Telecommunication Link Restoration Planning with Multiple Facility Types

Article *in* Annals of Operations Research · September 2001

DOI: 10.1023/A:1014509708610 · Source: DBLP

eBusiness Research Center Working Paper
5-2001

Telecommunication Link Restoration Planning with Multiple Facility Types

Anantaram Balakrishnan
Thomas L. Magnanti
Joel S. Sokol
Yi Wang

December 2000

eBusiness Research Center
401 Business Administration Building
University Park, PA 16802
Phone: 814.861.7575
Fax: 814.863.0413
Web: www.ebrc.psu.edu

A joint venture of Penn State's Smeal College of Business Administration
and the School of Information Sciences and Technology

Telecommunication Link Restoration Planning with Multiple Facility Types

Anantaram Balakrishnan
Penn State University, University Park, PA

Thomas L. Magnanti
Massachusetts Institute of Technology, Cambridge, MA

Joel S. Sokol
Georgia Institute of Technology, Atlanta, GA

Yi Wang
i2 Technologies, CA

December 2000

Abstract

To ensure uninterrupted service, telecommunication networks contain excess (spare) capacity for rerouting traffic in the event of a link failure. We study the NP-hard capacity planning problem of economically installing spare capacity on a network with steady-state working flows to permit link restoration. We present a spare capacity planning model that incorporates multiple facility types, and develop optimization-based heuristic solution methods based on solving a linear programming relaxation and minimum cost network flow subproblems. We establish bounds on the performance of the algorithms, and discuss problem instances for which the bounds are tight to within a constant factor. In tests on three real-world problems and several randomly-generated problems containing up to 50 nodes and 150 edges, the heuristics provide good solutions (often within 0.5% of optimality) to problems with single facility type, in equivalent or less time than methods from the literature. For multi-facility problems, the gap between our heuristic solution values and the linear programming bounds are larger. For small graphs, we show that the optimal linear programming value does not provide a tight bound on the optimal integer value, suggesting that the heuristic solutions are closer to optimality than implied by the gaps.

-

1. Introduction

By providing vastly increased network bandwidth, new telecommunication technologies offer the capability to transmit large amounts of data on just a few fibers. However, since each fiber carries so much traffic, the failure of even one of these fibers can seriously compromise the level of service provided to customers. To protect against service interruptions due to transmission failures, network planners install more capacity than is necessary on each link of a network, reserving the excess (spare) capacity for rerouting network traffic in the event of a link failure. Digital Cross-connect Switches (DCSs) installed at each node automatically reroute traffic along pre-specified paths when a network link fails.

Current systems use two types of rerouting schemes for network restoration: link restoration and path restoration. Link restoration reroutes all of the disrupted traffic from one endpoint of the failed link to the other, regardless of the origins and destinations of individual units of this traffic. Path restoration, on the other hand, separately considers each unit of traffic on the failed link, and reroutes this traffic from its origin to its destination. Although path restoration often requires less spare capacity than link restoration (Veerasamy, Venkatesan, and Shah [30]), it also requires more complex network hardware to reroute traffic to respond to any failure. In particular, the system must maintain information concerning the sources and destinations of the traffic flowing on each link, and have the ability to direct flow on alternate routes for each affected origin-destination pair. In practice, telecommunication networks employ both link and path restoration schemes ([8], [9]).

In this paper, we focus on spare capacity planning for link restoration. We study the problem of economically installing enough spare capacity on a network with given (steady-state) working traffic flows, so that the network can reroute traffic when any single link fails. We adopt a common single-link failure assumption used in both path and link restoration planning models, namely that at most one link will be out of service at any given time. Planners consider this assumption to be reasonable since link failure rates are typically much smaller than repair rates ([8], [9])[1]. The spare facilities needed to carry restoration traffic, comprising transmission lines and terminating equipment, are modular, i.e., each facility has a fixed capacity, but we can install many facilities on an edge. We consider *multiple facility types*, each with a specified capacity and fixed cost; these facilities can be installed in any desired combination to achieve the required

[1] Meshkovskiy and Rokotyan [24] discuss the contingency when a widespread disaster destroys many links of a network. In this case, the focus shifts from selecting alternate routes for traffic to prioritizing the links to repair in order to reconnect the network as quickly as possible.

spare capacity on a link. Generally, capacities are expressed in terms of Optical Carrier level *b* (OC-*b*) bandwidths. Common OC-*b* levels include OC-1, OC-3, OC-12, OC-48 and so on, with OC-1 representing a capacity of 51.84 Mbps. If we measure traffic in OC-1 units, then the capacity of an OC-*b* facility is *b* units; thus, for instance, an OC-12 has four times the transmission capacity of an OC-3. Consistent with actual cost structures, we assume that facility costs exhibit economies of scale, i.e., for $b > 1$, an OC-*b* facility costs less than *b* times an OC-1 facility. The literature on spare capacity planning for link restoration has not previously considered the possibility of installing multiple facility types. Most papers assume that capacity is available only in OC-1 units; a few permit a single facility type with capacity $b > 1$.

Venables [31] has shown that the spare capacity planning problem for link restoration is NP-hard, even with only a single facility type. Thus, practitioners are interested in efficient heuristics that provide near-optimal solutions. This paper develops optimization-based heuristic methods to generate cost-effective spare capacity solutions for link restoration. In Section 2, we formulate the multi-facility spare capacity planning problem as an integer program, and briefly review the related literature. In Section 3, we present heuristics based on linear programming and network flow optimization, establish worst-case bounds on the performance of our algorithms, and demonstrate using examples that the bounds are tight to within a constant factor. We also describe a local improvement method that attempts to reduce the capacity installed in any given spare capacity solution. Section 4 discusses our computational results. We tested our heuristics on three real-world networks provided by industry as well as 155 randomly-generated networks that permit testing the methods' sensitivity to a wider range of problem structures; these problems range in size from 10 nodes and 15 edges to 50 nodes and 150 edges. For each network, we considered both the single-facility and the multi-facility versions of the problem. Our best heuristic results are within 0.5% of the linear programming lower bound for most cases, with a maximum gap of 3.5% over all problem instances. For multi-facility test problems, although the heuristic costs were sometimes significantly higher than the optimal linear programming value, we show that these costs are within 10% of the true (integer) optimal value, suggesting that the linear programming bound is weak for these problems. To compare our heuristics' performance to results reported in the literature (by Venables, Grover, and MacGregor [32]), we also considered single-facility test problems with working flows of 0 or 1. Our heuristics provide solutions that are no more expensive than previous results from the literature, but require significantly less computational time. Section 5 summarizes our results and suggests some future research directions.

2. Multi-facility Spare Capacity Planning Problem

2.1. Problem definition

Let $G:(N,E)$ be the undirected graph representing the underlying telecommunications network. For each edge $e \in E$, we are given the steady-state *working flow* d^e, expressed in multiples of OC-1 traffic, that the edge will carry when the network is fully functional. Let $E_f \subseteq E$ denote the subset of $m_f \triangleq |E_f| \le m$ *vulnerable* edges that have positive working flows and can fail. For notation, let $n = |N|$ and $m = |E|$ denote the number of nodes and edges in G, let $O(e)$ and $D(e)$ denote the endpoints of edge $e \in E_f$, and let (i, j) refer to the any (undirected) edge of the network connecting node i and node j.

Suppose K *facility types* are available to provide spare capacity. We index these facility types from 1 to K in increasing order of capacity. For $k = 1, 2, \ldots, K$, let b^k be the *capacity* of facility type k expressed in multiples of OC-1 capacity, and let c_{ij}^k denote the (nonnegative) *cost* of installing one unit of facility type k on edge (i, j). Spare capacity is modular, i.e., we must install integer amounts of each facility type, but can combine different facility types to provide the required capacity on each edge. Clearly, the cost of different facility types must increase monotonically with capacity. That is, for all edges $(i, j) \in E$, if $k < l$ (and so $b^k \le b^l$ since we index facility types in order of increasing capacity), then $c_{ij}^k \le c_{ij}^l$. Otherwise, if $c_{ij}^k > c_{ij}^l$ for some $k < l$, then optimal solution will not use facility type k. We might also expect costs to increase at a decreasing rate with capacity, reflecting economies of scale, i.e., $c_{ij}^k / b_{ij}^k \ge c_{ij}^l / b_{ij}^l$ for all $k < l$. We refer to this cost structure, with larger facilities having lower per-unit costs, as *concave* facility costs.

