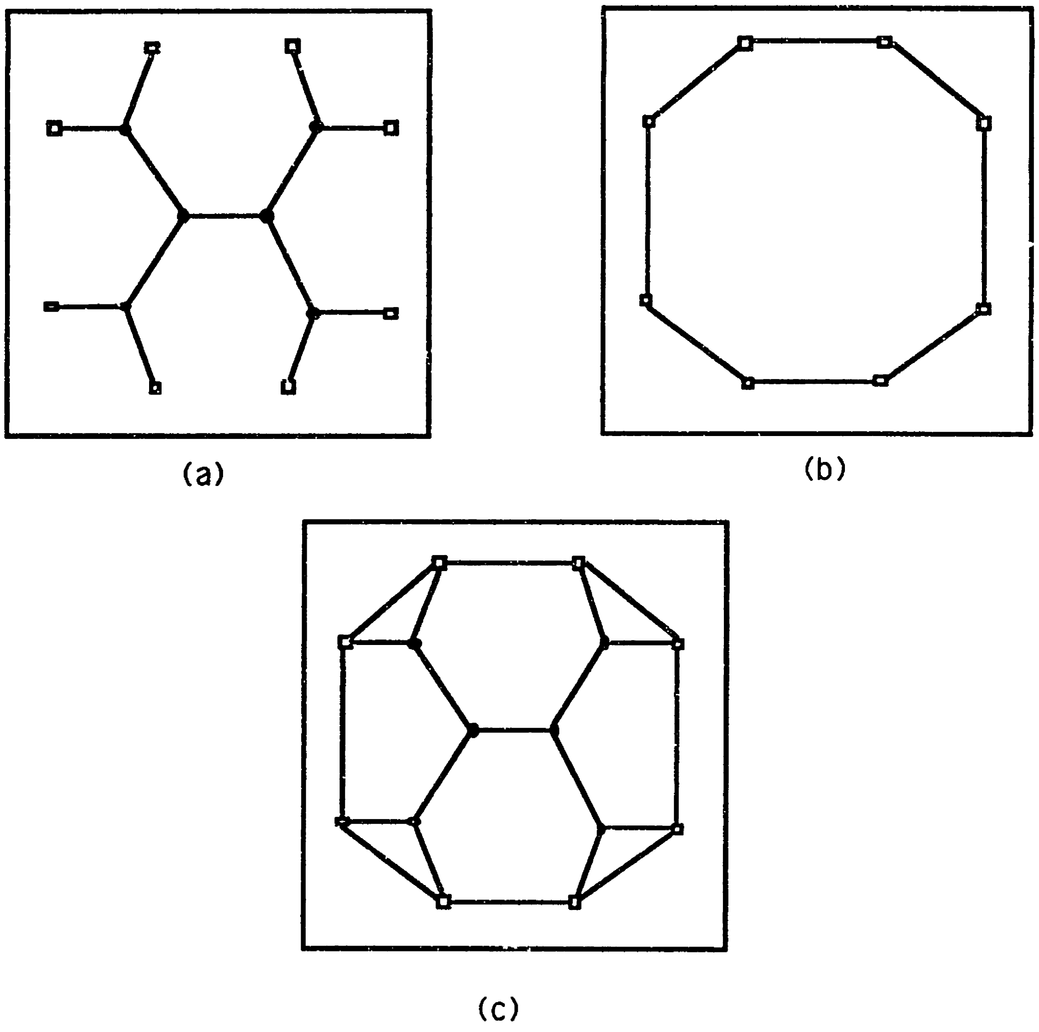


Figure 2.1: Minimum-cost one, two and three connected networks.

Thus the result says that under this model and arrangement of terminals, designing connected Steiner networks for situations with even connectivity requirements is easy and for odd connectivity requirements, it is no harder than solving a Steiner tree problem.

The rest of the chapter is organized as follows: In Section 2.2 we state the problem more precisely for situations both on the plane and on networks; in Section 2.3 we state some definitions and summarize the results that we need; in Section 2.4 we prove a theorem on planar graphs that will be the main tool for many of our proofs, in Section 2.5 we prove that the k-connected Steiner network is planar; in Section 2.6 we prove our main result, and state a fully polynomial approximation scheme for situations in which k is odd; in Section 2.7 we prove a generalization of a theorem of Erickson, Monma and Veinott; and, finally, we make some concluding remarks in Section 2.8.

2.2 Definitions and previous results

For one interesting class of networks - the τ-planar networks - the Steiner tree problem with terminal set Z, is polynomially solvable. A graph G, with a given subset of its nodes Z, is τ-planar (τ-planar stands for *terminal*-planar) if it has a planar representation with Z lying entirely on the boundary of G. It is possible to test whether a graph has a τ-planar embedding in polynomial time [3]. The following theorem due to Erickson, Monma and Veinott [19], when used in conjunction with the Dynamic Programming algorithm of Dreyfus and Wagner [15] and Levin [41], leads to a $O(n^2z^2 + n^3)$ (in this expression, $|Z| = z$) algorithm for finding a Steiner tree in τ-planar networks.

Theorem 2.1 *[19] Let P be any polygon in the plane, Z a subset of nodes of that polygon, and T a planar tree spanning Z contained entirely inside P. For each edge $e = (u, v) \in T$, let $T(e, v)$ be the connected subtree containing node v obtained if we remove e from T. Then $T(e, v)$ has the property that the nodes of Z contained in $T(e, v)$ form an interval in Z; that is, they appear consecutively on the boundary of G.*

In Section 2.7 we develop a generalization of this result that applies to minimum-cost k-connected Steiner networks; we prove that these networks have a property similar to that of trees.

2.3 A theorem on planar graphs

In this section we prove the following theorem on planar graphs:

Theorem 2.2 *Let G be a planar 2-vertex-connected graph with a fixed planar embedding. If G contains r_{uv} edge-disjoint paths between two nodes u and v, then it contains r_{uv} edge disjoint paths between u and v satisfying the property that at most two of them have an edge in common with the boundary of G.*

The next set of definitions and theorem are used in proving Lemma 2.1. Given a vertex $x \in V$ and two adjacent edges (x, u) and (x, v), we refer to the operation of removing these two edges and creating a new edge that connects u to v, as *splitting off* the two edges (x, u) and (x, v) from x. Note that splitting off two edges from a vertex of an Eulerian graph produces another Eulerian graph. Let $c_G(u, v)$ denote the maximum number of edge-disjoint paths between u and v in G. Lovasz [44] proved the following result:

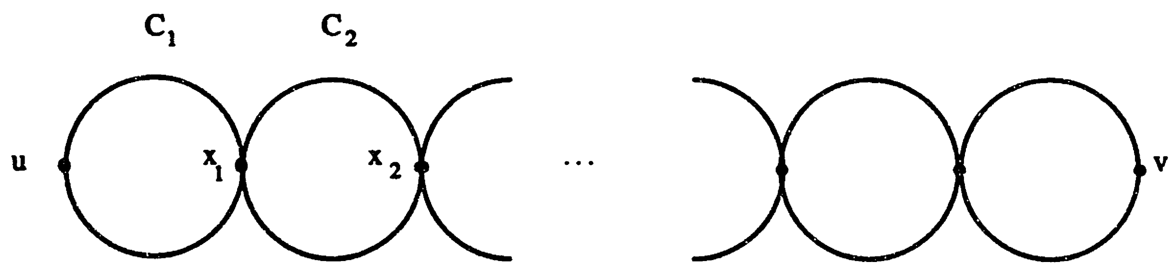


Figure 2.2: A uv-necklace

Theorem 2.3 *[44] Let G be an Eulerian graph, $x \in V$, and (x,u) an edge adjacent to x. Then the graph always contains another edge (x,v) adjacent to x with the property that splitting off the edges (x,u) and (x,v) from x, the resulting graph G' satisfies*

$$c_{G'}(a,b) = c_G(a,b)$$

for any two nodes $a, b \in V - \{x\}$.

For $u, v \in G$, a *uv-necklace* is a subgraph of G that is a sequence of cycles $C_1, C_2, \cdots, C_k$ satisfying the property that $u \in C_1$, $v \in C_k$, $C_i \cap C_j = \emptyset$ if $j \neq i+1$, $C_i \cap C_{i+1} = \{x_i\}$, and $x_i \neq x_{i+1}$ for $i = 1, 2, \cdots, k-1$ (See Figure 2.2).

Let $x_0 = u$ and $x_k = v$. For each cycle C_i, we refer to the two portions of C_i between x_i and x_{i+1} as the *upper* and *lower* parts of C_i.

Lemma 2.1 will give a useful tool for dealing with edge-disjoint paths. It essentially says that the smallest set of edges that form two edge-disjoint paths between a pair of nodes is a uv-necklace. Let Q_1 and Q_2 be two edge-disjoint paths in G between the nodes u and v. As shown in Figure 2.3, $Q_1 \cup Q_2$ need not be a necklace. But notice that in Figure 2.3, we can find a subgraph (not necessarily induced) of $Q_1 \cup Q_2$ that is a uv-necklace. The next result shows that this is always true; that is, the graph $Q_1 \cup Q_2$ always has a uv-necklace has a subgraph.

Lemma 2.1 *$Q_1 \cup Q_2$ contains a uv-necklace as a subgraph.*

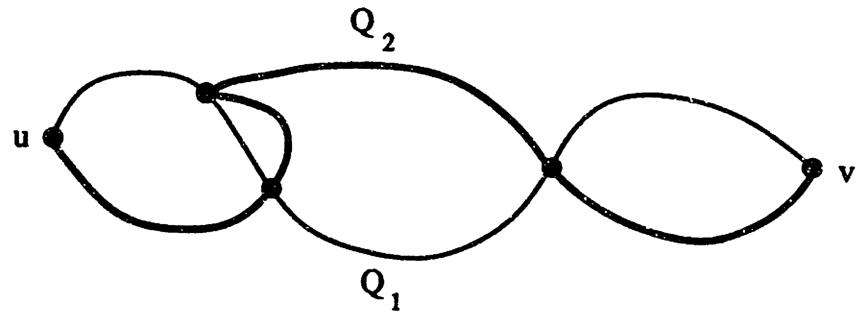


Figure 2.3: Two edge-disjoint paths need not be a necklace

Proof: Clearly $Q_1 \cup Q_2$ is a Eulerian graph in which all the nodes have degree two or four. We use induction on k, the number of degree four nodes. If $k = 0$, then $Q_1 \cup Q_2$ itself is the desired solution. Assume $k \geq 1$ and that the statement is true for all graphs with $k-1$ degree four vetrices. Let x be a degree four vertex. Clearly x is neither u nor v since our paths are simple. From Theorem 2.3, we can find two edges adjacent to x, say (x, a) and (x, b) that we can split off from x while preserving the connectedness between u and v. Let e be this new edge created between a and b. By the induction assumption, the resulting subgraph is a uv-necklace. Suppose edge e belong to cycle C_i of this necklace and node x to cycle C_j. If $i = j$, and e and x belong to different parts of C_i, then by putting back the two edges (x, a) and (x, b), we still have a necklace. If they belong to the same part, or we have $i \neq j$, we can replace the x to a portion of the necklace by (x, b) and still obtain a necklace. □

Let P be the boundary of G. Since G is 2-vertex-connected, P is a polygon. To prove Theorem 2.2, we have to show that given any set of r_{uv} edge-disjoint paths Q between u and v, we can obtain a new set of r_{uv} paths between u and v, Q', that satisfy the property that at most two of these new paths have an edge in common with P. We next describe such a procedure.

For two nodes a, b on P, let $[a, b]_P$ be the portion of P going clockwise along P from a to b. Assume that two paths Q_1 and Q_2 of Q lie inside P. Let $Q_i = q_0^i =$

$u, q_1^i, q_2^i, \cdots, q_{k_i-1}^i, v = q_{k_i}^i$ for $i = 1,2$. For $i = 1,2$, let $a_i = q_j^i$ where $(q_j^i, q_{j+1}^i) \in P$ is the first edge of Q_i to belong to P. Similarly, for $i = 1,2$, let $b_i = q_j^i$ where $(q_{j-1}^i, q_j^i) \in P$ is the last edge of Q_i to belong to P. We can assume, without loss of generality, that a_1 appears before b_1 on P in the clockwise direction. (Otherwise, we can switch the roles of u and v). If we assume that $\mathcal{Q}$ contains a minimal number of edges, by Lemma 2.1, $Q_1 \cup Q_2$ is a uv-necklace. We will show that we can obtain two new edge-disjoint uv-paths Q_1' and Q_2' in $Q_1 \cup Q_2$, satisfying the properties that $[a_1', b_1']_P$ has no edge in common with Q_2' - for $i = 1,2$, we define the nodes a_i' and b_i' on $P \cap Q_i'$ the same way as we have defined a_i and b_i for Q_i.

Give the polygon P a clockwise orientation. Call an edge $e \in (Q_1 \cup Q_2) \cap P$ *agreeaable* if going from u to v along Q_1 or Q_2, its direction agrees with the orientation given to P, and *disagreeable* otherwise.

Lemma 2.2 *In every cycle C_i of the uv-necklace $Q_1 \cup Q_2$, no part contains both agreeable and disagreeable edges.*

Proof: Let (p,q) and (r,s) be an agreeable and a disagreeable edge on the upper part of a cycle (belonging say, to Q_1) . Since the paths Q_1 and Q_2 are simple, node v lies in the region enclosed by $[q,s]_P - r - q$ (see Figure 2.4).

Thus in order for Q_2 to go from u to v, it must intersect Q_1 at some node between q and r, contradicting the assumption that all the nodes in this q-r segment have degree two in $Q_1 \cup Q_2$. □

Call a part of a cycle agreeable if it has agreeable edges and disagreeable otherwise.

Lemma 2.3 *No cycle of the uv-necklace $Q_1 \cup Q_2$ can have both parts of the same type.*

Proof: Let the upper part of C_i belong to Q_1 and the lower part to Q_2. Suppose both the upper and lower parts of a cycle C_i are agreeable. That is, there are edges on

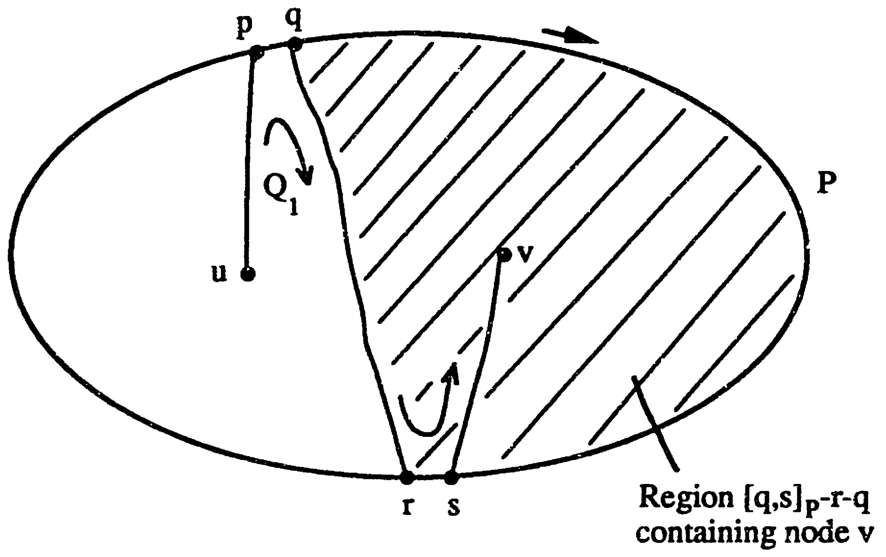


Figure 2.4: A part of a cycle does not have edges of both types.

