**Chapter 32**

# A Commercial Application of Survivable Network Design: ITP/INPLANS CCS Network Topology Analyzer

Milena Mihail* David Shallcross†

and

Nate Dean‡ Marco Mostrel§

## Abstract

The ITP/INPLANS[1] CCS Network Topology Analyzer is a Bellcore product which performs automated design of cost effective survivable CCS (Common Channel Signaling) networks, with survivability meaning that certain path-connectivity is preserved under limited failures of network elements. The algorithmic core of this product consists of suitable extensions of primal-dual approximation schemes for Steiner network problems.

Even though most of the survivability problems arizing in CCS networks are not strictly of the form for which the approximation algorithms with proven performance guarantees apply, we implemented modifications of these algorithms with success: In addition to duality-based performance guarantees that indicate, mathematically, discrepancy of no more than 20% from optimality for generic Steiner problems and no more than 40% for survivable CCS networks, our software passed all commercial benchmark tests, and our code was deployed with the August '94 release of the product.

CCS networks fall in the general category of low bit-rate backbone networks. The main characteristic of survivability problems for these networks is that each edge, once present, can be assumed to carry arbitrarily many paths. For high bit-rate backbone networks, such as the widely used ATM and SONET, this is no longer the case. We discuss versions of network survivability with capacitated edges that appear to model survivability considerations in such networks.

## 1 Introduction

In telecommunications we are often requested to design networks with connectivity properties that survive the failure of a (small) number of network elements. Moreover, our solutions should be cost effective.

In this extended abstract we focus on a class of survivable network problems as they arose in the automated design of CCS networks[2]. For these problems we implemented suitable adaptations of primal-dual heuristics that emerge from a sequence of recent theoretical results [7] [3] [10] [1] [4] [9] [8].

In theory, survivable network design considerations have been captured by the following formalization: Given an undirected graph with costs on its edges and non-negative path requirements $r_{uv}$ between certain pairs of vertices $u$ and $v$, find a low cost subgraph containing $r_{uv}$ (edge or vertex) disjoint paths between each pair $u$ and $v$. The interpretation is straightforward: if $r_{uv}-1$ network elements (links or nodes) fail, there is still a path from $u$ to $v$.

The above formalization can be viewed as an extension of the Steiner tree problem, thus it has

*Bell Communications Research and Columbia University: mihail@cs.columbia.edu.

†Bellcore: davids@bellcore.com.

‡AT&T Bell Laboratories: nate@research.att.com.

§Bellcore: marco@cc.bellcore.com.

[1]ITP stands for Integrated Technology Planning. INPLANS is a trademark of Bellcore.

[2]See [5] for a description of Common Channel Signaling in the context of telecommunications.

become known as the Steiner network problem. Of course, because they include Steiner tree as a special case, even the simplest forms of Steiner network problems are NP-complete [2]. We must therefore look for efficient approximations.

[10] gave the first polynomial time approximation for Steiner networks, with performance guarantee $2 \max_{u,v} r_{uv}$. Subsequently [1] obtained a faster implementation of the algorithm in [10], while [4] modified the algorithm and improved the performance guarantee to $2 \log \max_{u,v} r_{uv}$.