Let b_{ij} denote the excess capacity that is currently available on edge (i, j) for use as spare capacity, after netting out working flows. Some switching technologies require distinct working and spare capacities, i.e., they do not permit utilizing the residual capacity in the working links to route restoration traffic (see Balakrishnan, Magnanti, Sokol, and Wang [3]). In this *distinct* capacity special case, $b_{ij} = 0$ for all edges (i, j).

The *multi-facility capacity planning* (*MCP*) problem for link restoration requires finding the least-cost installation of spare capacity so that, for each vulnerable edge $e \in E_f$, we can reroute all the working traffic on this link if it fails. As noted earlier, this model (like others in the literature) effectively assumes that the repair rate for failed links is much greater than the failure rate of working links, so that no more than one link will be in a state of failure at any given time. For feasibility, we require the underlying graph G to be doubly-connected with respect to the

vulnerable edges. That is, for every edge $e \in E_f$, the network must contain at least one arc-disjoint path excluding edge *e* joining nodes *O*(*e*) and *D*(*e*). Otherwise, the failure of some edge would disconnect the graph, making restoration impossible. Our model permits routing restoration flows on multiple paths, rather than requiring all of the flows from a failed edge *e* to be routed on a single path from *O*(*e*) to *D*(*e*). Such flow splitting is feasible in practice, and reduces the total spare capacity needed in the network. Finally, we assume that flows and capacities are all integer-valued. (More generally, we permit these parameters to be rational numbers. We can then convert these numbers to integers by suitably scaling all parameters.)

2.2. Model formulation

For each edge $(i, j) \in E$ and facility type $k = 1, \ldots, K$, let y_{ij}^k be the nonnegative, integer amount of spare capacity of type *k* installed on edge (i, j), and let f_{ij}^e and f_{ji}^e be the flows from node *i* to node *j* and vice versa when a vulnerable edge *e* has failed. By definition, if edge *e* connects nodes *i* and *j*, then $f_{ij}^e = f_{ji}^e = 0$, i.e., when an edge *e* fails its own spare capacity is unusable.

We can formulate the multi-facility capacity planning problem as follows:

[*MCP*]
$$Z^* = \min \sum_{k \in K} \sum_{(i,j) \in E} c_{ij}^k y_{ij}^k \tag{2.1}$$

subject to:

$$\sum_{j:(i,j) \in E} (f_{ij}^e - f_{ji}^e) = \begin{cases} d^e & \text{if } i = O(e) \\ -d^e & \text{if } i = D(e) \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } i \in N, e \in E_f \; i \in N, e \in E, \tag{2.2}$$

$$f_{ij}^e + f_{ij}^e \le \sum_{k \in K} b^k y_{ij}^k + b_{ij} \qquad \text{for all } (i,j), e \in E_f \; (i,j), e \in E, \tag{2.3}$$

$$f_{ij}^e = f_{ji}^e = 0 \qquad \text{for all } (i,j) = e \in E_f \; (i,j) = e \in E, \tag{2.4}$$

$$f_{ij}^e \ge 0 \qquad \text{for all } (i,j), e \in E_f \; (i,j), e \in E, \tag{2.5}$$

$$y_{ij}^k \ge 0 \qquad \text{for all } (i,j) \in E, k = 1, \ldots, K, \text{ and} \tag{2.6}$$

$$y_{ij}^k \text{ integer} \qquad \text{for all } (i,j) \in E, k = 1, \ldots, K. \tag{2.7}$$

Formulation [*MCP*] has $m_f n + m + 2m_f \; mn + m + m^2$ constraints (excluding nonnegativity and integrality constraints), $2m \bullet m_f \; m^2$ continuous variables, and $m|K|$ integer variables.

Constraints (2.2) are flow balance constraints at each node *i* for the restoration flows when edge *e* fails. Since edges are bidirectional, we arbitrarily select node *O*(*e*) as the origin and node *D*(*e*) as the destination for the restoration flow when edge *e* fails. The current working flow d^e essentially represents the *demand* for a commodity that must be routed from origin *O*(*e*) to destination *D*(*e*). Constraints (2.3) ensure that we install sufficient spare capacity on each edge to accommodate the maximum restoration flow assigned to it in any of the restoration scenarios (i.e., over all edge failures), and constraints (2.4) prevent using the spare capacity on a failed edge to restore its own working flow.

Observe that we did not restrict the flow variables f_{ij}^e to be integral. Given the feasible capacity choices **y**, the constraints (2.2) to (2.5) to find feasible restoration flows decompose by edge *e* into |*E*| network flow subproblems that are each feasible. Since demands and capacities are integer-valued, these network subproblems have integer solutions. Consequently, we have the following result.

Flow Integrality Property: The MCP problem has an optimal solution with integral flows.

We refer to the special case of the MCP problem with just one facility type (|*K*| = 1) as the *single-facility capacity planning* (*SCP*) problem. For this special case, Balakrishnan et al. [3] have shown that if the single facility type OC-*b* has capacity $b > 1$ and if $b_{ij} = 0$ for all edges $(i, j) \in E$ (e.g., spare capacity must be distinct from working capacity), we can convert any given SCP problem instance into an equivalent (in terms of integer solutions) SCP instance with unit capacities. The transformation entails replacing the original demand parameters d^e with $\lceil d^e / b \rceil$ for all edges $e \in E_f$ $e \in E$. In other words, we express the demand in terms of OC-*b* units by scaling and rounding up the original working flows; the rounding up operation effectively increases the demand that spare capacities must accommodate. Intuitively, since capacities can be installed only in bundles of *b* units at a time, any feasible capacity plan for the original problem must also be adequate for the rounded-up demands. This transformation is advantageous because it strengthens the linear programming relaxation of the problem (without eliminating any integer solutions to the original problem), thus vastly improving the performance of LP-based solution methods (see Balakrishnan et al. [3]).

2.3. Literature review

Telecommunication carriers have begun deploying restoration technologies only relatively recently. So, the literature on spare capacity planning—for both path and link restoration—is also

recent, but continues to grow. As we noted earlier, none of the previous papers have considered the availability of multiple facility types in the context of link restoration.

Sakauchi, Nichimura, and Hasegawa [26] proposed an LP-based algorithm to solve the single-facility restoration planning problem, assuming link restoration. Their Iterated Cutsets Heuristic (ICH) iteratively solves the linear programming relaxation of a cutset formulation, generating violated cutset inequalities at each iteration to strengthen the linear program. Because the cutset formulation has exponentially many constraints, the algorithm might require excessive computation time. Further, because the constraints do not define the convex hull of the integer problem, the final solution might be fractional. The ICH method rounds up the fractional values to obtain a feasible solution. Herzberg [12] showed how to exploit the existence of two simple subgraphs, the triangle and the triangular pyramid, to enhance the linear program solved by the ICH method.

Grover, Bilodeau, and Venables [11] proposed an alternate heuristic called the Spare Link Placement (SLP) algorithm. The SLP method is a two-stage heuristic procedure. First, it creates a feasible solution by iteratively adding one, two, or an entire path of unit spare capacities. Then, it examines the spare capacity on each link to see if it can reduce the overall amount of spare capacity by removing capacity from one edge and adding, if necessary, a lower amount of capacity on another edge. The SLP method selects the restoration paths from a list of the k shortest origin-to-destination paths. Other researchers (Herzberg, Bye, and Utano [13] and Chujo, Komine, Miyazaki, Ogura, and Soejima [5]) have included "hop limit" constraints that impose an upper bound on the number of links on which rerouted traffic can travel. Our model does not impose such constraints on the available restoration paths. Lee and Chun [16] attempted to solve the restoration problem using an artificial neural network. They obtained good results for a 4-node test network.