$Q_1 \cap P \cap C_i$ and $Q_2 \cap P \cap C_i$ such that going from u to v along Q_1 and Q_2 respectively, they agree with the clockwise orientation of P. Let (p_1, q_1) and (p_2, q_2) be the first agreeable edges of the upper and lower parts respectively. Recall that x_i and x_{i+1} are the two degree four nodes of the cycle C_i. Now x_{i+1} lies in the closed region defined by $(x_i - [p_1, p_2]_P - x_i)$ as otherwise we would have a degree four node on the $x_i - p_2$ path of the lower part (see Figure 2.5).

But the path from q_2 to x_{i+1} will then intersect the upper part at some node as it has to enter the closed region defined by $(x_i - [p_1, p_2]_P - x_i)$ through some node and it cannot enter through a node on $[q_1, p_2]_P$ as Q_1 and Q_2 are contained in P, nor through the $x_i - p_2$ segment of Q_2 as all the nodes on this segment are assumed to be of degree two. Thus no cycle can have both parts of the same type. □

In this uv-necklace formed by $Q_1 \cup Q_2$, let Q'_1 be the path obtained by joining (i) the agreeable parts of the cycles and (ii) if neither part of a cycle is agreeable, the upper part. Let Q'_2 be the path obtained by joining the remaining parts of the cycles. In view of Lemmas 2.2 and 2.3, both the paths are well defined and edge-disjoint.

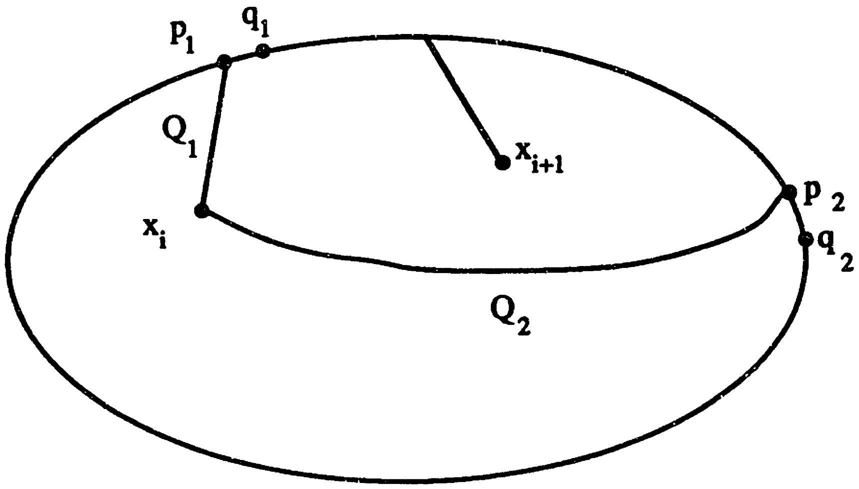


Figure 2.5: A cycle does not have both parts of the same type.

Let us denote this procedure of obtaining new paths Q_1' and Q_2' from Q_1 and Q_2 by the following notation:

$$[Q_1 * Q_2]_P^1 = Q_1'$$

$$[Q_1 * Q_2]_P^2 = Q_2'.$$

Lemma 2.4 *Q_2' has no edge in common with $[a_1', b_1']_P$.*

Proof: We use induction on the number of cycles r in the necklace $Q_1 \cup Q_2$. When $r = 1$, the two paths are node-disjoint (except at nodes u and v) and the proof is obvious from Figure 2.6 since Q_1' and Q_2' do not cross each other.

Assume the statement is true for all uv-necklaces with $r - 1$ cycles that join any two nodes and whose paths are defined the same way as Q_1' and Q_2'. The lower part of C_1 has no edge in common with $[a_1', b_1']_P$ since otherwise a node (other than x_1, see the definition of a necklace) on the lower part of C_1 will have degree four contradicting the fact that C_1 is the first cycle in the necklace. Now the upper and lower parts of the necklace from x_1 constitute a $x_1 v$-necklace. By the induction assumption, we obtain the conclusion of the lemma. □

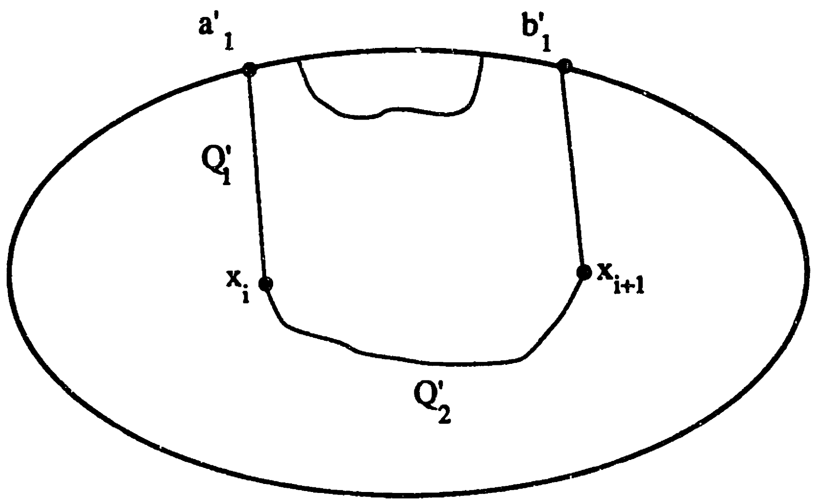


Figure 2.6: Two node-disjoint uv-paths.

Proof of Theorem 2.2 : If $r_{uv} \leq 2$, there is nothing to prove. Assume $r_{uv} = k > 2$. Let $\mathcal{Q} = \{Q_1, \cdots, Q_k\}$ be a set of k edge-disjoint paths from u to v. Assume that the first l of these paths $Q_1, \cdots, Q_l$ each have at least an edge in common with the P and the remaining paths do not have an edge in common with P. We assume that the subgraph of G induced by the paths in $\mathcal{Q}$ is minimal. That is, every set of k uv-edge-disjoint paths in this subgraph uses all the edges of the subgraph. Because of this assumption and Lemma 2.1, we can assume that $Q_i \cup Q_j$ is a uv-necklace for any pair i, j, $i \neq j$. If $l \leq 2$ we have nothing to prove. Assume $l > 2$. Let $H = \cup_{i=1}^{k} Q_i$. We will construct a path system in H that satisfies the conditions of the lemma. For $i = 1, \cdots, k$ let a_i and b_i be as defined before with respect to path Q_i and the polygon P. Note $a_i = u$, or $b_i = v$ (or both) are possible. What we do next is to untangle the paths.

Intuitively, as shown in Figure 2.7, if Q_1 and Q_2 are as shown in the picture, and if Q_3 is a third path that passes through P, we can find three edge-disjoint paths $u - a_1 - b_3 - v$, Q_2 and $u - g - v$ with the property that at most two of them have an edge in common with P. The difficulties arise because the paths need not be so

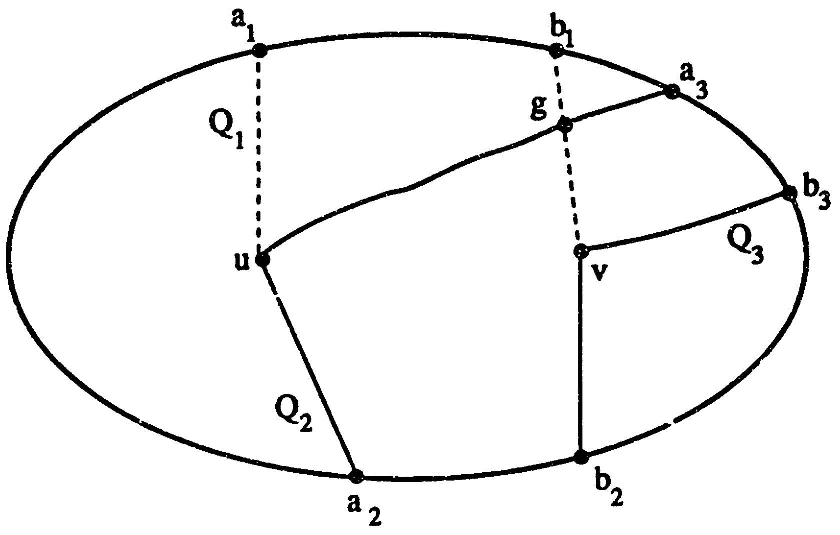


Figure 2.7: Obtaining three new paths from Q_1, Q_2 and Q_3

nicely vertex-disjoint and more than three paths might touch P.

Now, consider the output of the following algorithm:

```
*Initialize:*
for $i = 1, \cdots, k$, do
    $Q'_i = Q_i$
*Algorithm:*
for $j = 2, \cdots, l$, do
    $Q'_1 = [Q'_1 * Q'_j]^1_P$
    $Q'_j = [Q'_1 * Q'_j]^2_P$
```

Let $Q' = \{Q'_1, \cdots, Q'_k\}$.

By Lemmas 2.2, 2.3 and 2.4, it is clear that Q' is a set of k edge-disjoint paths with the following properties :

1. Q'_1 has only agreeable edges and no other path of Q' has any agreeable edges.

2. No path of Q' other than Q'_1 has an edge in the interval $[a'_1, b'_1]_P$.

3. By the minimality assumption on the subgraph created by Q the paths of Q' still satisfy the property that $Q'_i \cap Q'_j$ is a uv-necklace for any pair i, j, $i \neq j$.

The second property is true because at the end of each step of the algorithm, Q'_1 has no disagreeable edges and the clockwise interval on P spanned by its agreeable edges is non-decreasing.

Now, run the same algorithm on $\{Q'_2, \cdots, Q'_k\}$ with Q'_2 playing the role of Q'_1 and changing the orientation of P. The resulting Q''_2 will have only disagreeable (with respect to the previous clockwise orientation of P) edges and the paths of $\{Q''_3, \cdots, Q''_k\}$ will have neither agreeable nor disagreeable edges. Thus we have a set of paths Q'' that have at most two paths with an edge in common with P. □

Notice that when we apply the algorithm described in this proof to the three paths shown in Figure 2.7, we obtain the three paths $u - a_1 - b_1 - v$, Q_2, and $u - g - v$.

2.4 Planarity of the k-SN

In this section and the next we study k-SNPP, that is the k-SNP on the plane. The k-SN is a network on the plane that spans a given set of terminals and whose edges are straight lines between points. We prove that the k-SN has no crossing edges. The no-crossing property of k-SN is surprising because if we do not allow Steiner points, i.e., we ask for a minimum-cost network spanning the terminals and that uses only edges between terminals, the minimum-cost network can have crossing edges for odd k as the three connected network on four nodes in Figure 2.8 shows. (The main result of [27] is that such a network cannot have crossing edges for k even).

Theorem 2.4 *Let G be a k-SN. Then G does not contain a pair of crossing edges.*

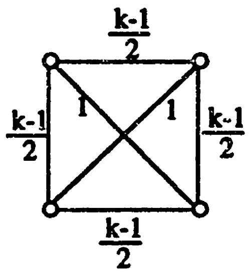


Figure 2.8: Counterexample for k odd, $k \geq 3$

Proof: Proof by contradiction. Suppose G has a pair of crossing edges (a', c') and (b', d') and that they cross at the point g. Choose four points a, b, c, d on $[a', g)$, $[b', g)$, $[c', g)$ and $[d', g)$, respectively, so that a, b, c, d forms a rectangle with opposite sides of the same length. Assume, without loss of generality, that the angle made by $a - g - d$ (and $c - g - b$) is less than or equal to ninety degrees. Let G^1 be the graph obtained by replacing the crossing edges by (a, d) and (b, d). and let G^2 be the graph obtained by replacing the crossing edges by a Steiner tree on a, b, c, d. Note that G^2 has two additional Steiner points e and f (see Figure 2.1). Moreover, note that the cost of both G^1 and G^2 is strictly less than that of G. Therefore, both G^1 and G^2 contain Steiner cuts of cardinality less than k. Let $[S^1, \bar{S^1}]$ be a Steiner cut in G^1 and $[S^2, \bar{S^2}]$ be a Steiner cut in G^2, both with cardinality less than k (Figure 2.9).

Without loss of generality we can assume that $a \in S^1$ and $a \in S^2$. Since G is a k-connected Steiner network, this implies that $d \in S^1$ and $b \in S^2$. Let

$$\begin{aligned} A &= S^1 \cap S^2 \\ B &= \bar{S^1} \cap S^2 \\ C &= \bar{S^1} \cap \bar{S^2} \\ D &= S^1 \cap \bar{S^2}. \end{aligned}$$

Clearly, $a \in A, b \in B, c \in C$ and $d \in D$. Also, since both S^1 and S^2 are Steiner cuts either B and D or A and C contain terminals. Assume without loss of generality that B and D contain terminals. The fact that $|[S^1, \bar{S^1}]| < k$ in G^1 and $|[S^2, \bar{S^2}]| < k$

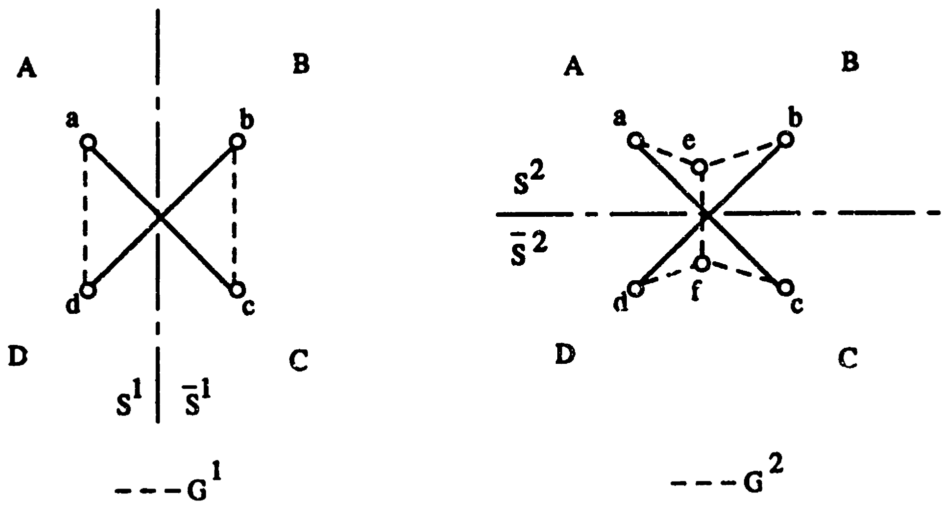


Figure 2.9: G^1 and G^2

in G^2 implies that

$$|[A,B]| + |[A,C]| + |[B,D]| + |[D,C]| \leq k+1 \tag{2.1}$$

$$|[A,D]| + |[A,C]| + |[B,D]| + |[B,C]| \leq k. \tag{2.2}$$

In these expressions, $[X,Y]$ $(X,Y \in \{A,B,C,D\})$ refers to the graph G and not to G^1 or G^2. The righthand side of the first expression is one greater than the right hand side of the second expression because in G^2 we are introducing the new edge (e,f). Adding (2.1) and (2.2), we find that

$$|[A,B]| + |[D,C]| + |[A,D]| + |[B,C]| + 2(|[A,C]| + |[B,D]|) \leq 2k+1. \tag{2.3}$$

Also the fact G is a k-SN implies the following two relations corresponding to the two Steiner cuts $[B,\bar{B}]$ and $[D,\bar{D}]$:

$$|[A,B]| + |[B,C]| + |[B,D]| \geq k$$

$$|[A,D]| + |[B,D]| + |[D,C]| \geq k.$$

Adding the inequalities, we get

$$|[A,B]| + |[D,C]| + |[A,D]| + |[B,C]| + 2|[B,D]| \geq 2k. \tag{2.4}$$

The inequalities (2.3) and (2.4) imply that

$$2|[A,C]| \leq 1, \tag{2.5}$$

a contradiction. □

The solution to the k-SNPP, will therefore be a planar graph.