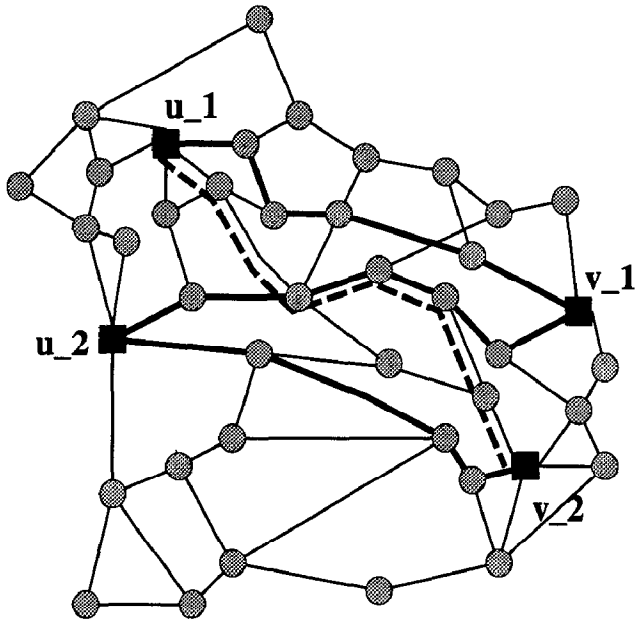


**Figure 1.** This instance can be thought of as representative of a local exchange telecommunications network, with the exception that real networks have twice to three times as many nodes (not shown here due to lack of space). A representative survivability requirement would be: indicate paths from $U = \{u_1, u_2\}$ to $V = \{v_1, v_2\}$ so that, if two nodes fail (other than both nodes in $U$ or both nodes in $V$), at least one path from $U$ to $V$ survives. We indicate above four $U$-$V$ paths satisfying such connectivity.

All the above algorithms first express path requirements in terms of equivalent requirements of covering cuts with edges. The fractional relaxation of this covering problem has a dual with a natural interpretation of packing cuts in to edges. The above algorithms use covering and packing schemes, with complementary slackness driving primal and dual updates and bounding the performance guarantee. The description and implementation of these algorithms is relatively easy; the hardness is in the performance bounds. On the positive side, because the bounds based on LP duality, the algorithms compute, as a byproduct, a dual feasible solution which can be used as a lower-bound of optimality.

In practice, the problems that arose in the design of survivable CCS networks were of the same spirit as the Steiner network formalization, requesting subgraphs of low total cost that contained diverse paths between specified locations. The main discrepancy from the standard Steiner problems was that, in many cases, instead of single nodes, say $u$ or $v$, we had pairs of nodes, like $U$=$\{u_1, u_2\}$ and $V$=$\{v_1, v_2\}$; see also Figure 1.

One of the difficulties created by the introduction of two-element locations was that, unlike Steiner network problems, we were not able to express the path diversity requirements in terms of equivalent cut requirements. Thus, we first devised a set of cut requirements $f_{\text{suf}}$ that were sufficient, in the sense that any subgraph that was feasible for $f_{\text{suf}}$ contained a diverse set of paths. However, there were instances containing a diverse set of paths which were not feasible for $f_{\text{suf}}$. Thus, we devised a second set of cut requirements $f_{\text{nec}}$ that were necessary, in the sense that any instance containing a diverse set of paths included a subgraph feasible for $f_{\text{nec}}$. However, there were instances containing subgraphs feasible for $f_{\text{nec}}$ which did not include a set of diverse paths. We would thus proceed to first look for a low cost feasible solution of $f_{\text{suf}}$ and, if such a solution was found, we would search exhaustively for a suitable set of diverse paths. If there was no feasible solution to $f_{\text{suf}}$, we would subsequently find a low cost feasible solution of $f_{\text{nec}}$ and, around that solution, we would look for the required set of paths.

This gave rise to a further heuristic suggestion: Suppose that we raise all cut requirements slightly higher, say by one or two units wherever possible, and we use the approximation heuristic to prune the original network to a much smaller (however non-empty) subnetwork on which we can afford to perform exhaustive search of paths. This would certainly improve the quality of our solution, and

indeed, it was a useful technique is practice.

To measure the performance of our implementations we used (a)the solution of the LP-relaxation and (b)a lowerbound to the solution of the LP-relaxation given by a feasible solution to the dual of this LP; this latter bound is a byproduct of the algorithm. For the commercial product, our implementations were tested successfully against benchmark instances and our code forms the algorithmic core of the commercial product. Prior to our contribution, the only known method to developers was exhaustive enumeration of paths.