The ICH and SLP heuristics, as well as the other previously cited research, assume (as we do) that restoration traffic can be split into multiple paths, rather than requiring all traffic to follow a single path. Veerasamy, Venkatesan, and Shah [29] found that allowing traffic splitting reduces the total spare capacity needed by approximately 35%, with greater reduction for larger, denser networks. Both the ICH and SLP heuristics also assume that the spare facilities are distinct from the working network. Most of the literature implicitly defines the spare capacity problem as the installation of OC-1 lines. Grover, Bilodeau, and Venables [11] consider a single facility type with capacity b (i.e., OC-b lines), and solve this problem using the SLP algorithm.

Magnanti and Wang [22] and Bienstock and Muratore [4] studied the polyhedral characteristics of the SCP problem and a related problem for which only a predetermined fraction of demand must be rerouted. They developed several classes of facets and described the convex hull of special cases. When incorporated in a cutting plane algorithm, these inequalities accelerated considerably the solution time of the integer program (Balakrishnan et al. [3]). The literature also includes examples of special cases and variants of the SCP problem. Minoux [25] used subgradient optimization to solve the case when spare capacity can be installed in any amount, rather than only discrete amounts. Stoer and Dahl [27] and Dahl and Stoer [6] studied an integrated model that selects the set of edges to be included in the network, in addition to the working and spare capacities on each edge. Lisser, Sarkissian, and Vial [20] also studied the integrated model, first determining base traffic and spare capacity using line restoration, and then computing the necessary spare capacity to protect the base traffic using path restoration. Lisser, Sarkissian, and Vial [19] used an analytic-center-based cutting plane algorithm to solve the linear programming formulation of a similar path restoration problem. Vachani and Kubat [28] studied the problem of minimizing spare capacity for line restoration in a special type of network, a bidirectional SONET ring. Kennington and Whitler [15] developed a decomposition algorithm for a more general SONET mesh architecture.

Groetschel, Monma, and Stoer [10] provided a comprehensive review of computational and polyhedral results in survivable network design. Two models from the network design literature—the capacitated network loading problem (Magnanti, Mirchandani, and Vachani [21]) and the multi-level network design problem (Balakrishnan, Magnanti, and Mirchandani [2])—consider multiple facility types. However, both of these models differ from our restoration planning problem in two ways: (i) the network design models consider simultaneous flows whereas restoration entails non-simultaneous flows, and (ii) the restoration setting imposes the special constraint (2.4) prohibiting flow on the failed edge.

3. Optimization-based Solution Methods

In this section, we first describe two heuristic methods for the MCP problem—an LP round-up method that constructs a feasible solution from the optimal solution to the MCP model's linear programming (LP) relaxation, and a Capacity Layering heuristic based on minimum cost network flow subproblems. We study the worst-case performance for both methods, and develop examples to show that the worst-case bound is achieved to within a constant factor. We conclude the section by describing a local improvement method that can often reduce the cost of any given heuristic solution such as those generated by the LP round-up or Capacity Layering procedures.

3.1. LP round-up heuristic

The LP round-up heuristic constructs a feasible MCP solution by selecting a minimum cost combination of facilities to install on each edge to provide at least as much capacity on the edge as suggested by the optimal solution to the linear programming (LP) relaxation of formulation [*MCP*]. The procedure consists of the following three steps:

1. Solve the LP relaxation of formulation [*MCP*], obtained by relaxing the integrality constraints (2.7) on the *y*-variables. Let $(\hat{\mathbf{y}}, \hat{\mathbf{f}})$ denote the optimal LP solution, and for each edge $(i, j) \in E$, let $T_{ij} = \sum_{k=1}^{K} b^k \hat{y}_{ij}^k$ denote the total capacity (in OC-1 units), possibly fractional, that the LP solution installs on that edge.
2. For each edge $(i, j) \in E$, round up the LP capacity T_{ij} to obtain the total *required* capacity $R_{ij} = \lceil T_{ij} \rceil$ on that edge.
3. For each edge $(i, j) \in E$, determine the least cost combination of available facility types to attain the required spare capacity of R_{ij} units or higher.

Step 3 requires solving, for each edge $(i, j) \in E$, the following *facility loading* subproblem.

$$[FL(i,j)] \qquad \text{Minimize} \quad \sum_{k=1}^{K} c_{ij}^k x_{ij}^k \tag{3.1}$$

subject to

$$\sum_{k=1}^{K} b^k x_{ij}^k \geq R_{ij} \qquad \text{for all } (i,j) \in E, \text{ and} \tag{3.2}$$

$$x_{ij}^k = 0 \text{ or } 1 \qquad \text{for all } k = 1, \ldots, K. \tag{3.3}$$

If the number of facility types is small or the values of T_{ij} (and so R_{ij}) are low, we can solve this subproblem by enumeration. Alternatively, the following dynamic programming recursion finds the optimal solution. Let $V_{ij}(t)$ denote the minimum cost combination of facilities to achieve a total spare capacity of *t* units or higher on arc (*i*, *j*).

Facility Loading Algorithm

Step 0: Set $V_{ij}(t) = 0$ for all $t \leq 0$.

Step 1: For $t = 1, 2, \ldots, R_{ij}$

Set $V_{ij}(t) = \min_{k=1,\ldots,K} \left\{ \min_{s=0,1,\ldots,b^k-1} \left[V_{ij}(t + s - b^k) + c_{ij}^k \right] \right\}$ (3.4)

next *t*;

$V_{ij}(R_{ij})$ is the optimal value of the subproblem [*FL*(*i*, *j*)].

The recursion (3.4) specifies that the minimum cost to achieve a spare capacity of *t* or higher equals the minimum cost, among all facility types *k*, of adding a type *k* facility to a system that has total capacity between $t - b^k$ and $t - 1$. Note that, if at an intermediate iteration of Step 1 the installed capacity, say, *w* exceeds the required capacity *t*, then we can skip the next (*w*-*t*-1) iterations (since the capacity *w* is feasible for all required capacity values from *t*+1 to *w*). We can obtain the optimal values of x_{ij}^k for problem [*FL*(*i*, *j*)] by keeping track of the facility choice that minimizes the right-hand side of equation (3.4) at each stage.

Observe that the optimal solution to the capacity installation subproblem [*FL*(*i*, *j*)] might provide some "free" spare capacity, i.e., $\sum_{k=1}^{K} b^k x_{ij}^k$ might exceed R_{ij}. As an example, suppose two capacity facility types are available, OC-1 lines and OC-3 facilities, at costs of 1 and 1.5 each. If the LP solution installs half a unit of an OC-3 facility on an edge (*i*, *j*), i.e., if $\hat{y}_{ij}^1 = 0$ and $\hat{y}_{ij}^2 = \frac{1}{2}$, then $R_{ij} = \lceil \frac{1}{2} 3 \rceil = 2$, and so we must install at least two units of capacity on this edge. We can do so by installing two OC-1 lines facilities at a total cost of 2; a cheaper alternative is to install one OC-3 facility at a cost of 1.5. This latter solution provides a capacity of three units, exceeding the required two units. Thus, the optimal capacity installation solution creates one unit of "free" spare capacity. Our local improvement heuristic (discussed in Section 3.3) attempts to exploit this excess capacity to reduce capacity on other edges.

Finally, note that, if the facility costs are the same for all edges, then we need to solve the facility loading subproblem only once overall—for the maximum capacity requirement over all edges. The method generates the optimal solutions for all smaller requirement values at intermediate steps.

3.1.1 Worst-case analysis

To characterize the cost performance of the LP round-up heuristic, we compare the heuristic cost to the optimal value of the LP relaxation. For this analysis, we assume that the facility costs are the same for all edges, and so we omit the subscript (i.e., the edge index) from these cost parameters.