Corollary 2.1 *The k-SN on n terminals that form the vertices of their convex hull on the plane is a τ-planar graph.*

2.5 k-SNPP on the vertices of a convex polygon on the plane

Recall that we are examining the problem of designing a minimum-cost k-connected network that spans a set of terminals that are points on the plane and whose edges are straight lines.

2.5.1 Structure

Define a *polygon tree* P as follows: A polygon tree is a connected graph P whose edges have a partition into cycles $P_1, \ldots, P_t$ that satisfy the property that $|P_i \cap P_j| \leq 1$ for $i \neq j$ (see Figure 2.10).

The following two propositions are obvious. In the first proposition, the boundary is not a cycle because when we traverse the boundary, we might revisit a node. The second proposition is true, because we have assumed that the costs are strictly positive and this prohibits the presence of redundant edges.

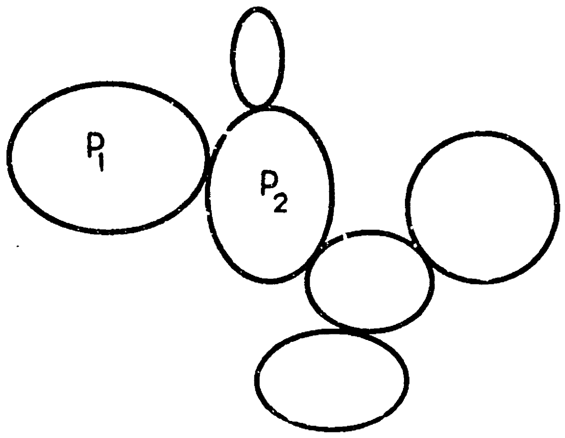


Figure 2.10: Polygon tree

Proposition 2.1 *For $k \geq 2$, the boundary of every k-connected planar graph is a polygon tree.*

Proposition 2.2 *For $k \geq 2$, the boundary of a k-SN in τ-planar graphs is a polygon tree.*

We now prove the main theorem of this chapter:

Theorem 2.5 *Let Z be n points on the plane that are the vertices of their convex hull. Then for k even,the k-SN on Z is $k/2$ copies of the boundary of the convex hull and for k odd, the k-SN on Z is $(k-1)/2$ copies of the boundary of the convex hull plus one copy of the Steiner tree.*

Proof: The proof is easy using Theorem 2.2 and Propositions 2.1 and 2.2. Clearly, among all polygon trees that span Z, the one with the minimum-cost is the convex hull. Now consider a k-SN for this configuration of the terminals. Its boundary is a polygon tree and by removing one copy of the boundary, by Theorem 2.2 we obtain a $(k-2)$-SN. Thus the boundary of the k-SN that we had removed must form the convex hull of Z, since otherwise we can obtain a k-SN of lesser cost. If $(k-2)$ is one, by definition we obtain a Steiner tree and if $k-2$ is two we obtain a polygon tree which again is the convex hull. □

Thus for $k=2$ we also obtain a minimum-cost 2-(vertex) connected network that spans Z. A Euclidean Steiner tree is called *full* if it contains $n-2$ Steiner points. Since a full Steiner tree does not use an edge between two terminals, if the Steiner tree on Z is full, we also obtain a minimum-cost 3-vertex connected network. The result in the next section also gives a fully polynomial approximation scheme for finding a minimum-cost 3-vertex connected network for this arrangement of terminals.

Because of this result, designing k-connected networks for this arrangement of terminals is trivial if k is even, and if k is odd the fully polynomial approximation scheme due to Provan for Steiner trees automatically gives a fully polynomial approximation scheme for k-SNPP.

2.5.2 Fully polynomial approximation scheme for k odd

In this section we state Provan's fully polynomial approximation scheme for finding the Euclidean Steiner tree when the terminals form the vertices of their convex hull.

Let P be a minimization problem and suppose that each instance I of P has a set of *feasible solutions*, each with a corresponding *value*. We assume that the values for I are positive and bounded. The *optimal value* $OPT(I)$ for an instance I is the infimum of the set of solution values. A *fully polynomial approximation scheme (f.p.a.s)* for P is an algorithm which, for each instance I of P and each $\epsilon > 0$, finds a solution for I with value v satisfying

$$\frac{v}{OPT(I)} \leq 1+\epsilon$$

and whose running time is bounded by a polynomial in $1/\epsilon$ and the length of the input for I [23].

Provan's algorithm solves the following problem.

Input: Terminal set Z of cardinality n, $\epsilon > 0$.

Output: A spanning graph of Z of length v satisfying

$$\frac{v}{OPT(Z)} \leq 1 + \epsilon$$

The Algorithm :

1. Find the convex hull R of Z with boundary polygon Γ, and also the values $x_{\min}, x_{\max}, y_{\min}$, and $y_{\max}$ of the minimum x-coordinate, maximum x-coordinate, minimum y-coordinate, and maximum y-coordinate, respectively of points in Z.
2. Let $N = \lceil(8n - 12)/\epsilon\rceil$ and let $x_{\min} = x_0 < x_1 < \cdots < x_N = x_{\max}$ and $y_{\min} = y_0 < y_1 < \cdots < y_N = y_{\max}$ be values spaced equally between their endpoints. Let $V_0 = \{(x_i, y_j) \ : i, j = 0, \ldots, N\}$ be the lattice of points with these values, and let $V_x(V_y$, respectively) be the set of points on nonvertical (nonhorizontal, respectively) edges of Γ whose x-coordinate (y-coordinate, respectively) is among these values.
3. Define the graph G_ϵ to be the complete graph with vertex set $V = Z \cup (V_0 \cap R) \cup V_x \cup V_y$ and edge costs $w(u, v)$ equal to the Euclidean distance between u and v, for all $u, v \in V$.
4. Find the Steiner tree T on the network G_ϵ for Z. This is the required spanning network.

The proof of the correctness and running time of this algorithm can be found in [58].

2.6 Structure of minimum-cost k-connected Steiner networks in τ-planar networks

In this section we prove two theorems regarding the structure of k-SN in τ-planar networks. The first one generalizes Theorem 2.1 due to Erickson, Monma and Veinott [19] while the second one proves an interesting property of k-SN for k even. Combined, these theorems suggest that it might be possible to solve the 2-connected SNP for τ-planar networks.

For a k-SN N and a minimal Steiner cut C in N, let N_C^1 and N_C^2 be the two components obtained after removing the edges in C. In a τ-planar graph G, define an *interval* $[u, v)$ between distinct terminals u and v in Z to be the set of terminals, including u but not v, visited by traversing the outer face from u to v in the clockwise direction.

Lemma 2.5 *For $k \geq 2$ a minimal cut of a k-SN of a τ-planar graph has exactly two edges of the outer face.*

Proof: It is clear that all elements of the minimal cut belong to the same two-connected component of the k-SN. This minimal contains at least two edges from the outer face as our graph is planar and two-connected. By removing two edges from the outer face, we partition the nodes on the cycle into two sets one belonging to N_C^1 and the other to N_C^2. If the cut had a third edge of the outer face in it, by putting it back we would still get a cut contradicting the minimality of the cut. □

In a Steiner tree, every edge is a minimal Steiner cut. Thus the following is a generalization of Theorem 2.1.

Theorem 2.6 *Let N be a k-SN of a τ-planar network G. Then for a minimal Steiner cut C, the nodes of Z in N_C^1 (and N_C^2) form an interval in Z.*

Proof: For $k = 1$ this is Theorem 2.1, whereas for $k = 2$, since the k-SN is a polygon tree, it is clear. Every minimal Steiner cut has all its edges in a single block of the k-SN. Since all the terminals in each block form an interval and since by Lemma 2.5 the minimal Steiner cut has exactly two edges from the outer face, removing the edges in the cut leaves two components whose terminals form intervals. □

Theorem 2.7 *For k even, every minimal Steiner cut in a k-SN of a τ-planar graph has exactly k edges.*

Proof: This is clearly true for the case $k = 2$, from Propositions 2.1 and 2.2. Now by Theorem 2.2, by removing a copy of the boundary from the k-SN, we obtain a $(k-2)$-SN. Thus, by induction, the minimal Steiner cut contains exactly $k-2$ edges from this $(k-2)$-SN. By Proposition 2.2, the k-SN contains exactly k edges. □

Chapter 3

Minimum-cost k-connected Networks on the Plane without Steiner points

In this chapter we consider the problem of designing a minimum-cost k-connected network spanning a given set of terminals using only edges that are straight lines *between pairs of the terminals.* That is, we prohibit the location of new points. We derive some structural properties for this problem that have applications in local improvemnt algorithms. The work reported in this chapter has been published in Goemans and Talluri [27].

3.1 Introduction

Local search algorithms for network design problems repeatedly search for a local transformation that preserves feasibility and improves the cost of the current network. The simplest and most commonly used transformation is the operation of replacing

k edges of a feasible network by k other edges. This operation is referred to as a k-change transformation and the resulting networks are called k-opt. Croes [11] and Lin [42] first applied 2-opt and 3-opt were to the traveling salesman problem. Their success motivated the use of 2-opt for the minimum-cost survivable network design problem, namely the problem of designing a minimum-cost network with connectivity requirements between any pair of vertices. When all connectivity requirements are equal, to say k, the problem reduces to the k-connected network design problem. Recently, Monma and Shallcross [48] have implemented several heuristics, including 2-opt and 3-opt, for the minimum-cost 2-connected network design problem. In the case of k-connected networks, it is not clear whether it is always possible to replace any 2 non-adjacent edges from the network design by 2 other edges and yet maintain feasibility of the resulting network. In this chapter, we show we can always apply such a 2-change transformation if k is even or $k = 1$. As a consequence, any Euclidean minimum-cost k-connected network has no crossing edges when k is even. Monma et al. [47], for k=2, and Bienstock et al. [2], for general k, have developed other structural properties of minimum-cost k-connected networks. In the last section, we show that similar results hold for a linear programming relaxation of the k-connected network design problem.

3.2 No-Crossing property for minimum k-connected networks

The following Theorem is the main result of this chapter:

Theorem 3.1 *Let $G = (V, E)$ be a k-connected graph and let (a, c) and (b, d) be two non-adjacent edges of G. Let G^1 be obtained from G by replacing (a, c) and (b, d) by (a, d) and (b, c) and let G^2 be obtained by replacing (a, c) and (b, d) by (a, b) and*

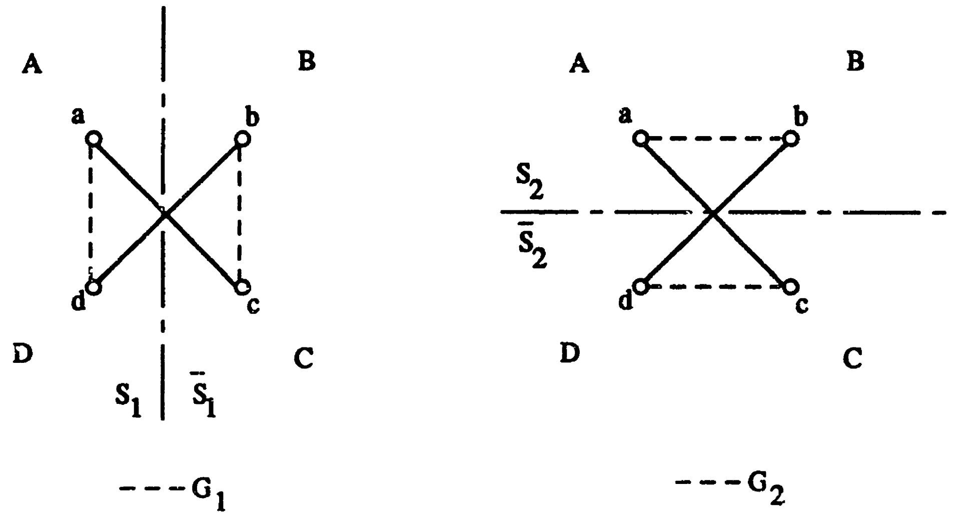


Figure 3.1: G^1 and G^2

$, d)$.

Then if k is even, either G^1 or G^2 is k-connected.

The proof is quite similar to Theorem 2.4. Before proving the result for any even number k, we give a short proof of its validity for $k = 2$. In that case, every two edges of G lie on a common circuit, where a circuit means a closed path without the restriction that the nodes are distinct. But, in a circuit, we can replace any two nonadjacent edges by two other edges so that the resulting graph is still a circuit, and thus 2-connected. The result follows by noticing that replacing an induced subgraph of a 2-connected graph by another 2-connected subgraph always creates a 2-connected graph. Monma and Shallcross [48] have used this observation to derive two-optimal and three-optimal heuristics for the problem of designing a minimum-cost 2-connected network.