Our problem formalization assumes that all cost is due to links. However, nodes of networks, so-called switches, host increasingly more complicated software, and are thus becoming increasingly more expensive and less reliable. It is therefore important to develop algorithms which account for cost on vertices as well as edges.

CCS networks carry low bit-rate signaling information, separate from data, and thus fall in the general category of low bit-rate backbone networks. In practical terms, this means that the CCS bit-rate is much smaller than the bandwidth capacity of nodes and links. Thus, once a node or link is present, it can be assumed to carry arbitrarily many CCS paths. This assumption cannot be made for high bit-rate data networks like the widely used ATM and SONET[3]. We outline versions of network survivability with capacitated edges that appear to model survivability considerations for high bit-rate networks.

In practice, local exchange telecommunication networks have additional simplifying properties: they do not exceed a few hundred nodes in size (typically containing less than one hundred nodes), they are "nearly planar" and they are of very low degree. In addition, because most network elements operate independently and have very low failure probabilities, we do not anticipate any instance where survivability under more than four failures will be required, while, in most cases, we are required to survive the failure of no more than one or two network elements. Finally, demand patterns are typically restricted in natural ways: most connections evolve around local clusters, and non-local connectivity involves certain "centers" of each cluster. Thus, algorithms for classes of telecommunication networks with the simplifying features listed above are also of special interest.

The rest of this extended abstract is organized as follows: Section 2 provides an outline of theoretical Steiner network algorithms. Section 3 describes the survivability requirements of CCS networks, and the adaptation of Steiner algorithms that we implemented in our solutions. Section 4 discusses experiments and testing. Open survivability issues are in Section 5.

# 2 Steiner Network Algorithms

## Survivability under Link Failures

The standard form of the Steiner network problem that provides survivability under link failures is as follows. $G(V, E)$ is an undirected graph and $c$ is a non-negative cost function on the edges $E$. We further have non-negative integral path requirements $r_{uv}$, for each $u, v \in V$. We want a low cost subgraph $H \subseteq E$ such that there are $r_{uv}$ edge-disjoint paths from each $u$ to each $v$ using edges in $H$. Where $f$ is a cut requirement function defined on subsets of $V$ by

$$f(S) = \max_{u \in S, v \in \bar{S}} \{r_{uv}\}, \quad \forall S \subset V$$

we are dealing with the following equivalent formalization as an integer program (IP):

$$\begin{array}{lrcl} & \min \sum_e x_e c_e & & \\ \text{s.t.} & \sum_{e \in \delta(S)} x_e & \geq & f(S), \quad \forall S \subset V \\ & x_e & \in & \{0,1\}, \quad \forall e \in E \end{array}$$

where $\delta(S)$ is the edge-cutset of $S$ in $G$. The above program can be thought of as requesting a covering of cuts with edges.

Below we outline the heuristic due to [10] that finds a low cost solution to (IP). Initially the solution $H$ is the empty set; $H$ is gradually augmented in $k = \max_S f(S)$ phases. At the end of phase $k$ we ensure that $f$ is satisfied up to $k$ in the sense that, for each cut $S \subset V$ with cut requirement

[3] See [6] for an introduction to gigabit data-networks, Asynchronous Transfer Mode and Synchronous Optical Networks.

$f(S) \geq k$, $H$ contains at least $k$ edges extending from $S$ to $\bar{S}$ at the end of phase $k$. How is $H$ augmented during phase $k$? Since the original problem is thought of as covering cuts with edges, we may think of the dual problem as packing of cuts in to edges; indeed [10] use packing-covering criteria for the augmentation of $H$. In particular, for every cut $S \subset V$, they introduce a dual variable $y_S$; $y_S = 0$ at the beginning of each phase. The heuristic in [10] repeatedly identifies collections of disjoint subsets $S \subset V$ which are still unsatisfied for phase $k$. The $y_S$'s of these unsatisfied sets are increased uniformly until some edge $e \in E \setminus H$ becomes tight:

$$\sum_{S:e\in\delta(S)} y_S = c_e$$

At that point $e$ is added to $H$. When we can find no more unsatisfied subsets for phase $k$ we know that $H$ satisfies $f$ up to $k$.[4]