For each node $i \in N$, let $D_{\max}(i) = \max_{e \in E_f : O(e) \text{ or } D(e)=i} d^e$ be the maximum working flow among all edges incident to node *i*. To reroute this working flow, the linear programming solution must install a total of at least $D_{\max}(i)$ units of spare capacity on the edges incident to node *i*. Thus, the

LP solution must install at least $\sum_{i \in N} D_{\max}(i)/2$ units of total capacity on edges of the network. Let $\hat{k} = \text{argmin}_{k=1,\ldots,K} c^k/b^k$ denote the index of facility type with the lowest per-unit cost; if the cost structure is concave, $\hat{k} = K$. Then, the optimal linear programming value Z^{LP} must be at least $(c^{\hat{k}}/b^{\hat{k}})\sum_{i \in N} D_{\max}(i)/2$.

Let us now develop an upper bound on the cost of the LP round-up heuristic solution. For any required capacity *r*, let *V*(*r*) denote the minimum cost to achieve a capacity of *r* or more on any edge (i.e., *V*(*r*) is the optimal value of the facility loading problem with required capacity *r*). The rounding up operation in step 2 of the LP round-up heuristic increases the capacity by at most one unit for each edge, i.e., $R_{ij} \le T_{ij} + 1$. Consequently, if $\hat{d} = \max_{e \in E_f} d^e$ denotes the maximum working flow over all the edges in the network, the LP round-up heuristic incurs a cost of no more than

$$\overline{C}_{ij} = (c^{\hat{k}}/b^{\hat{k}})(T_{ij} + 1) + \max_{r \le \hat{d}} \left(V(r) - r(c^{\hat{k}}/b^{\hat{k}}) \right) \tag{3.5}$$

to install the required capacity T_{ij} on any edge (*i*, *j*). The first term of this upper bound represents the lowest possible cost to install $T_{ij} + 1$ units of capacity on edge (*i*, *j*), and the second term is the maximum incremental cost due to integrality restrictions. Note that this second term, $\max_{r \le \hat{d}} \left(V(r) - r(c^{\hat{k}}/b^{\hat{k}}) \right)$, depends only on the facility costs, capacities, and maximum working flow, independent of the spare capacity installation decisions; so, we can compute it a priori for any given problem instance.

Summing the upper bound over all edges of the networks gives an upper bound on the total cost Z^{LRU} of the LP round-up solution. That is,

$$\begin{aligned} Z^{LRU} &\le \sum_{(i,j) \in E} (c^{\hat{k}}/b^{\hat{k}})(T_{ij} + 1) + \max_{r \le \hat{d}} \left(V(r) - r(c^{\hat{k}}/b^{\hat{k}}) \right) \\ &\le (c^{\hat{k}}/b^{\hat{k}}) \sum_i D_{\max}(i)/2 + m \max_{r \le \hat{d}} \left(V(r) - r(c^{\hat{k}}/b^{\hat{k}}) \right). \end{aligned}$$

Therefore, if r^{LRU} denotes the ratio of the LP round-up solution's cost to the optimal value of the LP relaxation, we have:

$$r^{LRU} = \frac{Z^{LRU}}{Z^{LP}} \le 1 + \frac{m}{\sum_i D_{\max}(i)/2} + \frac{m \max_{r \le \hat{d}} \left(V(r) - r(c^{\hat{k}}/b^{\hat{k}}) \right)}{(c^{\hat{k}}/b^{\hat{k}}) \sum_i D_{\max}(i)/2}. \tag{3.6}$$

Since the optimal value Z^* of the integer program [*MCP*] is greater than or equal to Z^{LP}, the right-hand side expression in equation (3.6) also applies to the ratio Z^{LRU}/Z^*.

For the *SCP* problem (single-facility case), assuming we have scaled the demands so that the facility capacity b equals one, the cost $V(r)$ exactly equals $r(c^1/b^1)$ for all integer values r, and so inequality (3.6) implies that:

$$Z^{LRU} \le Z^{LP}\left(1+\frac{2m}{\sum_i D_{\max}(i)}\right). \qquad (3.7)$$

We next discuss an example for which this bound on the performance of the LP round-up heuristic is tight up to a constant factor.

3.1.2 "Worst"-case example

Consider an SCP problem instance defined on a complete graph containing n nodes. Assume that the demands have been scaled so that the facility has unit capacity, i.e., $b = 1$. The cost per unit of spare capacity is the same for all edges, and so minimizing the total spare capacity installed also minimizes total cost. Suppose all edges of the network are vulnerable, and the working flow on each edge of the network is one unit, i.e., $d^e = 1$ for all $e \in E_f = E$.

Consider a fractional solution that installs $1/(n-2)$ units of spare capacity on each edge. This solution is feasible for the LP relaxation because, if any edge e fails, we can reroute $1/(n-2)$ units of this edge's working flow on each of the n-2 paths $O(e) \to l \to D(e)$, for all $l \in N, l \ne O(e), D(e)$. We next show that this solution is also optimal for the LP relaxation.

For every node i of the network and every edge (i, j) incident to this node, the LP solution must install at least a total of one unit of spare capacity on all of the remaining edges (other than edge (i, j)) incident to node i in order to protect against the failure of edge (i, j). That is, the LP solution must satisfy:

$$\sum_{\substack{j' \in N \\ j' \ne i, j}} y_{ij'} \ge 1 \qquad \text{for all } j \in N, j \ne i \text{ and } i \in N. \qquad (3.8)$$

Our solution $y_{ij} = 1/(n-2)$ for all $(i, j) \in E$ satisfies conditions (3.8) as an equality for all $j \in N, j \ne i$ and $i \in N$. Therefore, this solution installs the minimum possible total capacity, and must therefore be optimal for the LP relaxation.

The cost of this LP optimal solution is $m/(n-2) = n(n-1)/2(n-2)$. Starting with this LP solution, the LP round-up heuristic rounds each of the $1/(n-2)$ spare capacities to value 1. Thus, the cost of the LP round-up solution is $m = n(n-1)/2$, and so the ratio of actual cost of the LP-

round solution to actual LP value for this example is $(n-2)$ compared to the theoretical bound, from expression (3.6), of $1+2m/n = 1+(n-1) = n$. This example shows that the theoretical bound is nearly tight.

Now consider any feasible integer solution to formulation [MCP]. The spare capacity network must be connected since, if no path connects nodes *i* and *j*, then we cannot restore edge (*i*, *j*). Also, inequality (3.8) applies to the integer program as well. Summing these inequalities over all nodes $j \in N, j \neq i$ shows that

$$\sum_{\substack{j' \in N \\ j' \neq i}} (n-1) y_{ij'} \geq n-2 \qquad \text{for all } i \in N. \tag{3.9}$$

Dividing both sides of inequality (3.9) by $(n-1)$ and rounding up the right-hand side to two (since the *y* variables must be integral) shows that any feasible integer solution must install a total capacity of at least two units incident to each node $i \in N$. Consequently, the cost of the integer solution must be at least $2n/2 = n$. Since a Hamilton cycle has this cost and is feasible, it is an integer optimal solution, at a cost of *n*. Thus, the LP round-up cost divided by the optimal integer program value is $(n-1)/2$, showing that the theoretical bound of *n* is at least of the correct order of magnitude.