Proof of Theorem 3.1:

Proof by contradiction. Suppose both G^1 and G^2 contain cuts of cardinality less than k. Let $[S^1, \bar{S^1}]$ be a cut in G^1 and $[S^2, \bar{S^2}]$ be a cut in G^2, both with cardinality less than k (Figure 3.1). Without loss of generality we can assume that $a \in S^1$ and $a \in S^2$. This implies that $d \in S^1$ and $b \in S^2$ since otherwise we could contradict the k-connectedness of G. Let

$$\begin{aligned} A &= S^1 \cap S^2 \\ B &= \bar{S}^1 \cap S^2 \\ C &= \bar{S}^1 \cap \bar{S}^2 \\ D &= S^1 \cap \bar{S}^2. \end{aligned}$$

Clearly, $a \in A, b \in B, c \in C$ and $d \in D$. The fact that $|[S^1, \bar{S}^1]| < k$ in G^1 and $|[S^2, \bar{S}^2]| < k$ in G^2 implies that

$$|[A,D]| + |[A,C]| + |[B,D]| + |[B,C]| \leq k+1 \tag{3.1}$$

$$|[A,B]| + |[A,C]| + |[B,D]| + |[D,C]| \leq k+1. \tag{3.2}$$

In these expressions, the arc sets $[X,Y]$ (for $X,Y \in \{A,B,C,D\}$) refer to the graph G and not to G^1 or G^2. Adding (3.1) and (3.2), we obtain

$$|[A,B]| + |[A,C]| + |[A,D]| + |[B,C]| + 2(|[A,C]| + |[B,D]|) \leq 2k+2. \tag{3.3}$$

The k-connectedness of G implies the following four relations corresponding to the four cuts $[A,\bar{A}], [B,\bar{B}], [C,\bar{C}]$ and $[D,\bar{D}]$:

$$|[A,B]| + |[A,C]| + |[A,D]| \geq k$$

$$|[A,B]| + |[B,C]| + |[B,D]| \geq k$$

$$|[A,C]| + |[B,C]| + |[D,C]| \geq k$$

$$|[A,D]| + |[B,D]| + |[D,C]| \geq k.$$

Adding all four inequalities and dividing by 2, we obtain

$$|[A,B]| + |[A,C]| + |[A,D]| + |[B,C]| + |[A,C]| + |[B,D]| \geq 2k. \tag{3.4}$$

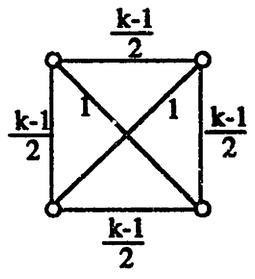


Figure 3.2: Counterexample for k odd, $k \geq 3$

The inequalities (3.3) and (3.4) imply that

$$|[A,C]| + |[B,D]| \leq 2, \tag{3.5}$$

and, therefore, $|[A,C]| = 1$ and $|[B,D]| = 1$. Thus, from (3.1) and (3.2), we find that

$$|[A,D]| + |[B,C]| \leq k-1,$$

$$|[A,B]| + |[D,C]| \leq k-1.$$

Now, because $k-1$ is odd, either $|[A,D]|$ or $|[B,C]|$ is less than or equal to $\lfloor (k-1)/2 \rfloor$ and either $|[A,B]|$ or $|[D,C]|$ is less than or equal to $\lfloor (k-1)/2 \rfloor$. Any combination of these outcomes implies that one of the cuts $[A,\bar{A}], [B,\bar{B}], [C,\bar{C}]$ or $[D,\bar{D}]$ in G has cardinality less than or equal to $k-1$, contradicting the k-connectedness of G. □

The graph depicted in Figure 3.2 shows that a version of Theorem 3.1 with k odd, $k \geq 3$ is not valid. The integers on the edges represent the multiplicity of the edges.

The proof of this theorem establishes the following is result:

Lemma 3.1 *Let G^1 and G^2 be defined as in Theorem 3.1. Then if k is odd and neither G^1 nor G^2 is k-connected, then both (a,b) and (c,d) have multiplicity one.*

Theorem 3.1 shows that, if k is even, we can always perform a 2-change transformation. The main motivation for deriving the previous two results comes from the Euclidean minimum-cost k-connected network design problem; in this context, V

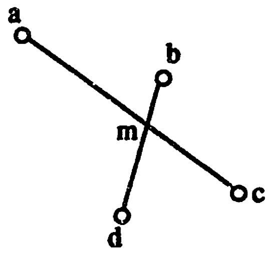


Figure 3.3: Two edges crossing in m

corresponds to a set of points in the Euclidean plane and c_e represents the Euclidean distance between the endpoints of e. An edge $e = (i, j) \in A$ of an optimal network can be represented as a straight line between the vertices i and j.

Theorem 3.2 *If k is even, any Euclidean minimum-cost k-connected network has no crossing edges; if k is odd, every pair of crossing edges in any Euclidean minimum-cost k-connected network can have multiplicity at most one.*

Proof:

The proof is by contradiction. Let $N = (V, A)$ be a Euclidean minimum-cost k-connected network and assume that (a, c) and (b, d) are two edges that cross at the point m (Figure 3.3). Theorem 3.1 and the fact that $c_{ad}+c_{bc} < c_{am}+c_{md}+c_{bm}+c_{mc} = c_{ac} + c_{bd}$ and $c_{ab} + c_{cd} < c_{am} + c_{mb} + c_{cm} + c_{md} = c_{ac} + c_{bd}$ contradict the optimality of N. □

3.3 Linear programming relaxation

In this section, we derive similar results for a linear programming relaxation of the Euclidean minimum-cost k-connected network design problem - namely, we show that the support of the optimal solution to an LP relaxation of the problem satisfies crossing properties similar to the ones shown in the previous section for optimal

Euclidean k-connected networks. More formally, we consider the following linear program:

$$\begin{array}{llll} & Z_k & = & Min \sum_{e \in E_V} c_e x_e \\ & & & \text{subject to} \\ (P) & & & \sum_{e \in [S,\bar{S}]} x_e \geq k \qquad \forall S \subset V, S \neq \emptyset \\ & & & 0 \leq x_e \qquad \forall e \in E_V, \end{array}$$

where V is a finite set of points in the Euclidean plane and c_e represents the Euclidean distance between the endpoints of e. By linearity, $Z_k = \frac{k}{2} Z_2$. Cunningham has shown that Z_2 is equal to the Held-Karp lower bound [35] for the traveling salesman problem when the cost function c satisfies the triangle inequality (see Monma et al. [47] or Goemans and Bertsimas [26]). Let x^* be an optimal solution to (P) and let $E = \{e : x_e^* > 0\}$. The graph $G = (V, E)$ possesses several nice properties, e.g. G is 1-tough i.e., if we remove any set of s vertices the resulting graph has at most s connected components. G also has the following property.

Theorem 3.3 *$G = (V, E)$ has no crossing edges.*

Proof:

Let (a, c) and (b, d) be two crossing edges of G. The optimality of x^* implies that G contains cuts $[S^1, \bar{S}^1]$ and $[S^2, \bar{S}^2]$ satisfying the following properties:

1. $a, d \in S^1$, $b, c \notin S^1$, $a, b \in S^2$ and $c, d \notin S^2$ (see Figure 3.1),
2. $x(S^1, \bar{S}^1) = k = x(S^2, \bar{S}^2)$ where $x(S, S') = \sum_{e \in [S,S']} x_e^*$.

Otherwise, we could decrease x_{ac}^* and x_{bd}^* by some positive value ϵ and increase either x_{ab}^* and x_{cd}^* or x_{ad}^* and x_{bc}^* by ϵ and obtain a feasible solution to (P) of lower total cost.

By defining A, B, C and D as in Theorem 3.1, decomposing $x(S^1, \bar{S}^1)$ and $x(S^2, \bar{S}^2)$ in terms of $x(X, Y)$ $(X, Y \in \{A, B, C, D\})$ and using the fact that $x(X, \bar{X}) \geq k$ for $X = A, B, C, D$, we find that $x(A, C) = x(B, D) = 0$. This result contradicts the fact that $x^*_{ac} > 0$ and $x^*_{bd} > 0$. □

We next show why a plausible conjecture relating the support of the LP solution and certain objects from Computational Geometry is false. For each vertex $i \in V$, the Voronoi region R_i associated with i is the set of points in the plane closer to i than to any other vertex. The Voronoi diagram of V consists of all Voronoi regions R_i. The dual of the Voronoi diagram is called the Delaunay triangulation and is the graph $DT(V)$ whose vertex set is V and whose edge set consists of all edges (i, j) for which R_i and R_j share a boundary of more than one point. The minimum spanning tree is always a subgraph of the Delaunay triangulation while the optimal traveling salesman tour need not be [14], [38]. The no-crossing property expressed in Theorem 3.3 and the 1-toughness are also properties of the Delaunay triangulation $DT(V)$ (for the 1-toughness results see [12]). Hence, one might suspect that G, the support of the optimal solution to (P), is a subgraph of the Delaunay triangulation. However, this is not true as can be seen from Dillencourt's example, reproduced in Figure 3.4, of a non-Hamiltonian Delaunay triangulation [13]. Otherwise, according to Cunningham's result [47], optimizing the objective function $\sum c_e x_e$ over the following Held and Karp polytope

$$\begin{aligned}
&\sum_{e \in [S, \bar{S}]} x_e \geq k && \forall\, S \subset V, S \neq \emptyset \\
&\sum_{e \in [\{i\}, V \setminus \{i\}]} x_e = k && \forall\, i \in V \\
&0 \leq x_e && \forall e \in E_D,
\end{aligned}$$

where E_D is the edge set of $DT(V)$, would give the same value as optimizing over (P). But it can easily be verified that the Held and Karp polytope for $DT(V)$ is empty.

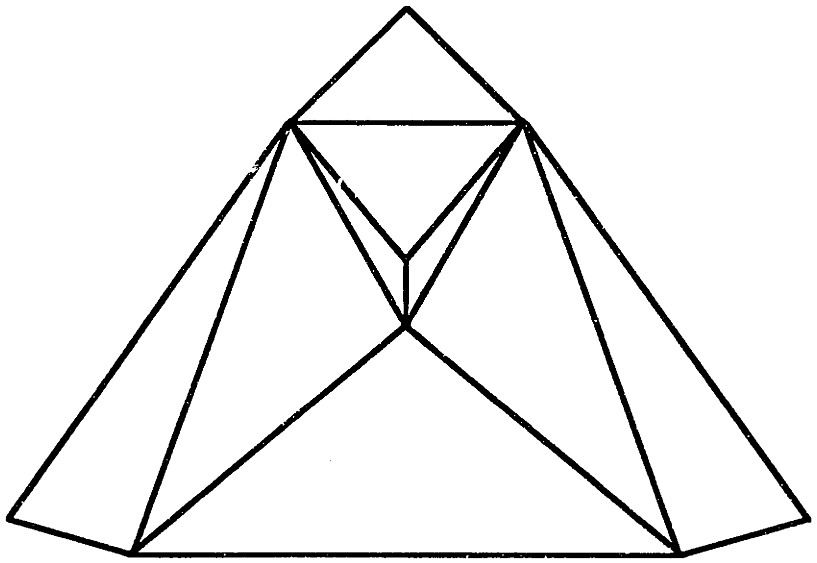


Figure 3.4: Dillencourt's example of a non-Hamiltonian Delaunay triangulation

Chapter 4

Network Synthesis with Few Edges

In Chapter 1 we defined the network synthesis problem as the LP relaxation of SNDP when the underlying graph G is complete. Although the LP relaxation has an exponential number of constraints, it can be solved efficiently by the Ellipsoid method because the separation problem can be solved in polynomial time - see Bland et. al. [4]. However when the costs are all uniform, Gomory and Hu [29] give an elegant algorithm that solves the problem in a simple manner. Gusfield [30] gives two alternative algorithms that improve upon the Gomory-Hu procedure in the sense that they produce networks with fewer (or equal number of) edges in the suport of the solution. In this chapter, we consider solving the problem for uniform costs with a network that has the fewest possible number of edges in the support of the solution. So, for the rest of this chapter we assume that G is a complete graph and that $c_{ij} = 1$ for all $(i, j) \in E$.

We first give an algorithm that is guarenteed to do no worse than the algorithms of Gusfield and in some cases does better. We prove its validity and show an example where it performs better than Gusfiled's algorithms. We next give a lower bound on the number of edges needed in any network that satisfies all the flow requirements.

4.1 Introduction

The problem considered is the following: Given a $n \times n$ matrix of non-negative numbers $r_{ij}, (r_{ij} = r_{ji})$, find a network on n nodes, and capacities on the edges of the network that satisfy the property that the total sum of the capacities on all the edges of the network is minimum and the maximum flow f_{ij} between i and j in this network satisfies $f_{ij} \geq r_{ij}$ (such a network is called a *feasible* network). This problem was essentially completely solved by Gomory and Hu [29]. They gave a lower bound on the total capacity of any feasible network and then gave a procedure for designing a network which achieves this lower bound. Note that our requirements r_{ij} are not necessarily of the form $\min(r_i, r_j)$, that is they are more general. However, Gomory and Hu [29] show that the optimal total capacity remains the same if we assume that the requirements r_{ij} are of the form $\min(r_i, r_j)$.

The rest of this chapter is organized as follows: In Section 4.2 we define our notation and terminology and summarize the previous work. In Section 4.3 we give a new algorithm to solve this problem which is guaranteed to use no more edges than Gomory-Hu's or either of Gusfield's algorithms. In Section 4.4 we give a lower bound on the number of edges that is required in any minimum capacity feasible network. This gives improved performance guarantees for both our and Gusfield's algorithms. We give an example which achieves this bound.

4.2 Definitions, notation and previous work

Given any undirected graph G with capacities k_{ij} on its edges, let f_{ij} represent the maximum possible flow between node i and node j. For a given $n \times n$ matrix R of non-negative numbers $r_{ij}(r_{ij} = r_{ji})$, we refer to the graph G as *feasible* if the flows in the graph satisfy the inequalities

$$f_{ij} \geq r_{ij} \quad \forall i, j,\ i \neq j.$$

The network synthesis problem is the one of finding a feasible network having the smallest total capacity. For each node i, let $r_i = \max_{j \neq i}(r_{ij})$. Call r_i the *node weight* of node i (the survivability requirement for SNDP). Assume that the r_i take on t distinct values, $w(1) > w(2) > \cdots > w(t)$. Then as shown in [29], the cost of the optimal network is unchanged if we assume that $r_{ij} = \min(r_i, r_j)$. So assume from now on that we have modified our requirements matrix so that it satisfies this condition. The capacity incident to every node i in any feasible network will be at least $\sum_j k_{ij} \geq r_i$. Hence, the total capacity installed is at least $(\sum_i r_i)/2$. Gomory and Hu [29] give a fast procedure that constructs a feasible network with total capacity equal to this quantity. Thus this problem is essentially solved. However, a naive implementation of this algorithm uses many edges. Gusfield [30] gives two new algorithms that are guaranteed to solve the feasibility problem using as few edges as the Gomory and Hu's algorithm. We next present the three algorithms and discuss some of their properties. We also illustrate their performance on an example.

The first algorithm is Gomory and Hu's algorithm which we will call algorithm A.

Algorithm A :

1. Compute a maximum weight spanning tree T using the weights r_{ij}.

2. Decompose T into a sum of subtrees, each having edges of equal weight. To do

this, define the decomposition of a tree T_i recursively. If k_i is the smallest edge in T_i, then T_i is decomposed into one copy of T_i with weight k_i on each edge, plus the decomposition of each subtree of T_i resulting from deleting all edges of weight k_i from T_i, and subtracting k_i from the weights of all the remaining edges.

3. For every tree T_i in the decomposition, create a cycle C_i containing all the nodes of T_i. Set the capacity of every edge in C_i equal to $k_i/2$ where k_i is the weight of each edge in T_i. Superimpose all of the cycles, merging common edges and summing the capacities. The resulting network G is a feasible network of minimum total capacity.

The next two algorithms are from Gusfield [30].

Algorithm A' :

1. Assume $r_i \geq r_{i+1}$ for $i = 1, \cdots, n$.
2. For $i = 2, \cdots, n$, repeat the following :

   Create edge $(i, i-1)$ with capacity $r_i/2$ and if $(r_i - r_{i+1})/2$ is nonzero, create edge $(i, 1)$.

Algorithm A^* :

1. Assume $r_i \geq r_{i+1}$ for $i = 1, \cdots, n$. We first construct a network $G(t)$ containing $t+1$ nodes, one for each of the distinct node weights $w(2)$ through $w(t)$, and two for the node weight $w(1)$. The former set of nodes is $\{2, \cdots, t\}$ and the latter set is $\{0, 1\}$. We construct $G(t)$ in steps 3 through 5.
2. Create an edge from node 0 to node 1 of capacity $(w(1) - w(2))/2$, and create an edge from node 0 to node 2 of capacity $w(2)/2$.

3. For $i = 2, \cdots, t-2$, do the following:

   Create an edge from node i to node $i+1$ of capacity $(w(i+1) - w(i+2))/2$ and an edge from node i to node $i+2$ of capacity $w(i+2)/2$.