For the implementation we need to explain how, given a set of edges $H \subseteq E$, we can test if $H$ satisfies $f$ up to $k$ and, if not, how to identify a disjoint collection of unsatisfied sets. For this, it suffices to perform a $u$-$v$ maxflow computation for each pair $u$ and $v$ with $r_{uv} \geq k$. If this maxflow is strictly smaller than $k$, using standard Picard Queyranne methodology, we can identify the minimal $u$-$v$ mincut containing $u$, and the minimal $u$-$v$ mincut containing $v$, and add them to the collection of unsatisfied cuts.

It was shown in [10] that the above procedure indeed yields a collection of disjoint unsatisfied cuts, and moreover, that this collection contains all minimal unsatisfied cuts. This property was essential to show the performance bound of the algorithm. However, it is not essential for the completion of the algorithm: It is easy to verify that the above described algorithm finds a feasible integral solution to (IP) (assuming that one such solution exits) provided that at each step we identify some set of unsatisfied cuts, raise their dual variables uniformly, and add unused edges as they become tight. We shall use this observation is Section 3 as we introduce extensions of the Steiner network problem.

[4]The algorithm in [10] proceeds to simplify $H$, by considering the edges in an order reverse than the one in which they were included in $H$, and checking whether, after removing the edge, $f$ is still satisfied up to $k$. This step is crucial for the performance bound and we included it in our implementations. However, it is not crucial for outlining the skeleton of our work, we thus omitt further reference to it from this abstract.

## Survivability under Node Failures

The standard form of the Steiner network problem that provides survivability under node failures is as follows. $G(V, E)$ is an undirected graph, $c$ is a non-negative cost function on the edges $E$, and $r_{uv}$ is a non-negative integral path requirement. We want a low cost subgraph $H \subseteq E$ such that there are $r_{uv}$ node disjoint paths from each $u$ to each $v$ using edges in $H$.

An approximation algorithm for this problem was developed in [8] along the lines of [4]. The heuristic that we implemented was developed independently from [8], and along the lines of [10]. All these heuristics follow along the principles of the previous subsection, under suitable definitions of cut requirements for the vertex-disjoint path requirement. Further details can be found in [8].

## ALG 1: Steiner Network Implementation

For Steiner network problems there are algorithms other than the one outlined here that are both more efficient and yield better performance guarantees [1] [4]. We chose to implement the algorithm of [10] because it is the simplest (and the one that we were most familiar with) thus the easiest to adapt to the more general CCS network requirements. (As it is typical in the industry, we had to complete this project within two months from assignment.) We call this implementation ALG 1.

We find indicative of the power of the general primal-dual methodology the fact that, despite the minimal effort put in fine-tuning, our heuristics performed with efficiency and accuracy. Prior to our contribution, the only alternative of the developers was path enumeration, as exhaustive as possible. In an initial straightforward class of instances that we constructed to show the power of

our methods, our code returning the optimal solution within three seconds, while prototypes previous to ours were running for over half an hour.

# 3 Survivability in CCS and Low Bit-Rate Backbone Networks

The Common Channel Signaling network carries signaling separate from data; in fact, it is the most widely used signaling network. Like any typical signaling network, CCS carries protocols that are vital for the integrity of data networks. For example, it carries the SS7 protocol. It is therefore of paramount importance that the CCS network is itself highly protected against failures of network elements. For this, if a location $U$ needs to establish connectivity with a location $V$, this connection is established by multiple so-called "diverse" paths that have been precomputed and stored in the memory of $U$ and $V$, with each one of these paths carrying simultaniously the same signaling information.