3.2 Capacity Layering heuristic

The *Capacity Layering* (*CL*) heuristic adds spare capacity in "layers" by iteratively considering increasing levels of working flows (or demands) to be restored. Let $\hat{d}$ denote the maximum working flow over all edges, and suppose we consider *L* different *demand levels* $1 \leq h_1 < h_2 < \ldots < h_L = \hat{d}$ in the range $[1, \hat{d}]$. Let *H* denote the set of chosen demand levels. For instance, *H* might consist of all integer values from 1 to $\hat{d}$, or all the distinct values of working flows in the original network. For $l = 1, 2, \ldots, L$, let G^l denote the original network but with working flows reduced to $\min(d^e, h_l)$ for every edge $e \in E$. The CL heuristic consists of *L* major iterations, successively adding capacity at each iteration to permit restoration of the increasing demand levels. At the end of iteration *l*, the method provides a feasible spare capacity installation plan for network G^l. The next iteration adds spare capacity to this solution to generate a feasible plan for network G^{l+1}. To add spare capacity, the method sequentially considers each edge *e* whose working flow equals or exceeds h_{l+1}, and augments the capacity of the existing spare network in order to restore a flow of h_{l+1} on edge *e*. To augment capacity cost effectively, the method solves a minimum cost network flow (abbreviated henceforth as min-cost flow) subproblem that incorporates the capacity that has already been installed and the

incremental cost of adding capacity on every edge of the network. We first describe the generic method, and later discuss our specific implementation.

In the following description of the CL heuristic, R_{ij} denotes the spare capacity currently available on edge (i, j), and E^l represents the subset of edges of the original network whose working flows exceeds the $(l\text{-}1)^{\text{th}}$ demand level h_{l-1}. Recall from Section 3.1 that $V_{ij}(r)$ is the minimum cost of providing a total spare capacity of r units or higher on edge (i, j), obtained by solving the facility loading subproblem [$FL(i, j)$].

Capacity Layering (CL) heuristic

Step 0: Initialization

Set $R_{ij} = b_{ij}$ for all $(i, j) \in E$, and $h_0 = 0$

Step 1: Iterative capacity addition

For $l = 1, \ldots, L$

Set $E^l = \{e \in E_f : d^e > h_{l-1}\}$

For each edge $e \in E^l$

Set $q_l^e = \min(d^e, h_l)$

Step 1a: Solve the minimum cost flow restoration subproblem

- Construct a network $G' : (N, E')$ defined over the original node set N but containing two parallel edges $(i, j)_1$ and $(i, j)_2$ corresponding to every original edge $(i, j) \in E \setminus \{e\}$. Edge $(i, j)_1$ has capacity equal to the current spare capacity R_{ij} on edge (i, j) and zero cost. Edge $(i, j)_2$ has capacity equal to $(q_l^e - h_{l-1})$ and per unit flow cost of $\left(V(R_{ij} + q_l^e - h_{l-1}) - V(R_{ij})\right) \Big/ \left(q_l^e - h_{l-1}\right)$.
- Solve a minimum cost network flow problem to route $q_l^e = \min(d^e, h_l)$ units of working flow on G' from $O(e)$ to $D(e)$. For each edge $(i, j) \in E \setminus \{e\}$, let f_{ij} denote the total flow on the parallel edges $(i, j)_1$ and $(i, j)_2$ in the min-cost flow solution.
- For each edge $(i, j) \in E \setminus \{e\}$:
  - solve the facility loading problem [$FLP(i, j)$] assuming a requirement of f_{ij} units of spare capacity on that edge;
  - set R_{ij} equal to the total capacity installed by the optimal solution to this problem.

  Next edge (i, j);

Next edge e;

Next l;

Starting with the existing spare capacities, the CL method first installs enough spare capacity to accommodate link restoration for the network G^1 in which the working flow of all edges is

restricted to be less than or equal to h_1. It then progressively augments this capacity to build feasible solutions for successively higher demand levels. At each major iteration *l*, the method sequentially considers each edge *e* that has flow greater than the previous demand level h_{l-1}, and uses the min-cost flow model to decide the paths needed to restore $q_l^e = \min(d^e, h_l)$ units of flow if this edge fails. The min-cost flow model considers the capacity that has already been installed (either available originally or installed in previous iterations); it assigns this capacity to the parallel edge $(i, j)_1$ at zero cost. The network flow solution can install additional capacity by routing flow on the second parallel edge $(i, j)_2$. Since, at iteration *l*, the current capacity is adequate to restore h_{l-1} units of flow on every edge $e \in E^l$, the network flow solution will never install more than $(q_l^e - h_{l-1})$ of additional capacity on any edge. So, we can impose an upper bound of $(q_l^e - h_{l-1})$ on the flow on edge $(i, j)_2$. The method sets the unit cost of flow on this second edge equal to the average incremental cost of increasing the capacity of edge (i, j) from its current level of R_{ij} to $(R_{ij} + q_l^e - h_{l-1})$. Note that this cost of new spare capacity is not exact since the network flow solution might install less than $(q_l^e - h_{l-1})$ units of additional capacity. If the cost structure is concave, the per-unit cost underestimates the true cost.

Different choices of the set of demand levels *H* and different sequencing rules for selecting the edges to be restored (in the augmentation step) within each major iteration give rise to various versions of the CL heuristic. First, let us consider some possible choices for the set *H*. If we set $H = \{\hat{d}\}$, then the algorithm performs only one major iteration (since all edges have working flows between 0 and $\hat{d}$). For each edge in turn, the method installs adequate capacity to restore the full working flow d^e if that edge fails. We refer to this version of the CL heuristic, with *L* = 1, as the *max step* version. At the other extreme, suppose *H* contains all integers from 1 to $\hat{d}$, and so $L = \hat{d}$. In this case, each "layer" covers one more unit of demand. This *unit step* version entails $\hat{d}$ major iterations. An intermediate choice sets *H* equal to the set of all distinct values of working flows among all the edges of the network. We refer to this choice of demand levels as the *demand step* version. Note that, for this version, $L \le \min(m_f, \hat{d})$. Furthermore, since the heuristic reduces the number of edges in E^l by at least one in each major iteration, and since $L \le m_f$, the demand step version of the CL method solves at most $m_f(m_f - 1)/2$ min-cost flow problems.

The max step version requires the least computational effort since it requires only one major iteration, and solves m_f min-cost flow problems in this iteration. However, this method tends to install large amounts of spare capacity on some edges, and in our preliminary computational tests produced solutions that generally had a much higher cost than the demand step version. The unit

step version requires much more solution time than the demand step method since it must solve up to $m_f \hat{d}$ min-cost flow problems. The solutions cost approximately the same as those from the demand step version, so the extra running time is not worthwhile. Within each major iteration, we need to specify the order for considering edges for solving the min-cost flow restoration problems. We tested several approaches, including sorting the edges in increasing order of working flow, decreasing order of working flow, random order, and lexicographic order (in terms of the node numbers of the endpoints of the edges). We found that, for all of our test cases, considering the edges in increasing order of working flow gave the best results. The computational results for the CL heuristic that we report in Section 4 pertain to the demand step version with edges sequenced in increasing order of working flow.

3.2.1. Worst-case analysis

For each failed edge $e \in E_f$, let $SP(e)$ denote the length of the shortest restoration path (excluding edge e) from node $O(e)$ to node $D(e)$ using $V_{ij}(d^e - b_{ij})$ as the length of each edge (i, j). Recall that $V_{ij}(d^e - b_{ij})$ is the minimum total cost to install a spare capacity of $(d^e - b_{ij})$ or more on edge (i, j). The value $SP(e)$ is the "full" cost of installing a feasible restoration path if edge e fails, without taking advantage of spare capacities needed to restore flows when other edges fails. Therefore, the sum of these shortest path costs over all edges, $\sum_{e \in E_f} SP(e)$, is an *upper bound* on the total cost Z^{CL} of the CL heuristic solution. Note also that $SP(e)$ is the optimal value of a "relaxed" MCP problem in which only edge e can fail. Consequently, the value of $SP(e)$ for any edge $e \in E_f$ is a lower bound on the optimal value of the original MCP problem. In particular, if $e' = \text{argmax}_{e \in E_f} SP(e)$, then $SP(e')$ is a *lower bound* on the optimal value, and hence on the cost of the CL heuristic solution. That is, if Z^* denotes the optimal value of the MCP problem, the cost of the CL heuristic solution is bounded above and below as follows:

$$\max_{e \in E_f} SP(e) \le Z^* \le Z^{CL} \le \sum_{e \in E_f} SP(e). \tag{3.10}$$

Since $\sum_{e \in E_f} SP(e) \big/ m_f \le m_f \max_{e \in E} SP(e) \big/ m_f = \max_{e \in E_f} SP(e)$, the bounds in (3.10) imply that the CL heuristic solution costs no more than m_f times the optimal value of the MCP problem.