4. Create an edge from node $t-1$ to node t of capacity $w(t)/2$.

5. If R contains more than one node of weight $w(i)$, for $i > 1$, then insert the new nodes of weight $w(i)$ into the $(i, i-2)$ edge of G, creating a path of edges between node i and $i-2$, each with capacity $w(i)/2$. If R contains more than two nodes of weight $w(1)$, then first split the $(0,1)$ edge with two parallel edges, one with capacity $w(1)/2$, and the other with capacity $(w(1) - w(2))/2$. Then insert the new nodes of weight $w(1)$ into the edge with capacity $w(1)/2$, creating a path of edges between nodes 0 and 1, each with capacity $w(1)/2$.

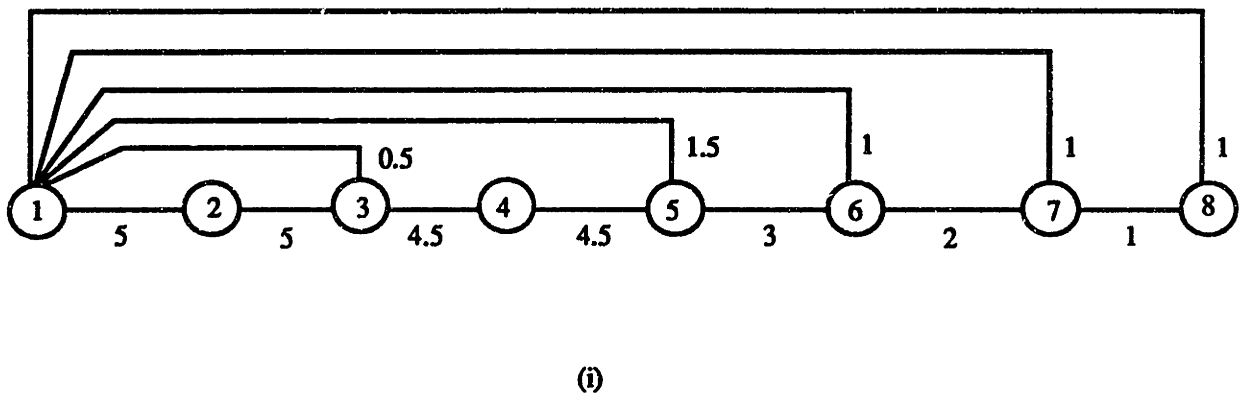
6. The result of steps 1) through 6) is a minimum capacity feasible network.

Let G_A, $G_{A'}$ and G_{A^*} be the networks generated by the three algorithms. We illustrate the performance of algorithms A' and A^* on an example.

Example: (see Figure 4.1) Consider designing a network on eight nodes with $r_1 = 10$, $r_2 = 10$, $r_3 = 10$, $r_4 = 9$, $r_5 = 9$, $r_6 = 6$, $r_7 = 4$, $r_8 = 2$.

We state some properties of the networks G_A, $G_{A'}$ and G_{A^*} generated by the three algorithms. The following result can be found in [30]. Let N_1 be the number of nodes of G with node weights r_1.

Theorem 4.1 *For a given requirements graph R, $G_{A'}$ and G_{A^*} contain $n-1+t$ edges if $N_1 > 2$ and $n-2+t$ otherwise. G_A contains at least as many edges as $G_{A'}$ or G_{A^*}.*



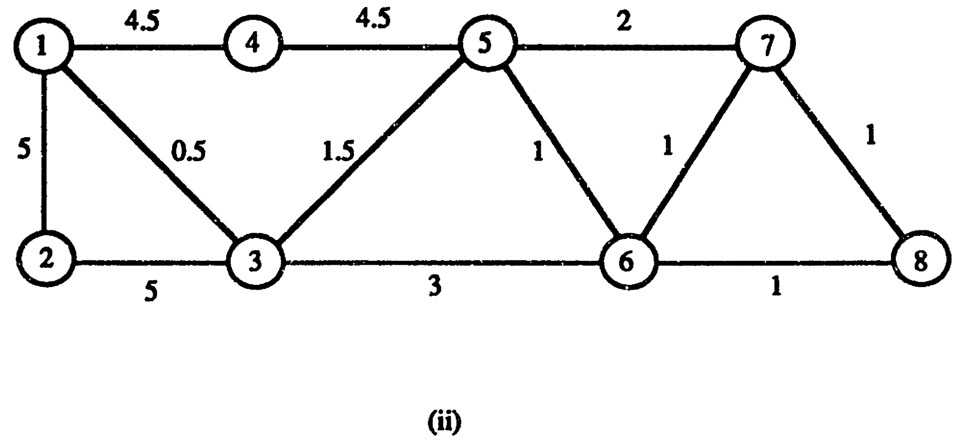


Figure 4.1: (i) $G_{A'}$ (ii) G_{A^*}

4.3 A new algorithm for network synthesis

In this section we present a new algorithm that produces a feasible network of minimum capacity and is guaranteed to have fewer or equal number of edges than either of Gusfield's algorithms. We first describe the algorithm and prove its validity, and then show that it uses no more than the number of edges used by algorithms A' or A^*. Finally, we give an example in which this algorithm actually does better than either of these algorithms.

Algorithm B^* :

1. Assume $r_1 = r_2 \geq \cdots \geq r_n$. Let k_{uv} be the capacity installed on edge (u,v) at any stage of the algorithm.

2. *Initialize :* Let G_2 be the graph on nodes 1 and 2 with two edges, each with capacities $r_1/2$.

3. For $k = 3, \cdots, n$, do

   Let edge $(i,j) \in G_{k-1}$ satisfy the condition

   $$(i,j) = argmin\{k_{uv} - r_k/2 \mid (u,v) \in G_{k-1}, k_{uv} - r_k/2 \geq 0\}.$$

   Obtain G_k from G_{k-1} by adding edges (k,i) and (k,j) with capacities $r_k/2$ and changing the capacity of (i,j) to $k_{uv} - r_k/2$.

4. $G_{B^*} = G_n$ is the output of the algorithm.

In Figure 4.2 and Figure 4.3 we show how the algorithm works on the previous example.

Theorem 4.2 *G_{B^*} is a feasible network of minimum capacity.*

Proof: We will show that G_k is a feasible network of minimum capacity for nodes $1, 2, \cdots, k$ for $k = 2, \cdots, n$. For $k = 2$, this is obvious. Assume G_l is a feasible network of minimum capacity for $1, 2, \cdots, l$. Note that G_l has an edge of capacity $r_l/2$. Thus, it is always possible to add node $l+1$ to G_l as stated in the algorithm. Suppose the algorithm node connects $l+1$ to nodes i and j. Then capacity of edge (i, j) in G_l is greater than or equal to $r_{l+1}/2$. Now, we can route any flow beyond the capacity of (i, j) which has to pass through edge (i, j) in G_l through node $l+1$. Thus the new network can satisfy all the flow requirements between nodes numbered l or less. Since $r(l+1) \leq r(k)$ for $k \leq l$, the flow requirement between node $l+1$ and any other node is $r(l+1)$. Between i and j, we can send a flow of $\min(r(i), r(j))/2$ without using edge (i, j). Thus, between $l+1$ and i or j, we can send a flow of $r(l+1)/2$. Between any other node m and $l+1$, since we can send a flow of $\min(r(m), r(i))$ between i and m, and we can send a flow of $\min(r(i), r(j))/2$ without using edge (i, j), we can always manage to satisfy the requirement between m and $l+1$. Since the total installed capacity at any stage is equal to $(\sum_{i=1}^{k} r_i)/2$, G_k is of minimum capacity. □

Next, we show that algorithm B^* uses at least as few edges as algorithm A'.

Theorem 4.3 *$G_{A'}$ contains at least as many edges as G_{B^*}*

Proof: Recall that the flow requirements take on t distinct values $w(1), \ldots, w(t)$. For k from 1 through t, let N_k be the number of nodes of weight $w(k)$. As is shown in [30], $G_{A'}$ has exactly $n-1+t$ edges if $N_1 > 2$ and $n-2+t$ if $N_1 = 2$. In step $l+1$ of algorithm B^*, if $r(l+1) = r(l)$, the number of edges in G_{l+1} is one more than those in G_l, and if $r(l+1) < r(l)$, the number of edges increases by at most two. Thus, we use no more edges than algorithm A'. □

As shown by Figures 4.2 and 4.3 Algorithm A^* gives a network with twelve edges for our example (as opposed to eleven from our algorithm). Thus, algorithm B^* will sometimes use fewer edges than algorithms A' or A^*.

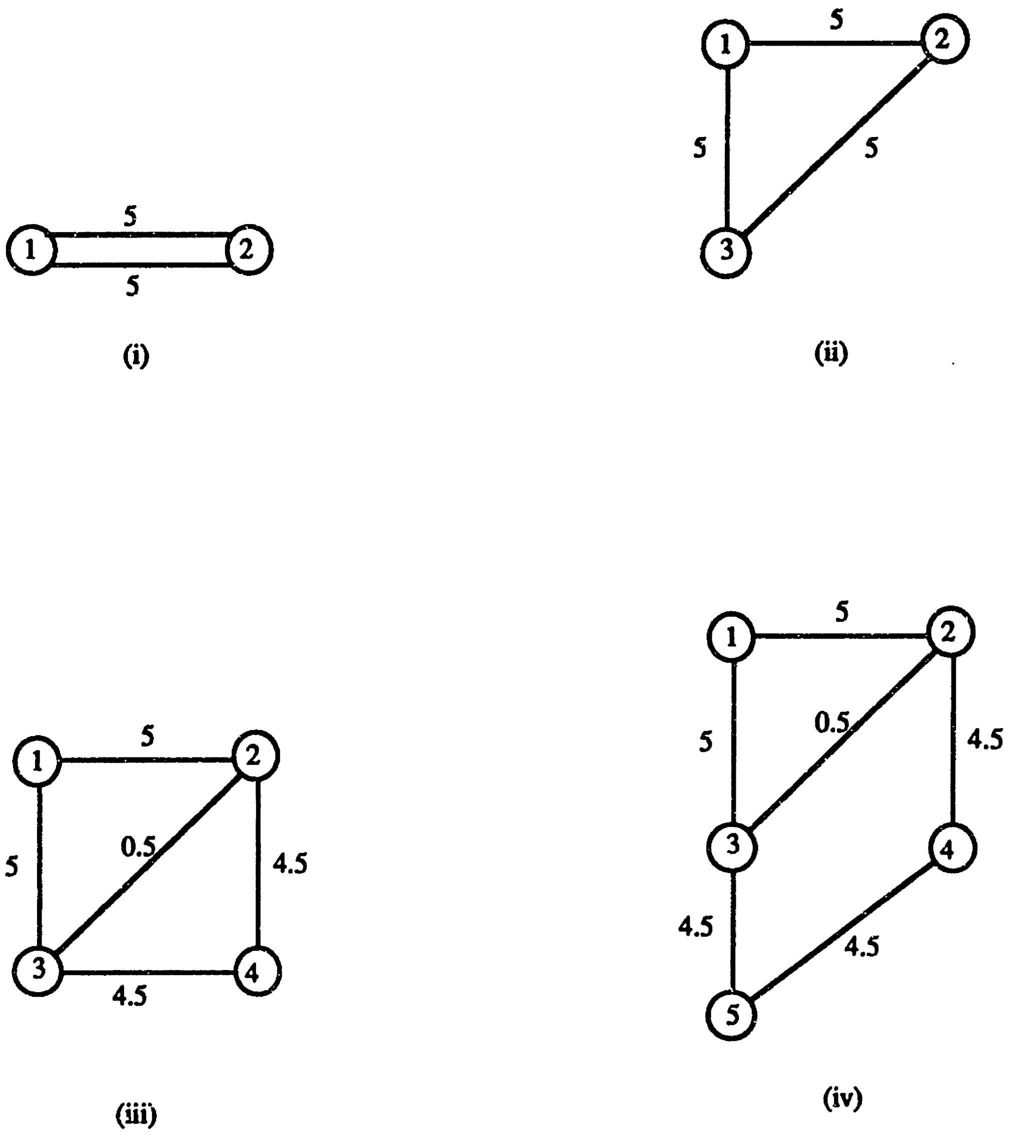


Figure 4.2: Algorithm B^* on the example in Gusfield

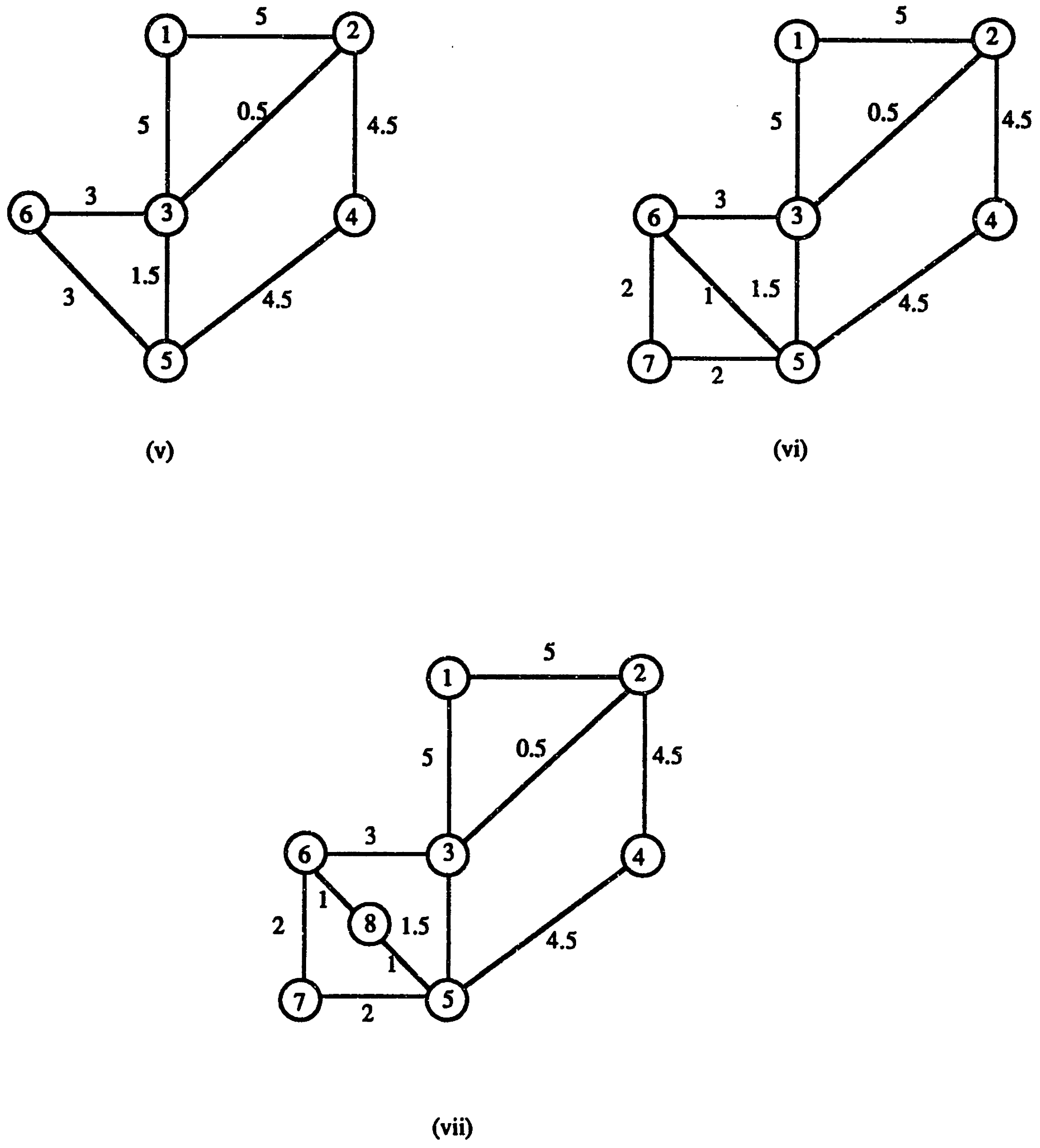


Figure 4.3: Output of algorithm B^* has eleven edges

4.4 A lower bound on the number of edges needed

In this section we show that any minimum capacity feasible network needs at least $n + t/5$ edges. Let us first assume that $t = n - 1$. That is,

$$r_1 = r_2 > r_3 > \cdots > r_n.$$

For $n \leq 3$, the problem is trivial. Therefore, assume $n \geq 4$. We are also working under the assumption that $r_{ij} = \min(r_i, r_j)$. For any cut $(S, \bar{S})$, let $k(S, \bar{S})$ denote its capacity.