The physical (nodes and links) layer of the CCS network is modeled as an undirected graph with a non-negative cost function assigned on its edges. Certain nodes are designated as singletons and they are called SEP's for Signal Ending Points. Other nodes are designated mated as STP pairs, for Signal Transfer Points. In practice, STP pairs denote supernodes of heavily switching traffic that serve as each other's backup. Each node is either a SEP, or it belongs to a unique STP pair. We refer to nodes and STP pairs as locations.

We say that a location $U$ is $n$-way edge connected to a location $V$ if we can identify $U$-$V$ paths such that, if $n$–1 links fail, at least one $U$-$V$ path survives. We say that a location $U$ is $n$-way vertex connected to a location $V$ if we can identify $U$-$V$ paths such that, if $n$–1 nodes fail (other than all nodes in $U$ or all nodes in $V$), at least one $U$-$V$ path survives. In practice, $n$-way connectivity was required between the nodes of an STP pair, for $n \in \{3, 4\}$. We were also asked for so-called "path diversity": in what follows we give an example of such a requirement.

Where $U = \{u\}$ is a SEP and $V = \{v_1, v_2\}$ is an STP pair, for 3-way edge connectivity, it can be verified that the following cut requirements are sufficient and necessary:

$$f(S) = \begin{cases} 3 & \text{if } u \in S \ \& \ v_1, v_2 \notin S \\ 0 & \text{o.w.} \end{cases}$$

while, for 3-way vertex connectivity, it can be verified that the following cut requirements are sufficient and necessary (under suitable definitition of vertex-cut, eg see [8]):

$$f(S) = \begin{cases} 3 & \text{if } u \in S \ \& \ v_1, v_2 \notin S \\ 2 & \text{if } v_1 \in S \ \& \ u, v_2 \notin S \\ 2 & \text{if } v_2 \in S \ \& \ u, v_1 \notin S \\ 0 & \text{o.w.} \end{cases}$$

and similarly we can identify sufficient and necessary conditions for 4-way connectivity.

Where $U = \{u_1, u_2\}$ and $V = \{v_1, v_2\}$ are both STP pairs, for 3-way edge connectivity, it can be verified that the following cut requirements are sufficient and necessary:

$$f(S) = \begin{cases} 3 & \text{if } u_1, u_2 \in S \ \& \ v_1, v_2 \notin S \\ 0 & \text{o.w.} \end{cases}$$

while, for 3-way vertex connectivity, it can be verified that the following cut requirements are sufficient and necessary (under suitable definitition of vertex-cut):

$$f(S) = \begin{cases} 3 & \text{if } u_1, u_2 \in S \ \& \ v_1, v_2 \notin S \\ 1 & \text{if } u_1, v_1 \in S \ \& \ u_2, v_2 \notin S \\ 1 & \text{if } v_1, u_2 \in S \ \& \ u_1, v_2 \notin S \\ 2 & \text{if } |S \cap (V \cup U)| = 1 \\ 0 & \text{o.w.} \end{cases}$$

and similarly we can identify sufficient and necessary conditions for 4-way connectivity.

Now for $U$ and $V$ STP pairs as above, consider the following path diversity requirement: we want one $u_1$-$v_1$ path, one $u_1$-$v_2$ path, one $u_2$-$v_1$ path and one $u_2$-$v_2$ path such that any two paths sharing the same endpoint are (say, edge) disjoint and, either $u_1$-$v_1$ is (edge) disjoint from $u_2$-$v_2$, or $u_1$-$v_2$ is (edge) disjoint from $u_2$-$v_1$. In Figure 2a we give an example of such paths.