3.2.2. "Worst"-case example

As with the LP round-up heuristic, we will create a class of problem instances where the worst-case bound on the CL heuristic is tight up to a constant factor.

In the network shown Figure 1, in the main cycle of length M ($M \ge 4$ and even), every second edge (edges $(i, i+1)$ for all odd i) is vulnerable and has a working flow of one. All other working

flows are zero. For every vulnerable edge (i, i+1), the network has a possible restoration side path of length M-1-(i+1)/2. As before, we assume that all edges have the same facility costs. Since working flows are all one or less, we can consider unit cost for each facility, effectively minimizing the total spare capacity installed.

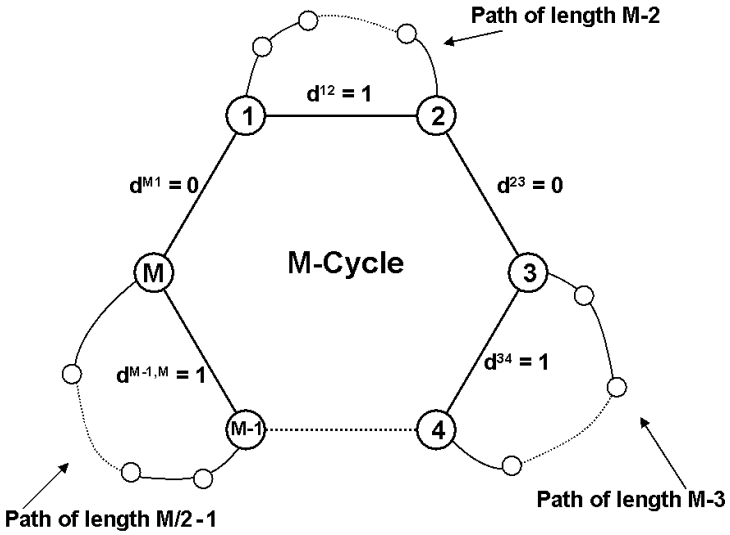

Figure 1. Worst-case example for the Capacity Layering heuristic.

The optimal solution to this problem installs one unit of spare capacity all around the M-cycle, at a total cost of M. Since the maximum flow and hence L is one, the CL heuristic performs only one major iteration. If we sort the vulnerable edges in lexicographic order (since they all have the same working flow), then for each vulnerable edge e, the min-cost flow solution selects the side path from $O(e)$ to $D(e)$ of length M-1-(i+1)/2. Thus, all of the attached paths will have spare capacity installed, for a total cost of $\sum_{i=1}^{M/2}(M-(i+1)) = 3M^2/8 - M/4$.

Since the network contains $m_f = M/2$ $\hat{m} = M/2$ vulnerable edges, the lower bound on the heuristic solution is M/2. The ratio of the heuristic solution to the optimal solution is approximately 3M/8, and so the worst-case bound of $m_f = M/2$ is tight to within a constant factor.

3.3. Capacity Reduction method for local improvement

Given a feasible MCP solution, the *Capacity Reduction* (*CR*) method attempts to locally improve this solution by removing units of spare capacity that are "redundant," i.e., whose removal retains sufficient restoration capacity for all edges. Suppose the current solution installs a total spare capacity of S_{ij} on edge $(i, j) \in E$. Since spare capacity is modular, S_{ij} might include some "free"

spare capacity (see Section 2.1) that is not needed for restoration but arises because the total capacity of the least cost combination of facilities chosen by the facility loading problem *FL*(*i, j*) exceeds the required capacity. Let p_{ij} be the minimum amount of capacity that must be removed from edge (*i, j*) to reduce the cost of spare capacity on this edge, i.e., $p_{ij} = \min_{p=1,\ldots,S_{ij}} \{V_{ij}(S_{ij} - p) - V_{ij}(S_{ij}) > 0\}$. For example, suppose two facility types, OC-1 and OC-3 costing 1 and 1.5 are available, and suppose $S_{ij} = 3$. The least cost facility for edge (*i, j*) is, therefore, an OC-3 facility costing 1.5. If we were to decrease the requirement to 2, the cost would still be 1.5, whereas decreasing the requirement to 1 reduces the cost to 1. Thus $p_{ij} = 2$ in this example. For each edge $(i, j) \in E$, the CR method first determines the value of capacity reduction p_{ij} needed to reduce the cost of spare capacity on this edge. The method then verifies, by solving min-cost flow problems for each vulnerable edge, whether the MCP solution maintains feasibility when we remove p_{ij} units from the currently installed capacity of S_{ij}. If the new solution is feasible, then we reduce the capacity on edge (*i, j*) to $S_{ij} - p_{ij}$. Otherwise, this edge retains its existing capacity S_{ij}, and the method attempts to reduce the capacity on another edge.

A formal description of this CR method follows.

Capacity Reduction (CR) procedure

Step 0: Set S_{ij} equal to the total capacity installed on each edge $(i, j) \in E$ in the given starting solution.

Step 1: For each edge $(i, j) \in E$

Step 1a:

- Calculate p_{ij}, the minimum capacity reduction needed to reduce the cost on edge (*i, j*)
- Remove p_{ij} units of spare capacity from edge (*i, j*) (while retaining the current spare capacities for all other edges) to obtain the "reduced" capacity network G"; set the cost of flow on each edge equal to zero.
- For all vulnerable edges $e \in E_f$

  Step 1b: Solve the min-cost flow problem on network *G*" with edge *e* removed to determine if the reduced capacity network has adequate spare capacity to restore d^e units of flow from *O*(*e*) to *D*(*e*).

  Next edge *e*;
- If the reduced capacity network can fully restore the flows for all vulnerable edges (i.e., if none of the min-cost flow problems in Step 1b are infeasible), then update $S_{ij} \leftarrow S_{ij} - p_{ij}$ and repeat Step 1a.

Next edge (*i, j*);

Interestingly, the CR heuristic will generally run faster for multi-facility (MCP) problems than for the single-facility special case. MCP solutions might have some "free" spare capacity, so that the

algorithm needs to solve fewer min-cost flow problems. SCP solutions, on the other hand, do not contain free spare capacity (assuming the parameters have been scaled so that $b = 1$), and so the algorithm may need to solve more min-cost flow problems.

In general, if T is the amount of spare capacity removed by the CR procedure, the method must solve Tm_f min-cost flow problems (less if "free" spare capacity can be removed). For the Capacity Layering heuristic, T could be large. For the LP round-up heuristic, however, at most m units of spare capacity can be removed; and so the CR procedure solves at most $m \bullet m_f$ min-cost flow problems to reduce capacity for a starting solution obtained using the LP round-up method.

The CR heuristic might not necessarily improve either the CL or LP round-up solutions or the bounds on the heuristics. Consider, for example, the CL and LP round-up solutions (for either the multi-facility or single-facility versions of the problem) to the problem instance shown in Figure 1. In both heuristic solutions, every unit of spare capacity is essential to the feasibility of the solution, so the CR method does not provide any improvement.

4. Computational Results

To test the performance of our heuristic procedures, we applied them to three real-world networks and 155 randomly-generated networks ranging in size from $n = 10$ nodes to $n = 50$ nodes, $m = 3n/2$ edges to $m = 3n$ edges, and average working flows from 5 to 175 units. To compare our methods' performance with previous results reported by Venables et al. [32], we also tested problems with $m = 2n$ edges, each having a working flow of 0 or 1 unit.