Theorem 4.4 *Any feasible minimum capacity network requires at least* $n + (n-1)/5$ *edges.*

Proof: We need some preliminary lemmas.

Lemma 4.1 *Let* $(S, \bar{S})$ *be a cut. Then,*

$$k(S, \bar{S}) \geq \min\,[\max_{i \in S} r_i,\ \max_{i \in \bar{S}} r_i].$$

Proof: This result follows trivially from the weak version of the max-flow min-cut theorem. □

Lemma 4.2 *Let i and j be two adjacent nodes in the network satisfying the property that one of r_i or r_j does not equal $w(1)$. Let k_{ij} be the capacity of edge (i, j). Then,*

$$k_{ij} \leq \min(r_i, r_j)/2.$$

Proof: Assume $r_i \leq r_j$. Let $S = \{i, j\}$. Consider the cut $(S, \bar{S})$. Since the network is of minimum capacity, the sum of capacities of all the edges incident to node i (j

respectively) equals r_i (r_j respectively). Now, if $k_{ij} > \min(r_i, r_j)/2 = r_i/2$, then the value of the cut $(S, \bar{S})$ satisfies

$$\begin{aligned} k(S,\bar{S}) &= r_i - k_{ij} + r_j - k_{ij} \\ &= r_i + r_j - 2k_{ij} \\ &< r_j. \end{aligned}$$

But by Lemma 4.1, the value of this cut must be greater than or equal to r_j. □

Proof of Theorem 4.4:

Lemma 4.2 implies that if node $i \neq 1$ is connected to node j and $i > j$, then node i has degree at least three. Thus, the only way a node $i \neq 1, 2$ has degree 2 in the network is if it is connected to two nodes both with node labels less than i and, moreover, both these have degree at least three; in other words, no degree two node is connected to another degree two node unless it is node 1 (in which case it can only be connected to node 2). Now, let g be the number of degree two nodes that are connected to another degree two node. By our previous argument, g can be at most two. If g is equal to two, contract the edge that connects these two degree two nodes to obtain a graph G' on $n-1$ nodes in which no degree two node is connected to another degree two node and if g is equal to zero, let $G' = G$. Let G' have $n - l$ nodes. G has l edges more than G'. Partition the node set V' of this new graph G' into degree two nodes and non-degree two nodes, T and $\bar{T}$. Further, partition $\bar{T}$ according to the number of nodes of T that they are connected to. Let t_0 nodes of $\bar{T}$ be connected to zero nodes of T, t_1 nodes of $\bar{T}$ be connected to exactly one node of T, and so on ending with t_k. Thus sum of the degrees of all the nodes of $\bar{T}$ is greater or equal to

$$(3t_0) + (2t_1 + t_1) + (t_2 + 2t_2) + (3t_3) + (4t_4) + \cdots + (kt_k).$$

The number of degree two nodes equals to $(t_1 + 2t_2 + \cdots + kt_k)/2$. Thus, the total number of edges in the network is at least $(3t_0+4t_1+5t_2+6t_3+8t_4+\cdots+(2k+2)t_k)/2$.

We will minimize this lower bound by first formulating this problem as an integer program with one constraint.

$$\begin{aligned} \text{Min} \quad & (3t_0 + 4t_1 + 5t_2 + 6t_3 + 8t_4 + \cdots + (2k+2)t_k) \\ \text{such that} \quad & (t_1 + 2t_2 + \cdots + kt_k)/2 + (t_0 + t_1 + t_2 + \cdots + t_k) \geq n - l \\ & t_i \geq 0, \ \textit{integer}, \ i = 0, 1, \cdots, n. \end{aligned}$$

We obtain a lower bound on the optimal value of this problem, by further relaxing the integrality restriction, producing the following linear program with one constraint.

$$\begin{aligned} \text{Min} \quad & (3t_0 + 4t_1 + 5t_2 + 6t_3 + 8t_4 + \cdots + (2k+2)t_k) \\ \text{such that} \quad & (2t_0 + 3t_1 + 4t_2 + 5t_3 + 6t_4 + \cdots + (k+2)t_k) \geq 2(n - l) \\ & t_i \geq 0, \ i = 0, 1, \cdots, n. \end{aligned}$$

But we can be solve this problem by inspection. Since $\min(3/2, 4/3, 5/4, 6/5, 8/6, 10/7, \cdots)$ equals $6/5$, the optimal value equals $12(n-l)/5$. Thus, the network G' must contain at least $6(n-l)/5$ edges. Now, since $n - l \geq n - 1$, G has at least $l + 6(n-l)/5 \geq n + (n-1)/5$. □

For the general case when $t \neq n - 1$, if $t \leq 2$ clearly n edges suffice and when $t \geq 3$, the following result can be proved.

Theorem 4.5 *Any minimum capacity feasible network uses at least* $n + t/5$.

Proof: The proof is exactly the same as above, except that now, if a degree two node is connected to another degree two node, both the nodes have the same node weight. Moreover, since $t \geq 3$, the resulting network has at least four nodes. Thus, we have $n - l \geq t$ which completes the proof of Theorem 4.4. □

Figure 4.4 gives an example with $n = 8$, $t = 6$ and uses only ten edges. Hence this lower bound cannot be improved without using the actual values of r in the lower bound.

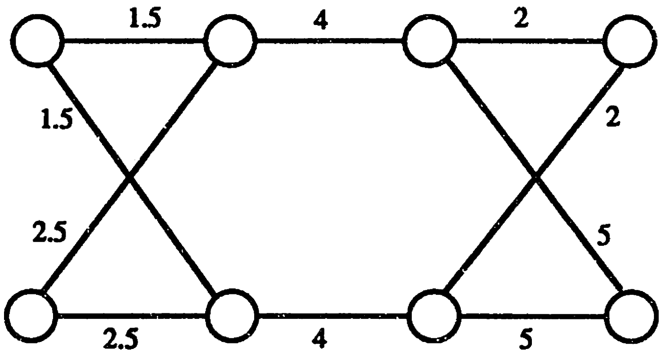


Figure 4.4: This network uses 10 edges on 8 nodes

Chapter 5

Disjoint $[s,t]$-dicuts

5.1 Introduction

Given a directed graph G with two distinguished nodes s and t, a dicut is a *minimal* set of arcs that (i) disconnects G, and (ii) whose arcs that emanate from the component containing node s are in one direction. A dicut is an (s,t)-dicut if in the disconnected graph s and t belong to different components and all the arcs emanating from the component containing node s point out of that component. Each arc has a nonnegative real cost c_e associated with it. For a subset A of arcs, $c(A) = \sum_{e \in A} c_e$ is the *cost* of A. The k-dicut problem considered by Wagner [60], is the problem of finding k (s,t)-dicuts that are pairwise disjoint and that have the smallest possible total cost for the union of the k dicuts.

The related problem of computing minimum-cost k disjoint (s,t)-cuts can be transformed to the (s,t)-dicut problem [60]. We recall that a cut is a set of arcs that disconnects the graph. For the rest of this chapter, a cut will refer to a *minimal* cut; that is removing arcs of the cut results in exactly two components of the graph. This problem generalizes the classical problem of computing the minimum-cost cut in a directed graph and is motivated by applications in network reliability computations. Disjoint (s,t)-cuts can be used to bound two-terminal reliability as follows. Let $C_1, C_2, \dots, C_k$ be k edge disjoint (s,t)-cuts in G. Suppose edge e of the graph fails with a probability p_e and is operational with a probability $1-p_e$. G is *operational* if it has at least one path from node s to node t. G is operational only if every (s,t)-cut in G has at least one operational edge. Therefore, the probability that G contains an (s,t)-path is bounded above by $\prod_{i=1}^{k}(1-\prod_{e\in C_i} p_e)$. This bound is very easy to compute and has been found to be competetive with more sophisticated bounds (see Colbourn [10]).

Wagner [60] solves this problem by formulating it as a linear program; the extreme points of this linear program are the incidence vectors of subgraphs of G that have a partition into dicuts, exactly k of which are (s,t)-dicuts. In this chapter, we characterize such subgraphs in terms of (s,t)-paths. This result gives insight into the nature of Wagner's result and makes the generalization from the 1-dicut problem more transparent. In Section 5.2 we state some preliminary results, in Section 5.3 we describe Wagner's solution, in Section 5.4 we state and prove our main results, in Section 5.5 we study a problem that is dual to the disjoint (s,t)-dicut problem, and in Section 5.6 we look at path formulations of a recent generalization of the max-flow min-cut theorem due to Nishihara and Inoue [54] and Orlin and Wagner [55].

5.2 Definitions and some preliminary lemmas

Let $G=(V,A)$ be a directed graph with n nodes and m arcs. A path in G does *not* mean a directed path unless specified otherwise. Since we are interested in dicuts that separate s and t, we assume that G is 2-vertex connected and that all the arcs incident to t are directed into t. Let H be a subgraph of G such that H can be partitioned into t dicuts, of which exactly k are (s,t)-dicuts. Call all such subgraphs *$k-(s,t)$-dicut subgraphs.*

In G, with any path P between two nodes a and b, and with any cycle C with a particular orientation, we associate a m-element $\{+1,0,-1\}$ column vector, which by a slight abuse of notation we also denote by P and C. This vector consists of 0 in the ith position if arc i does not belong to the path or cycle, $+1$ if the direction of the path or cycle agrees with that of the arc, and -1 otherwise. We begin with the following well-known result.

Lemma 5.1 *Let G be a directed graph and let T be a spanning tree of G. Let P be a path in T between any two nodes of G, and let P' be any path in G between the same two points. Then,*

$$P' = P + \sum \lambda_i C_i$$

where C_i are the fundamental cycles created by arcs in $P' \setminus T$ and $\lambda_i \in \{+1,0,-1\}$.

Proof: Let P and P' be paths joining s to t. The proof is by induction on $|P'|$, the case $|P'| = 1$ being trivial. Let $|P'| = l$ and suppose the statement of the lemma is true for all paths in G of length $< l$. If P and P' have an arc in common, then if we invoke the induction argument, the lemma is clearly true. So assume that P and P' are arc-disjoint. Since both the paths start and end at the same point and are arc-disjoint, at least one fundamental cycle created by an arc of $P' \setminus T$ contains a segment of P. Let (a,b) denote that arc and let C denote the associated fundamental cycle. Let P_{ab} be the $\{+1,0,-1\}$ vector corresponding to the single edge (a,b). Now let P_1 and P_2 be the s to a and b to t paths respectively, and P_1' and P_2' be the corresponding paths in P'. By the induction argument,

$$\begin{aligned} P'_1 &= P_1 + \sum \lambda_i^1 C_i^1 \\ P'_2 &= P_2 + \sum \lambda_i^2 C_i^2. \end{aligned}$$

Note that, if the direction of (a,b) agrees with the s to t path of $P_1 \cup P_2 \cup (a,b)$,

$$P + C = P_1 + P_2 + P_{ab}$$

and if it does not,

$$P - C = P_1 + P_2 - P_{ab}.$$

Now in the former case,

$$\begin{aligned} P' &= P'_1 + P'_2 + P_{ab} \\ &= P_1 + \sum \lambda_i^1 C_i^1 + P_2 + \sum \lambda_i^2 C_i^2 + P_{ab} \\ &= P + \sum \lambda_i C_i, \end{aligned}$$

while when the direction of (a,b) does not agree with the path $P_1 \cup P_2 \cup (a,b)$ from s to t,

$$\begin{aligned} P' &= P'_1 + P'_2 - P_{ab} \\ &= P_1 + \sum \lambda_i^1 C_i^1 + P_2 + \sum \lambda_i^2 C_i^2 - P_{ab} \\ &= P + \sum \lambda_i C_i. \end{aligned}$$

□

The following is a well known theorem due to Lovasz [43].

Theorem 5.1 *[43] G is 2- vertex connected if and only if for every edge (a,b) in the graph, the graph has an $a-b$ numbering: a numbering of the nodes from 1 to n, with a being assigned the number 1 and b the number n, with the property that every other node has a neighbor with a lower number as well as one with a higher number.*

□

As a consequence of this theorem, we can prove,

Lemma 5.2 *In a 2-vertex connected graph G, for any pair of nodes s and t and edge (a, b), the graph contains a path from node s to node t that contains edge (a, b).*

Proof: Consider an $a - b$ numbering of the graph which exists by Theorem 5.1, and assume without loss of generality that node s gets a lower number than node t. Then by starting from node s, and always taking the edge to a lower numbered node, we get a path from s to a and by starting from t always taking an edge to a higher numbered node, we obtain a path from node t to node b and these two paths are disjoint. Thus we have a path from s to t that uses the edge (a, b). □

Lemma 5.3 *In a 2-vertex connected graph G, for any pair of nodes s and t and edge (a, b), the graph contains a (s, t)-cut containing (a, b).*

Proof: Consider an $a - b$ numbering and assume without loss of generality that s has a lablel less than t. Let C be the set of arcs between all nodes with labels less than that of t and those with labels greater than or equal to that of t. By Theorem 5.1, the graph contains a path from node a (which has label 1) to every node with label less than that of t, and a path from b (which gets label n) to every node with label greater than or equal to that of t; moreover the labels of the nodes on each of these paths are either increasing or decreasing. Therefore, removing the arcs in C disconnects G into exactly two components, implying that C is an (s, t)-cut. Clearly, (a, b) belongs to C, proving the Lemma. □

5.3 Wagner's solution of the k-(s, t)-dicut problem

We briefly describe Wagner's solution - for a more detailed description, see [60]. To G, we add a set of $k - 1$ nodes and a directed path S of length k going from node s to node t and using only these new $k - 1$ nodes. Call this new graph G'. Since every

(s,t)-dicut of G' uses exactly one arc from this path and the (s,t)-dicuts of G are of the form $K \setminus S$ for some (s,t)-dicut K of G'. If we assign a cost of zero to each of these new arcs, solving the k (s,t)-dicut problem in G' solves the problem for G.