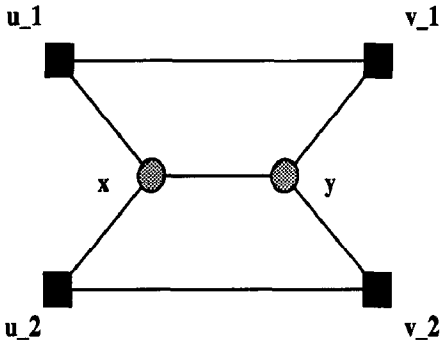


**Figure 2a.** An instance of STP pairs $U = \{u_1, u_2\}$ and $V = \{v_1, v_2\}$ and SEPs $x$ and $y$. We can identify paths $(u_1, v_1)$ and $(u_2, v_2)$ disjoint from both $(u_1, x, y, v_2)$ and $(u_2, x, y, v_1)$, and $(u_1, v_1)$ is disjoint from $(u_2, v_2)$. This instance satisfies $f_{\text{nec}}$ but not $f_{\text{suf}}$.

It can be verified that the following conditions are necessary for the existence of such paths:

$$f_{\text{nec}}(S) = \begin{cases} 3 & \text{if } u_1, u_2 \in S \ \& \ v_1, v_2 \notin S \\ 1 & \text{if } u_1, v_1 \in S \ \& \ u_2, v_2 \notin S \\ 1 & \text{if } v_1, u_2 \in S \ \& \ u_1, v_2 \notin S \\ 2 & \text{if } |S \cap (V \cup U)| = 1 \\ 0 & \text{o.w.} \end{cases}$$

however, Figure 2b suggests that these conditions are not sufficient.

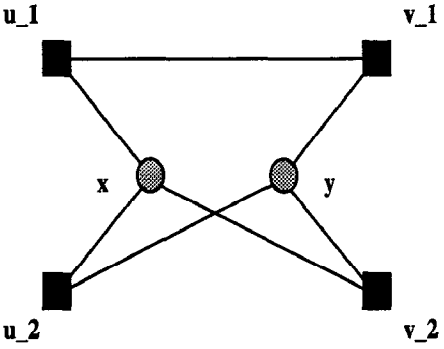


**Figure 2b.** An instance of STP pairs $U = \{u_1, u_2\}$ and $V = \{v_1, v_2\}$ and SEPs $x$ and $y$. We cannot identify paths $u_1$-$v_1$ and $u_2$-$v_2$ disjoint from $u_1$-$v_2$ and $u_2$-$v_1$ such that either $u_1$-$v_1$ is disjoint from $u_2$-$v_2$ or $u_1$-$v_2$ is disjoint from $u_2$-$v_1$. This instance satisfies $f_{\text{nec}}$ but not $f_{\text{suf}}$.

On the other hand, it can be also verified that the following conditions are sufficient for the existence of such paths:

$$f_{\text{suf}}(S) = \begin{cases} 3 & \text{if } u_1, u_2 \in S \ \& \ v_1, v_2 \notin S \\ 1 & \text{if } u_1, v_1 \in S \ \& \ u_2, v_2 \notin S \\ 1 & \text{if } v_1, u_2 \in S \ \& \ u_1, v_2 \notin S \\ 3 & \text{if } S \cap (V \cup U) = 1 \\ 0 & \text{o.w.} \end{cases}$$

however Figure 2a suggests that all these conditions are not necessary.

In practice, we would be specified which nodes were SEPs and which pairs of nodes were forming STP pairs, together with sets of requirements indicating 3 or 4-way connectivity between the nodes of an STP pair, 3 or 4-way connectivity between a SEP and an STP pair, 3 or 4-way connectivity between two STP pairs, or special "path diversity" requirements like the example presented above.

In our implementation we first translated the path requirements in terms of cut requirements, either exact, or sufficient and necessary. In a first run, we would try to find a low cost feasible solution with each cut satisfying the maximum of all sufficient requirements through that cut. To find such a solution, we would use the natural modification of the heuristic described in Section 2 (ie, finding disjoint sets of unsatisfied cuts during repeated iterations of a phase and adding unused edges as they became tight). If such a solution was found, we would look exhaustively for the required sets of diverse path. If such a solution was not found, we would try to find a low cost feasible solution with each cut satisfying the maximum of all necessary requirements through that cut, and proceed as previously. (The cases where both of the above approaches failed were treated in further compromising ways; details are omitted from this abstract). We call the above approach ALG 2.