4.1 Test problems

A telecommunications company (see Balakrishnan et al. [3]) provided, two of the real-world test problems. The smaller instance contained 10 nodes, 14 edges, and working flows between 13 and 302, while the larger instance had 53 nodes, 79 edges, and working flows between 0 and 94. We obtained a third test problem from the literature ([11], [24], [33]), with 11 nodes, 23 edges, and working flows between 16 and 81. To test the methods over a wider range of problem parameters, we also generated numerous random test problems using the procedure described in Appendix A. Given user-specified values for the desired number of nodes n, number of edges m, and range of working flows $[p, q]$, the problem generator first constructs a doubly-connected random graph of appropriate dimensions. All edges of the graph are vulnerable. For each edge, the method sets the working flow equal to a random integer selected from the interval $[p, q]$. We refer to problem instances with working flows in the range $[p, q]$ as $[p, q]$-problems. We

considered 10 different network sizes, ranging from 10 nodes and 15 edges to 50 nodes and 150 edges. Five networks are "sparse" with $m = 3n/2$ edges and five networks are "dense" with $m = 3n$ edges[2]. For each size, we considered three different ranges of working flow levels: [0,10], [0,100], and [0,350]. We assume that the current network does not contain any free capacity that can be used for restoration (i.e., $b_{ij} = 0$ for all edges (i, j)). For each of the 30 size-flow range combinations, we generated five random networks; all of the statistics we report are averaged over these five instances. For each random network instance, we considered both the single-facility version (with $b = 1$) and multi-facility versions with two different cost structures, giving a total of 450 test problem instances. To benchmark our methods with those reported by Venables et al. [32], we also considered five network sizes with single facility and [0,1]-flows, generating five random networks for each size.

All of our test problems assume, for convenience, that facility costs are the same for each edge. For convenience, we also set the unit cost c^1 of the first (lowest capacity) facility type equal to one. Most previous research has also assumed edge-invariant costs. This assumption holds in practice if the fixed cost of transmission lines and terminating equipment dominate any cost components that depend on the edge lengths. Our solution methods, of course, apply to the more general situation when facility costs vary by edge. For single-facility problems with $b = 1$, when costs do not vary by edge, minimizing total cost is equivalent to minimizing the total spare capacity installed. For multi-facility (MCP) problems, we must define values for the facility capacity b^k and cost c^k. We considered three facility types with relative capacities of $b = 1$, 3, and 12 lines (representing, for example, OC-1, OC-3, and OC-12 facilities). By pricing the hardware, we found, as anticipated, that actual facility costs reflect economies of scale. In particular, the relative costs of the different facility types (with OC-3 as facility type 2, and OC-12 as facility type 3) approximately satisfy the following relationships: $c^2 = c^1 + (b^2 - b^1)(0.6)$ and $c^3 = c^2 + (b^3 - b^2)(0.6)^2$. Thus, costs exhibit quadratic economies of scale, with a scale factor $d = 0.6$. In addition to this real-world scale factor, we solved for two extreme scale factors: $d = 0.1$ representing strong economies of scale and $d = 1.0$ for situations with no economies of scale. Note that, when $d = 1$, the MCP problem reduces to the SCP special case (since, without economies of scale, installing OC-1 facilities is as cost-effective as higher capacity facilities). We set the unit cost c^1 of the first (lowest capacity) facility type equal to one.

[2] These choices for edge densities were motivated by two observations: the real-world problems have approximately m = 3*n*/2 edges, and the maximum density of a planar graph is on the order of $m = 3n$

To solve the linear programming relaxation in the LP round-up heuristic, we used CPLEX 3.0, running on a Sun Sparc workstation. To obtain the optimal value as a benchmark for comparison, we also attempted to solve the integer program [*MCP*] optimally for SCP problems and small MCP problems, using the CPLEX solver. To solve the minimum cost network flow subproblems for the Capacity Layering (CL) heuristic and the Capacity Reduction (CR) procedure, we used the network flow algorithm developed by Lee [17], implemented on a DEC 5000/133 workstation. For all test problems, we applied the LP round-up heuristic and the CL heuristic, and then applied the local improvement CR procedure to both of these solutions.

4.2. Results for SCP problems

Table 1 shows the results for 30 randomly generated SCP problems and the three real-world problems. The table reports the percentage difference between the heuristic solution values (for the LP round-up and CL heuristics; before and after improvement using the CR procedure) and the optimal LP value. Also, included are statistics showing the percentage gap between the theoretical upper bound (3.6) on the LP round-up cost and the optimal LP value. The last column of the table shows the percentage gap between the best (lowest) heuristic cost and the optimal value of the integer program (IP). All statistics for the randomly generated networks represent averages over five problem instances for each network size-flow range combination.

For the randomly generated networks, the LP round-up heuristic was more effective than the CL heuristic. Although the worst-case performance guarantees (theoretical upper bound) on the cost of the LP round-up solutions ranged as high as 70% from optimality, the solutions themselves were within 4% of the LP value, with the exception of the dense ($m = 3n$) [0,10]-problems. The CR method, applied to the LP round-up solutions, further reduced the gaps to within 0.5% of the linear programming bound. For the dense [0,10]-problems, the heuristic to LP gaps were within 5%. On the whole, the CR method reduced the gaps by a factor of five or more in almost every instance. In fact, by solving the IPs to optimality, we found that the improved solution generated by the CR method, starting with the LP round-up solutions, was optimal for every random SCP test problem. Thus, for all these test problem instances, the observed heuristic-to-LP gaps represent exactly the duality gaps between the optimal IP and LP values. The CL heuristic was less effective for the random instances of the SCP problem, with the gaps for the improved CL solutions ranging from 3% to 63%. Interestingly, the performance of the CL heuristic appears to be very sensitive to the density of the networks. For sparse networks, the improved CL solutions were no more than 5% from optimal.

In general, the LP round-up method was fast, often running in a few seconds and always less than one minute for small problems. The method required approximately 1 hour for 40 node, 120 edge problems, and around 2.5 hours for our largest problems. Because the CR local improvement method must solve at most 2*m* min-cost flow problems when applied to the LP round-up method, it ran in a few seconds for small problems, and required no more than 15 minutes for the largest problems.

When applied to the three real-world problems, the LP round-up heuristic followed by the CR local improvement method found solutions that were within 0.2% of optimal, running in under 10 seconds for the smallest problem and requiring just over two minutes for the largest problem. Again, for these problems, the CL heuristic gave solutions that were further from optimality (up to 5.65%).

4.3. Results for MCP problems

As we noted earlier, our MCP problems incorporate three facility types with capacities $b^1 = 1,\ b^2 = 3,$ and $b^3 = 12$, and costs $c^1 = 1$ and $c^k = c^{k-1} + (b^k - b^{k-1})(d)^{k-1}$ for k = 2, 3. We considered two values for the scale factor—d = 0.6 and d = 0.1—reflecting realistic (moderate) and strong economies of scale. Tables 2 and 3 present the computational results for these two cost scenarios for 30 randomly generated MCP test problems, and the three real-world problems. Again, the statistics for the random problems are averages over five problem instances for each network size-flow range combination. Unlike the SCP problems, finding optimal solutions to MCP problems was very difficult; CPLEX could find the optimal integer solutions for only small MCP problem instances. A blank in the last column of Tables 2 and 3 indicates that we were not able to optimally solve the integer program for any problem instance corresponding to that network size and flow range. For the remaining cases, the notation "$\leq x$ %" indicates that we could not solve the integer program to optimality for some instances, but obtained, during the branch-and-bound procedure, a better lower bound than the LP value; in this situation, x % represents the average of the exact heuristic-to-IP gaps and the heuristic-to-lower bound gaps.

The gaps between the costs of our heuristic solutions and the optimal LP values were higher than those for SCP problems. With a cost scale factor of d = 0.6 (Table 2), the LP roundup heuristic performed on average no worse than, and often better than, the CL heuristic. For the d = 0.1 case (Table 3), the LP roundup results were superior for [0,350] and [0,100] problems. However, for [0,10] problems, the CL method with improvement gave better results. Although heuristic-to-LP gaps are relatively large for MCP problems, the heuristic costs are within 3.5% of the optimal IP

value for the small problems that we could solve, or at least bound. These results suggest that the heuristic solutions are near-optimal, but the linear programming relaxation is very weak. We might expect the LP bounds to be poor, especially for smaller values of d, because the optimal linear programming solution will use fractional amounts of the most cost-effective (but high capacity) facility type whereas the true solution might require integer amounts of the smaller, less cost-effective facilities.