Let N' be the vertex-arc incidence matrix of G' with one row removed and let T' be a spanning tree of G'. Let B' be the submatrix of N' corresponding to the arcs in T'. Then the matrix $B'^{-1}N$, which is known as the *fundamental-arc matrix* of G' is of the form $[I \;\; R]$. The matrix $[-R^T \;\; I]$ known as the *fundamental-cut matrix* of G'. The rows of $[-R^T \;\; I]$ are the incidence vectors of fundamental cycles created by each of the arcs not in T'.

Consider the following polytope:

$$\begin{aligned} (\,-R^T \quad I\,)x &= 0 \\ 0 \le \; x_e \; &\le 1 \quad e \in A \\ x_e \; &= \; 1 \quad e \in S. \end{aligned}$$

The following is the main result of [60]:

Theorem 5.2 *Let x^* be an extreme point of this polytope. Then $x^*_e \in \{0,1\}$. Moreover, the set $\{e \in A \cup S \,|\, x^*_e = 1\}$ has a partition $\{K_1,\ldots,K_t\}$ into pairwise-disjoint dicuts of G' with exactly k of the dicuts being (s,t)-dicuts.*

5.4 Characterization of $k - (s,t)$-dicut subgraphs

Given an (s,t)-path P, let $f(P)$ be the set of forward arcs of P, i.e. arcs whose direction agrees with that of P in the direction from node s to node t and let $r(P)$ be the set of reverse arcs of P, i.e. arcs whose direction disagrees with that of P from node s to node t.

We prove the following:

Theorem 5.3 *The vertices of the following polytope which we refer to as Q, are the incidence vectors of $k-(s,t)$-dicut subgraphs:*

$$x(f(P)) - x(r(P)) = k, \ \forall\, (s,t)\text{-paths } P$$

$$0 \le x \le 1.$$

The following result is a corollary this theorem.

Corollary 5.1 *H is a $k-(s,t)$-dicut subgraph if and only if H satisfies the condition*

$$|f(P)\cap H| - |r(P)\cap H| = k$$

for all (s,t)-paths P.

Proof: The *(if)* part follows from the theorem.

(only if): Let $H = H_1 \cup H_2 \cup \ldots \cup H_k \ldots \cup H_t$ be a partition of the arc set H into dicuts $H_1, H_2, \ldots, H_t$, i.e., $H_i \cap H_j = \emptyset$, when $i \neq j$; and $H_1, H_2, \ldots, H_k$ are the k (s,t)-dicuts. Then for every (s,t) path P, $|f(P)\cap H_i| - |r(P)\cap H_i| = 1$, if $i \le k$, because every time the path leaves the component containing s, it contributes one to the first term, and whenever it enters the component it contributes one to the second term, and every (s,t) path leaves the component once more than it enters. For $i > k$, every path crosses the dicut while going one way as many times as it does the other way, contributing nothing to the righthand side. Since the sets are disjoint, H satisfies the condition $|f(P)\cap H| - |r(P)\cap H| = k$. □

Proof of Theorem 5.3: We will show that polytope (Q) is a reformulation of Wagner's polytope for a specific choice of a spanning tree of G'. Recall that, by assumption, G is 2-vertex connected and that t has only incoming arcs. Thus, let T be a spanning tree of $G - \{t\}$. Then $T \cup S$ is a spanning tree of G'. The columns of $-R^T$ corresponding to $e \in S$ has -1 in the rows corresponding to arcs incident to node t and 0 elsewhere.

Of all the arcs of G, let us give the arcs in T the first $(n-2)$ labels and the arcs coming into t, the last labels. Consider the following matrix M and vector d; M has m columns and $(m-(n-2))$ rows, and d has $(m-(n-2))$ elements.

$$M = \begin{array}{c} \overbrace{1 \ldots n-2}^{T} \quad n-1 \qquad \ldots \quad \ldots \quad m \\ \left(\begin{array}{ccccc} & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & \ddots \\ & & & & & 1 \end{array} \right) \end{array}$$

and

$$d = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ k \\ k \\ \vdots \\ k \end{pmatrix},$$

where the k's correspond to the edges that come into the node t.

The rows of M correspond to the arcs of G that are not in T. If arc i is not incident to node t, then adding arc i closes a cycle in G. The first $n-2$ elements of the row corresponding to i consists of:

$$m_{ij} = \begin{cases} 0 & \text{if edge } j \text{ does not belong to the cycle} \\ +1 & \text{if the direction of } j \text{ agrees with that of } i \\ -1 & \text{if the direction of } j \text{ does not agree with that of } i \end{cases}$$

If arc i is incident to node t, then adding arc i closes a (s,t)-path in G. The first $n-2$ elements of the row corresponding to i consists of:

$$m_{ij} = \begin{cases} 0 & \text{if edge } j \text{ does not belong to the path} \\ +1 & \text{if } j \text{ is a forward arc in the path} \\ -1 & \text{if } j \text{ is a reverse arc in the path} \end{cases}$$

We show that

$$\begin{array}{lccc} (R) & Mx & = & d \\ & & 0 \leq x \leq 1 & \end{array}$$

represents the same polytope as Q.

$R \supseteq Q$: We have to show that every equality of R is implied by the system defining Q.

The equalities of $Mx = d$ that correspond to edges directed to node t are clearly among the equalities of Q. Therefore, we have to look only at equalities of $Mx = d$ corresponding to edges that are not in the tree T and that are not incident to node t. Let i be such an arc and let C be the fundamental cycle created by i. Lemma 5.2 implies that we can find two paths P and P' from node s to node t, whose symmetric difference is exactly C. Let P' be the path that uses arc i. Let

$$\begin{array}{llcl} A: & x(f(P)) - x(r(P)) & = & k \\ B: & x(f(P')) - x(r(P')) & = & k \end{array}$$

be the two equations in Q corresponding to P and P' respectively. Consider $A - B$, if the arc i is forward in P' and $B - A$, if it is reverse in P'. In either case, we generate the desired equality of $Mx = d$.

$Q \supseteq R'$: We have to show that for every path P, we can obtain

$$x(f(P)) - x(r(P)) = k$$

by adding and subtracting some rows of Q.

Let j be the last arc of P. Let P' be the path formed in T by adding j to T. Since a row in Q corresponds to the path P'. From Lemma 5.1 we can obtain P from P' by adding and subtracting the rows corresponding to the fundamental cycles.

Moreover, the system $Mx = d$ is the same as Wagner's polytope for the choice of the spanning tree T' and by substituting the variables corresponding to the path S and bringing it to the right-hand side. Thus we have proved the theorem. □

5.5 The Dual problem

We next consider a problem that in some sense is dual to the problem that we have been considering: Find k disjoint (s,t)-directed paths of minimum-cost in G. We can easily solve this problem by solving a min-cost flow problem of sending k units of flow at minimum cost from node s to node t in G with arc capacities all equal to one. Let N be the vertex-arc incidence matrix of G with the row corresponding to t removed and let d be a vector with k in the row corresponding to node s and 0 elsewhere. The vertices of the following polytope which we refer to as F, are the incidence vectors of subgraphs of G that have a partition into k disjoint (s,t)-directed paths and directed cycles.

$$\begin{array}{lccc} (F) & Nx & = & f \\ & & 0 \leq x \leq 1 & \end{array}$$

Call all such subgraphs $k-(s,t)$-dipath subgraphs. We can prove results dual to Theorem 5.3 and Corollary 5.1.

Theorem 5.4 *The vertices of the following polytope which we refer to as J, are the incidence vectors of $k-(s,t)$-dipath subgraphs:*

$$x(f(C)) - x(r(C)) = k, \ \ \forall (s,t)\text{-cuts } C$$

$$0 \leq x \leq 1.$$

The following result is a corollary of this theorem.

Corollary 5.2 *H is a $k-(s,t)$-dipath subgraph if and only if H satisfies the condition*

$$|f(C) \cap H| - |r(C) \cap H| = k$$

for all (s,t)-cuts C.

Proof: The proof of the corollary is similar to the proof of Corollary 6 and can be obtained by interchanging paths and cuts, directed paths and dicuts adn so forth. We omit the details. □

Proof of Theorem 5.4: As before, we show that J and F are the same polytopes.

$J \supseteq F$: Let C be a (s,t)-cut. We obtain the equality of J corresponding to C from those of F by adding all the equalities of F corresponding to the nodes of G that are in the component of G containing s that we obtain after removing the arcs in C.

$F \supseteq J$: The equality of F corresponding to the node s is clearly among the equalities defining J. Consider an equality of F corresponding to a node i. Let C be a (s,t)-cut in G satisfying the properties that removing arcs in C, (i) node i belongs to the component containing node s, and (ii) C contains at least one arc that is incident to node i (Lemma 5.3 implies the existence of such a cut). If we remove node i from this component, we obtain another (s,t)-cut C' and, subtracting the equalities of J corresponding to C and C', we obtain the desired equality of F corresponding to node i. □

5.6 Path formulations of generalizations of the max-flow and min-cut problems

The well known max-flow min-cut theorem of Elias, Feinstein and Shannon [18] and Ford and Fulkerson [20] states that the value of a maximum flow is equal to the capacity of a minimum cut. Recently Nishihara and Inoue [54] and Orlin and Wagner [55] have given generalizations of both the maximum flow problem and the minimum cut problem, and proved an analogue of the max-flow min-cut theorem. In this section we establish easy extensions of the path formulations of the previous sections to these more general problems.

Let u and c be two non-negative vectors of capacities and costs defined on the arc set of G. A (k, c)-*cut* is a collection of k cuts satisfying the property that arc e of G belongs to at most c_e of the cuts. The capacity of a cut is the sum of the capacities of all the forward arcs in the cut. The *capacity* of a (k, c)-cut is the sum of the capacities of the cuts in it. A *minimum* (k, c)-cut is one of minimum capacity. The (k, c)-*cut problem* on (G, s, t, u) is that of finding a minimum (k, c)-cut, if one exists. If c is a vector of all $+1$'s, then the problem is that of finding k pairwise edge-disjoint cuts of minimum capacity.

The (k, c)-*flow problem* (G, s, t, u) is the problem of finding a vector $u' \geq u$, if one exists, and a maximum flow f in (G, s, t, u') that maximizes the value of a (k, c)-flow,

$$\text{val}(k, c, f) := k\text{val}(f) - \sum_{e \in A} c_e(u'_e - u_e).$$

The interpretation is that the capacity of any arc e can be increased at a cost of c_e per unit. Each unit of flow sent from node s to node t yields a payoff of k dollars. The problem is to determine how much extra arc capacity to purchase and how much to send so as to maximize the net profit. The main result of [54] and [55] is that the value of a maximum (k, c)-flow is equal to the capacity of a minimum (k, c)-cut. Next we will relate this result to our work in Section 5.4.

Finding a minimum-cost k-cut is related to finding the minimum-cost k-dicut by the following transformation: replace each arc $e = (x, y)$ by a new vertex w and the directed arcs $f = (x, w)$ and $g = (y, w)$; and define $u_f = u_e$ and $u_g = 0$. Then a minimum cost k-cut in the original network corresponds to a minimum cost k-dicut in the transformed network. Since we now have a cost associated with the arcs, define $c_f = c_e$ and $c_g = c_e$. It is clear that we can obtain a minimum-cost (k,c)-cut by finding a minimum-cost (k,c)-dicut in this transformed digraph. Therefore, we will consider the (k,c)-dicut problem. Next we prove a generalization of Theorem 5.3. The proof is essentially a combination of Theorem 5.3 and Theorem 1 of Wagner[60].

Let $(-R^T \ \ I)$ be a fundamental-cut matrix, and consider x and y in the null space of $(-R^T \ \ I)$. Then y *conforms* to x if whenever $y_e \neq 0$ for some e, then $x_e y_e > 0$. The support of x is $\{e | x_e \neq 0\}$. The vector x is *elementary* if it has non-empty support and no other vector in the null space of $(-R^T \ \ I)$ has support properly contained in that of x. An elementary vector is *primitive* if its components belong to the set $\{1,-1,0\}$. Dicuts correspond precisely to the supports of nonnegative elementary vectors. A *conformal decomposition* of x is a set of vectors $\{x^1,\ldots,x^t\}$ satisfying the properties that

1. x^i is primitive and conforms to x, for $1 \leq i \leq t$, and
2. $x = \sum_{i=1}^{t} \beta_i x^i$ for nonnegative scalars $\beta_1,\ldots,\beta_t$.

Every non-zero vector in the null space of $(-R^T \ \ I)$ has a conformal decomposition (see Fulkerson [22]).

Define a (k,c)-dicut subgraph as a union of dicuts of G that contains (i) exactly k of (s,t)-dicuts, and (ii) arc e in at most c_e of the dicuts.

Theorem 5.5 *The vertices of the following polytope are the incidence vectors of (k,c)-dicut subgraphs:*

$$x(f(P)) - x(r(P)) = k, \ \ \forall (s,t)\textit{-paths } P$$

$$0 \leq x \leq c.$$

Proof: As shown in the proof of Theorem 5.3, if S is a set of k new arcs constituting a directed path from s to t, the system

$$x(f(P)) - x(r(P)) = k, \quad \forall (s,t)\text{-paths } P$$

is the same as

$$\begin{aligned} (\,-R^T \quad I\,)\,x &= 0 \\ x_e &= 1 \quad e \in S. \end{aligned}$$

Let x^* be an extreme point of the polytope

$$\begin{aligned} (\,-R^T \quad I\,)\,x &= 0 \\ 0 \leq x_e &\leq c_e \quad e \in A \\ x_e &= 1 \quad e \in S. \end{aligned}$$

The total unimodularity of $(-R^T \; I)$ implies that x^* is an integer vector (see Hoffman and Kruskal [36]). Let $\{x^1, \ldots, x^t\}$ be a conformal decomposition of x^*. Each x^i is a $\{0,1\}$ vector since it is primitive and conforms to x^* (note that x^* is nonnegative). Since each x^i is elementary and nonnegative, its support is a dicut of G. If K_i is a subgaph of G corresponding to the support of x^i, the collection $\{K_1, \ldots, K_t\}$ is a collection of dicuts of G and each arc e of G appears in at most c_e of these dicuts. Since a dicut of G is an (s,t)-dicut if and only if it contains exactly one arc of S, the constraints $x_e = 1$ for $e \in S$ force exactly k of the dicuts in the partition to be (s,t)-dicuts. □

Next, we explore the connection between the (k,c)-dicut problem and the (k,c)-flow problem. The following theorem is the main result of [55] and [54].