## ALG 2: Prune and Search, and Fine-Tuning

The above procedures gave rise to a further heuristic suggestion: Suppose that we raise all cut requirements slightly higher, say by one or two units wherever possible, and we use the approximation heuristic to prune the original network to a much

smaller (however non-empty) subnetwork on which we can afford to perform exhaustive search of paths. We call this heuristic Prune and Search, and refer to in Section 4 as ALG 2. It was used for both CCS network survivability as well as Steiner networks. Finally, a last phase of fine-tuning was performed for improvements of the final set of paths by obvious shortcuts.

# 4 Experiments and Tests

Using a common set of core routines, we implemented Steiner network algorithms ALG 1 of Section 2 and ALG 2 (ie, with the Prune and Search heuristic) for problems where there is a single subset of nodes which have to be either $k$-edge connected or $k$-vertex connected, for some selected $k$. These were run on a set of random graphs. The random graphs were chosen by first picking random points in the unit square for nodes, and then selecting each possible edge $e$ independently with probability $1-\alpha\cdot\text{length}(e)$. The parameter $\alpha$ was chosen so that the density of the graph would be low, but the graph would be still well connected.

We compared these runs with the linear programming lower bounds, the lower bound obtained by the dual solution constructed by the algorithm and a solution by enumeration of paths.

We also implemented survivable CCS network algorithms ALG 1 and ALG 2. These were run on a different set of random graphs built by choosing random points on the unit square, creating the Delaunay triangulation and then adding edges between nodes which had a path of length two between them.

All tests were performed on a SPARC-10; sample tables of performances follow. Data on benchmark test cases are Bellcore proprietary and cannot be disclosed.

# 5 Questions: The Cost of Nodes and High Bit-Rate Networks

Our problem formalization assumes that all cost is due to links. However, nodes of telecommunications networks, so-called switches, become increasingly more sophisticated as they host complex routing software and delicate electronics and optics. This makes them also more expensive and less reliable. It is therefore important to develop algorithms which account for cost on vertices as well as edges. A place to start could be the Steiner network problem, with maximum requirements as low as 2 or 3, and with unit costs on vertices and zero cost on edges. We anticipate that we will be required to address such problems in the near future (and we have actually seen such problems in gigabit networks that we discuss below).

CCS networks transmit signaling information at rates of a few kbits/sec, while the nodes and links carrying these messages operate at minimum bandwidth of tens of Mbits/sec. Thus, for all practical purposes, we may assume that nodes and links, once in place, are of infinite capacity as far as the CCS network is concerned. However, high bit-rate networks such as SONET which runs at Gbits/sec, or ATM which runs at hundreds of Mbits/sec, perceive their carriers (nodes and links) as having bandwidth a few times larger than their own, typically 4, 8, 16, or 32. For this setup the Steiner network formalization of survivability is inadequate. We outline the following lines of modification to capture survivability in high bit-rate networks:

- For given edge capacity $w$, solve the Steiner network problem where, if $L$ paths use an edge, this edge is charged $\lceil \frac{L}{w} \rceil$ times. This version would become important in optical networks: SONET and optical multiwavelength.
- For given edge capacity $w$, and given flow requirements $r_{uv}$ (not necessarily integral), find a low cost subgraph such that, if $r_{uv}-1$ elements fail, a flow of at least 1 from $u$ to $v$ survives. Moreover, if an edge supports a flow of value $L$, then its cost is charged $\lceil \frac{L}{w} \rceil$ times. Such versions would become important in ATM packet switched networks.