Theorem 5.6 *The value of a maximum (k,c)-flow is equal to the capacity of a minimum (k,c)-cut.*

Assume that we have performed the following transformation to G mentioned previously: replace each arc $e = (x,y)$ by a new vertex w and two new arcs $f = (x,w)$

and $g = (y, w)$ with $c_f = c_g = c_e$ and $u_f = u_e$ and $u_g = 0$. Call this transformed graph G^*. Then we obtain a minimum (k, c)-cut in G by solving the following linear program defined on G^*.

$$\begin{array}{lll}
Min & \sum_{e \in A(G^*)} u_e x_e & \\
& \text{subject to} & \\
& x(f(P)) - x(r(P)) = k & \forall (s,t)\text{-paths } P \\
& -x_e \geq -c_e & \forall e \in A(G^*) \\
& 0 \leq x_e & \forall e \in A(G^*).
\end{array}$$

Consider the linear programming dual to this linear program with dual variables y_P corresponding to the path equalities and the dual variables z_e corresponding to the greater than or equal to inequalities. The dual is

$$\begin{array}{llll}
& Min & \sum_{(s,t)\text{-paths } P} y_P - \sum_{e \in A(G^*)} c_e z_e & \\
& & \text{subject to} & \\
(D) & & \sum_{\{P \mid e \in f(P)\}} y_P - \sum_{\{P \mid e \in r(P)\}} y_P - z_e \leq u_e & \forall e \in A(G^*) \\
& & 0 \leq z_e & \forall e \in A(G^*).
\end{array}$$

Notice that the y_P variables can assume negative values. However, the transformation from G to G^* ensures that the flow on any arc in G is non-negative. (If we have replaced arc $e = (x, y)$ is replaced by $f = (x, w)$ and $g = (y, w)$, consider the inequalities corresponding to the arc g - they ensure that the flow on any arc is greater than or equal to $u_g + z_g \geq z_g \geq 0$. Thus, $\sum_{(s,t)\text{-paths } P \text{ in } G^*} y_P$ gives the value of the flow in G. Consequently, the objective function of the linear program D gives the value of the (k, c)-flow val(k, c, f).

Chapter 6

Summary and Open Problems

In this thesis we have studied issues that arise in the design and analysis of networks with various connectivity requirements. These networks are of importance in the design of telecommunication networks whenever facilities are subject to failure. This general problem can give rise to various models; we have concentrated on only a few. Specifically, we have studied problems defined on the plane with Euclidean distances as costs; we have considered situations that permit Steiner points and those that don't. We conclude this thesis with a brief summary of our contributions and with some open problems related to this research.

In Chapter 2 we studied the design of k-connected networks with Steiner points on the plane when the terminals are vertices of a convex polygon. We characterized the structure of optimal solutions for these problems; we showed that the optimal design is $k/2$ copies of the boundary when k is even, and $k - 1/2$ copies of the boundary plus one copy of the Steiner tree, when k is odd. When combined with some prior results of Provan on the computation of optimal Steiner trees on the plane, these results lead to a fully polynomial approximation scheme for the design of minimum cost networks when the connectivity requirement between each pair of nodes is odd. The methods we have employed use some graph-theoretic properties and exploit the geometry of the plane. The most important open problem here is the complexity of finding minimum cost two-vertex and two-edge connected networks in τ-planar graphs when we do not allow the duplication of edges. We suspect that this problem might be solvable in polynomial time.

Also, no one has yet shown how to construct an Euclidean Steiner tree for the "convex-hull" arrangement of terminals. We have a conjecture related to this problem. Du and Hwang [17] have recently settled the long-standing Gilbert and Pollack conjecture that states that the ratio of the Euclidean Steiner tree to the Euclidean spanning tree is always greater than or equal to $\sqrt{3}/2 = 0.866\ldots$ This bound is tight as the simple example of three vertices of an equilateral triangle shows. In fact, appropriately piecing together three-terminal sets shows that this bound is tight for any number of terminals. However, this example does not satisfy the "convex-hull" property. For terminals that satisfy this property, we can approach this bound arbitrarily closely by taking three terminals as forming the vertices of an equilateral triangle and then taking all the other vertices very close to one of the terminals. If, however, we insist that the terminals constitute the vertices of a convex polygon and moreover the edges of the polygon fall within certain upper and lower bounds, we suspect the ratio is much higher. In fact, even for four terminals, this ratio seems to be higher. Du et. al. [16] provide some evidence to support our conjecture: they showed that for six or more terminals, if all the sides of the convex polygon are equal, then the ratio is equal to one.

In Chapter 3 we studied the design of minimum cost k-connected networks without Steiner points. We showed that the problem always has a solution in which no two edges of the optimal design cross each other. For k odd, we have given a class of counterexamples that show that the optimal design might have crossing edges. We have also established that the support of the linear programming relaxation of a integer programming formulation of this problem has the same "no-crossing" property.

In Chapter 4 we studied a variation of the classical network synthesis problem, that is the problem of designing a network to satisfy given flow requirements. We gave a new algorithm for designing such networks. This design has the property that it uses fewer (or no more) edges than other algorithms from the literature.

In Chapter 5 we studied some issues that arise in the computation of network reliability. This chapter is of mostly theoretical interest and gives some new insight into recent results of Wagner, Nishihara and Inoue, and Orlin and Wagner. Specifically, we have given path formulations for the problem of finding k disjoint dicuts between a specified pair of nodes. This formulation extends the well-known path formulations for the max-flow problem. An open problem is to establish a connection between this work and a well known min-max theorem in combinatorics called the Lucchesi-Younger theorem. A *dicut covering* of a directed graph is defined as a subset of arcs satisfying the property that every dicut uses at least one element of this subset. The Lucchesi-Younger theorem states that

Theorem 6.1 *[45] The maximum number of (edge) disjoint dicuts in a directed graph is equal to the minimum cardinality of a dicut covering.*

The results of Wagner give a linear programming solution to a two-node version of this theorem. Whether similar techniques would provide a linear programming solution to the all-pair case stated in the theorem is an open question. In [21], Frank gives a $O(n^5)$ algorithm for finding the minimum-cost covering in a directed graph. Finding a linear programming based theory for the Luchessi-Younger theorem might lead to a better complexity algorithm.

Bibliography

[1] A. Balakrishnan, T. L. Magnanti and R. T. Wong, " A dual-ascent procedure for large-scale uncapacitated network design", *Operations Research*, volume 32, **5**, 716-740 (1989).

[2] D. Bienstock, E. F. Briskell and C. L. Monma, "Properties of k-connected networks", Technical Report, Bell Communications Research, Morristown, NJ (1987).

[3] D. Bienstock, and C. L. Monma, "Optimal enclosing regions in planar graphs", *Networks* **19**, 79-94 (1989).

[4] R. G. Bland, D. Goldfarb and M. J. Todd, "The ellipsoid method: a survey", *Operations Research*, **29**, 1039-1091 (1981).

[5] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications.* North-Holland, New York (1976).

[6] N. Christofides and C. A. Whitlock, "Network synthesis with connectivity constraints-a survey". In J. P. Brans (Ed.) *Operational Research '81*, North-Holland, Amsterdam 705-723 (1981).

[7] E. J. Cockayne, "Computation of minimal length full Steiner trees on the vertices of a convex polygon", *Mathematics of Computation*, **23**, 521-531 (1969).

[8] E. J. Cockayne, "On the efficiency of the algorithm for Steiner minimal trees", *SIAM Journal on Applied Mathematics*, **18**, 150-159 (1970).

[9] C. J. Colbourn, *The Combinatorics of Network Reliability*, Oxford University Press (1987).

[10] C. J. Colbourn, "Edge-packings of graphs and network reliability", *Discrete Mathematics*, **72**, 49-61 (1987).

[11] G. A. Croes, "A method for solving traveling-salesman problems", *Operations Research*, **6**, 791-812 (1958).

[12] M. B. Dillencourt, "Toughness and Delaunay triangulations", Proceedings of the Third ACM Symposium on Computational Geometry (1987).

[13] M. B. Dillencourt, "A non-Hamiltonian, nondegenerate Delaunay triangulation", *Information Processing Letters*, **25**, 149-151 (1987).

[14] M. B. Dillencourt, "Traveling salesman cycles are not always subgraphs of Delaunay traingulations or of minimum weight triangulations", *Information Processing Letters*, **24**, 339-342 (1987).

[15] S. E. Dreyfus and R. A. Wagner, "The Steiner Problem in graphs", *Networks* **1**195-207 (1972).

[16] D. Z. Du, F. K. Hwang and S. C. Chao, "Steiner minimal trees for points on a circle", *Proc. Amer. Maih. Soc.*, **95**, 613-618 (1985).

[17] D. Z. Du and F. K. Hwang, "A proof of Gilbert-Pollak's conjecture on the Steiner ratio", Bell Laboratories, Murray Hill, NJ, preprint (1990).

[18] P. Elias, A. Feinstein and C. E. Shannon, " A note on the maximum flow through a network". *IRE Transactions on Information Theory*, **2**, 117-119 (1956).

[19] R. E. Erickson, C. L. Monma, and A. F. Veinott, "Send and split method for minimum-concave-cost network flows", *Mathematics of Operastions Research*, **12**, 634-664 (1987).

[20] L. R. Ford Jr. and D. R. Fulkerson, " Maximal flow through a network", *C*[illegible]*a-dian Journal of Mathematics*, **8**, 399-404 (1956).

[21] A. Frank, "How to make a digraph strongly connected", *Combinatorica*, **1**, 145-153 (1981).

[22] D. R. Fulkerson, "Networks, frames and blocking systems". In *Mathematics of the Decision Sciences, Part 1*, eds. G. B. Dantzig and A. F. Veinott. American Mathematical Society, Lectures in Applied Mathematics Vol.11, Providence, R.I, 303-334 (1968).

[23] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. H. Freeman and Co., San Francisco (1979).

[24] E. N. Gilbert and H. O. Pollak, "Steiner minimal trees", *SIAM Journal on Applied Mathematics*, **16**, 1-29 (1968).

[25] M. X. Goemans, " Analysis of Linear Programming Relaxations for a class of Connectivity Problems", Ph.D Thesis, Sloan School of Management, MIT (1990).

[26] M. X. Goemans and D. J. Bertsimas, "On the parsimonious property of connectivity problems", Proceedings of the 1st ACM-SIAM Symposium On Discrete Algorithms, San Fransisco, CA (1990).

[27] M. X. Goemans and K. T. Talluri, "2-change for k-connected networks", *Operations Research Letters*, March (1991).

[28] R. E. Gomory and T. C. Hu, "An application of generalized linear programming to network flows", *IBM Research Report*, RC-319 (1960).

[29] R. E. Gomory and T. C. Hu, "Multi-terminal network flows", *SIAM Journal of Applied Mathematics*, **9**, 551-570 (1961).

[30] D. Gusfield, "Simple constructions for the multi-terminal network flow synthesis", Department of Computer Science, Yale University, preprint.

[31] M. Grötschel, C. L. Monma and M. Stoer, "Facets for polyhedra arising in the design of communication networks with low-connectivity constraints", Report 187, Institut für Mathematik, Universität Augsburg (1990).

[32] M. Grötschel, C. L. Monma and M. Stoer, "Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints", Report 188, Institut für Mathematik, Universität Augsburg (1990).

[33] M. Grötschel, C. L. Monma and M. Stoer, "Polyhedral approaches to netwrok survivability", Report 189, Institut für Mathematik, Universität Augsburg (1990).

[34] S. L. Hakimi, "Steiner's problem in graphs and its implications", *Networks*, **1**, 113-133 (1971).

[35] M. Held and R. M. Karp, "The traveling-salesman problem and minimum spanning trees", *Operations Research*, **18**, 1138-1162 (1970).

[36] A. J. Hoffman and J. B. Kruskal, "Integral boundary points of convex polyhedra". In *Linear Inequalities and Related Systems*, ed. H. W. Kuhn and A. W. Tucker, 223-246. Princeton University Press, Princeton, N.J (1956).

[37] F. K. Hwang, G. D. Song, G. Y. Ting and D. Z. Du, "A decomposition theorem on Euclidean Steiner minimal trees", *Discrete and Computational Geometry*, **3**, 367-382 (1990).

[38] V. Kantabutra, "Traveling salesman cycles are not always subgraphs of Voronoi duals", *Information Processing Letters*, **16**, 11-12 (1983).

[39] J. Krarup, "The generalized Steiner problem". Unpublished note (1978).

[40] J. B. Kruskal Jr., "On the shortest spanning subtree of a graph and the travelling salesman problem", *Proceedings of the American Mathematical Society*, **7**, 48-50 (1956).

[41] A. J. Levin, "Algorithm for the shortest connection of a group of graph vertices", *Soviet Mathematics Doklady*, **12** 1477-1481 (1971).

[42] S. Lin, "Computer solutions to the traveling salesman problem", *Bell System Tech. J.*, **44**, 2245-2269 (1965).

[43] L. Lovasz,"*Combinatorial Problems and Exercises*", North Holland, Amsterdam (1979).

[44] L. Lovasz, "On some connectivity properties of eulerian graphs", *Acta Mathematica Academia Sinica Hungaria*, **228**, 129-138 (1976).

[45] C. L. Lucchesi and D. H. Younger, " A minimax relation for directed graphs", *Journal of the London Mathematical Society* (2), **17**, 369-374 (1978).

[46] T. L. Magnanti and R. T. Wong, "Network design and transportation planning: models and algorithms". *Transportation Science*, **18**, 1-55 (1984).

[47] C. L. Monma, B. S. Munson and W. R. Pulleyblank, "Minimum weight two-connected spanning networks", *Mathematical Programming*, **46**, 153-172 (1990).

[48] C. L. Monma and D. F. Shallcross, "Methods for designing communication networks with certain two-connected survivability constraints", *Operations Research*, **37**, 531-541 (1989).

[49] New York Times, "Phone system feared vulnerable to wider service disruptions", *New York Times*, (May 26 1988).

[50] New York Times, "Experts say phone system is vulnerable to terrorists", *New York Times*, (Feb 8 1989).

[51] Newark Star Ledger, "Damage to fiber cabel hinders phone service", *Newark Star Ledger*, (Sept 22 1987).

[52] Newark Star Ledger, "Cable snaps, snags area phone calls", *Newark Star Ledger*, (Feb 26 1988).

[53] Newark Star Ledger, "Phone snafu isolates New Jersey; long-distance cable snaps", *Newark Star Ledger*, Nov 19 (1988).

[54] O. Nishihara and K. Inoue, "An algorithm for a multiple disconnecting set problem", Department of Aeronautical Engineering, Kyoto university, Japan, preprint.

[55] J. B. Orlin and D. K. Wagner, "A Generalization of the Max-flow Min-cut Theorem", School of Industrial Engineering, Purdue University, preprint.

[56] R. C. Prim, "Shortest connection networks and some generalizations", *Bell Syst. Tech. Journal*, **36**, 1389-1401 (1957).

[57] J. S. Provan, "A polynomial algorithm for the Steiner tree problem on terminal planar graphs", Tech. Rep. 83/10, Department of Operations Research, University of North Carolina, Chapel Hill (1983).

[58] J. S. Provan, "Convexity and the Steiner tree problem", *Networks*, **18**, 55-72 (1988).

[59] K. Steiglitz, P. Weiner and D. J. Kleitman, "The design of minimum-cost survivable networks", *IEEE Transactions on Circuit Theory*, CT-16, 455-460 (1969).

[60] D.K.Wagner, "Disjoint (s,t)-cuts in a network", *Networks*, **20**, 361-371 (1990).

[61] P. Winter, "Generalized Steiner problem in outerplanar networks", *BIT*, **25**, 485-496 (1985).

[62] P. Winter, "Steiner problem in Networks: A survey", *Networks*, **17**, 129-167 (1987).

[63] P. Winter, "Generalized Steiner problem in series-parallel networks". To appear in *Journal of Algorithms*.

[64] P. Winter, "Generalized Steiner problem in Halin networks", Technical University, Denmark, preprint.

[65] R. T. Wong, " Dual ascent approach for Steiner tree problem on a directed graph", *Mathematical Programming*, **28**, 271-287 (1984).