| n | terms | k | LP bound | time | ALG 1 | time | ALG 2 | time |
|---|---|---|---|---|---|---|---|---|
| 10 | 3 | 2 | 1.209880 | 0.662 | 1.209880 | 0.012 | 1.004116 | 0.014 |
| 40 | 3 | 2 | 1.122670 | 1.175 | 1.138476 | 0.061 | 1.083979 | 0.090 |
| 40 | 3 | 3 | 1.148745 | 1.319 | 1.186349 | 0.099 | 1.179641 | 0.357 |
| 40 | 7 | 2 | 1.215072 | 2.111 | 1.351092 | 0.178 | 1.267500 | 5.088 |
| 40 | 7 | 3 | 1.282923 | 1.989 | 1.392669 | 0.292 | * | * |
| 160 | 3 | 2 | * | * | 1.170883 | 1.008 | 1.160413 | 1.918 |
| 160 | 3 | 3 | * | * | 1.191722 | 1.578 | 1.226544 | 2.803 |

**Table 1**: Steiner network problems, averages of 10 trials each. Here $n$ is the number of nodes overall, *terms* is the number of nodes that must be connected, and $k$ is the number of edge-disjoint paths required between each pair. The LP bound and the solutions from ALG 1 and ALG 2 have been scaled by the dual bound from ALG 1. The run times are in seconds on a SPARC-10. The scaled bound, scaled solutions, and run times are averaged over ten runs. Missing LP bounds took too much time to compute. Missing values for ALG 2 were because of the graphs lacking the required additional connectivity.

| n | ALG 1 | time | ALG 2 | time | enum | time |
|---|---|---|---|---|---|---|
| 10 | 1.325295 | 0.065 | 1.325295 | 0.103 | 1.315267 | 0.519 |
| 20 | 1.243418 | 0.184 | 1.242435 | 0.263 | 1.238595 | 2.668 |
| 40 | 1.406135 | 0.721 | 1.392866 | 1.055 | 1.345479 | 12.627 |
| 80 | 1.380433 | 3.117 | 1.367483 | 4.657 | * | * |
| 160 | 1.523245 | 13.505 | 1.460573 | 21.644 | * | * |

**Table 2**: CCS network problems, averages of 10 trials each. Four paths are required between two pairs of STPs. Here $n$ is the number of nodes. The solutions from ALG 1 and ALG 2, and the solution by partial enumeration have been scaled by the dual bound from ALG 1. The run times are in seconds on a SPARC-10. The scaled bound, scaled solutions, and run times are averaged over ten runs. Missing solutions for enumeration took too much time to compute. For about 15% of these problems, the subgraph generated by ALG 1 did not have the required four paths. In these cases, the cost of the subgraph is substituted here for the solution cost.

## References

[1] H. Gabow, M. Goemans, and D. Williamson, "An Efficient Approximation Algorithm for Survivable Network Design Problems", *Proc. IPCO 93.*

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

[3] M. Goemans and D. Williamson, "A General Approximation Technique for Constrained Forest Problems", *Proc. SODA 92.*

[4] M. Goemans, A. Goldberg, S. Plotkin, D. Shmoys, E.Tardos, and D. Williamson, "Improved Approximation Algorithms for Network Design Problems", *Proc. SODA 94.*

[5] D. Minoli, *Telecommunications Technology Handbook*, Artech House, Boston-London, 1991.

[6] G. Partridge, *Gigabit Networking*, Addison Wesley, 1994.

[7] R. Ravi and P. Klein, "When Cycles Collapse: A General Approximation Technique for Constrained Two-Connectivity Problems", *Proc. IPCO 93.*

[8] R. Ravi and D. Williamson, "An Approximation Algorithm for Minimum-Cost Vertex-Connectivity Problems", *Proc. SODA 95.*

[9] D. Williamson and M. Goemans, "Computational Experience with an Approximation Algorithm on Large-Scale Euclidean Matching Problem", *Proc. SODA 94.*

[10] D. Williamson, M. Goemans, M. Mihail, and V. Vazirani, "A Primal-Dual Approximation Algorithm for Generalized Steiner Network Problems", *Proc. STOC 93.*