Note that the optimal solution to the LP relaxation is the same for all values of d since this solution always uses the facility type with the lowest per-unit cost. Therefore, the running times for the LP round-up heuristic were the same for the MCP problems as for the SCP problems. On the other hand, because MCP solutions typically contain free spare capacities, the min-cost flow heuristics (CL and CR procedures) were up to twice as fast for MCP problems compared to SCP problems.

For the [0,10]-problems, the CL heuristic provided better solutions and was also faster than the LP round-up method. In these cases, the running times (including the CR procedure) ranged from under 10 seconds (for the 10-node, 15-edge problems) to about 40 minutes (for the 50-node, 150-edge problems).

For the MCP versions of the real-world problems (see Tables 2 and 3), the LP round-up method performed better (as was the case for random problems with similar parameters). The cost of the best heuristic solution for each problem ranged from 1.1% from LP optimal to 12.8% from LP optimal, although we suspect that the gaps relative to the optimal IP value are much smaller.

4.4. Comparisons to ICH and SLP results

Venables, Grover, and MacGregor [32] tested both the ICH heuristic and three implementations of the SLP heuristic on some [0,1]-problems (problems in which each working flow is either 0 or 1) with unit facility capacities. Note that, for [0,1]-problems, only the SCP case is meaningful since the optimal solution will not install more than one unit of spare capacity on any edge. For networks with $m = 2n$ edges, Venables et al. reported both the results and execution time for their computational tests. To compare our heuristics' performance with their methods, we tested our heuristics on [0,1] problems with $m = 2n$ edges, for random networks containing 10, 20, 30, 40, and 50 nodes generated using the same methodology as described in Venables et al.

For the 10 and 20 node problems, we were able to find exact IP solutions using CPLEX. For 30, 40, and 50 node problems, we obtained lower bounds on the optimal IP value by using connectivity arguments of the following nature: Let E_f be the set of edges of E with positive working flow (of one). Suppose the graph $G^*:(N,E_f)$ consists of T connected components, and for $t = 1, \ldots, T$, let N_t denote the number of nodes in component t. By construction of the components, the edges containing spare capacity in each component t must also define a connected graph over N_t. Therefore, each component must contain at least $(N_t - 1)$ edges with spare capacity (of one unit). Summing $(N_t - 1)$ over all components t gives a lower bound on the total spare capacity that any feasible solution must install.

Table 4 contains results for the [0,1]-test problems. As the comparison of heuristic-to-LP gaps with the heuristic-to-IP gaps in Table 4 reveals, the LP relaxation is quite weak for [0,1]-problems. The CL heuristic ran much faster than the LP round-up heuristic, and when followed by the CR procedure, generated solutions that were at least as good (see Table 4). Although these solutions can cost up to 12% more than the optimal solution, in four of the five test cases, including the 50-node, 100-edge problem, the CL heuristic with local improvement produced a solution that installed at most one unit of spare capacity more than the optimal plan. These solutions are no more costly, and in most cases less costly, than solutions for similar problems obtained using the ICH and SLP heuristics [32].

The running times of our best solutions ranged from under 3 seconds (for the 10-node problem) to just over two minutes (for the 50-node problem). By comparison, the SLP heuristics took over 2 hours and 45 minutes (for the fastest SLP algorithm; the slowest took more than 1.5 days) to solve a 50-node, 100-edge problem, and the ICH heuristic was unable to find a solution after 4 days of calculation [32]. (Although these algorithms were implemented on different platforms, we expect the running times to be of the same order of magnitude as our implementation.)

5. Summary

As our extensive computational tests demonstrate, the LP roundup method with local improvement via capacity reduction provides good results for SCP problems with moderate or high working flows. For all our test networks, the heuristic solutions were actually optimal (although the heuristic to LP gaps were positive, but small). The LP round-up method takes advantage of the possibility of sharing spare capacities, and has a tighter worst-case bound than the Capacity Layering (CLO) heuristic. For problems with low working flows, including the

[0,1]-problems, the CL method with local improvement appears to both provide better quality of solutions and require less computational time. The method generates solutions that are often within one unit of spare capacity of the optimal IP solution, and requires less than two minutes of computational time—a marked improvement over previous solution methods from the literature.

For the MCP problem, the LP roundup method with improvement works best except in the extreme cases of low working flows and a very high economy of scale (a low d-value). For these latter cases, the CL method gives much better results. Our MCP results, especially for situations with smaller working flows, show that the heuristic-to-LP gaps are quite large. We speculate, based upon our attempts to solve the integer program optimally that most of this gap is due to a weak LP bound, rather than poor heuristics. In fact, we tried solving the integer program for one 10-node, 30-edge, [0,10]-problem instance for which the LP bound was approximately 40% of the improved CL solution's cost. After running five days, CPLEX had still not solved the IP, but had established a lower bound that was only 7.5% below our heuristic cost; moreover, the intermediate feasible solutions generated by the branch-and-bound procedure did not improve upon the CL solution.

The algorithms and results reported in this paper suggest two areas for further research: (i) exploring enhancements to the heuristic methodology to improve both their speed and accuracy, and (ii) identifying new classes of valid inequalities that strengthen the LP formulation for MCP problems so that we can obtain more accurate lower bounds on these problems. As our computational results demonstrate, the heuristic-to-LP gaps grossly overestimate the actual heuristic-to-IP gaps for cases when we could find the optimal integer solution or identify better lower bounds. Strengthening the LP relaxation can not only help better assess the quality of the heuristic solutions, but also vastly improve the performance (both computational time and bounds) of linear-programming based solution methods such as branch-and-bound or cutting plane algorithms. As we discussed in Section 2.3, several authors have studied the polyhedral structure of the SCP problem, and developed new classes of valid inequalities that have proved to be computationally effective. Similar work needs to be conducted for the MCP problem.

Table 1. Computational results for SCP problems

Problem Size			% from optimal LP value					% from optimal IP value
Nodes	Edges	Flow Range	Upper Bound on LP round-up	LP round-up cost	LP round-up + CR cost	CL cost	CL + CR cost	Best heuristic cost
Random problems								
10	15	[0,350]	1.16%	0.06%	0.02%	7.04%	5.16%	0.00%
20	30	[0,350]	1.17%	0.06%	0.01%	6.77%	4.92%	0.00%
30	45	[0,350]	1.14%	0.09%	0.01%	5.70%	4.06%	0.00%
40	60	[0,350]	1.21%	0.07%	0.01%	4.49%	3.21%	0.00%
50	75	[0,350]	1.20%	0.09%	0.01%	6.06%	4.19%	0.00%
10	15	[0,100]	4.06%	0.08%	0.03%	6.99%	5.12%	0.00%
20	30	[0,100]	4.07%	0.43%	0.05%	6.77%	4.84%	0.00%
30	45	[0,100]	4.00%	0.38%	0.05%	5.94%	4.15%	0.00%
40	60	[0,100]	4.21%	0.30%	0.05%	4.35%	2.89%	0.00%
50	75	[0,100]	4.18%	0.31%	0.04%	6.14%	4.25%	0.00%
10	15	[0,10]	39.66%	0.83%	0.28%	6.95%	4.95%	0.00%
20	30	[0,10]	39.91%	2.41%	0.31%	6.35%	4.07%	0.00%
30	45	[0,10]	39.10%	3.44%	0.33%	6.78%	4.36%	0.00%
40	60	[0,10]	40.96%	2.95%	0.40%	4.75%	3.27%	0.00%
50	75	[0,10]	40.79%	2.73%	0.43%	6.01%	3.74%	0.00%
10	30	[0,350]	2.03%	0.55%	0.10%	69.75%	63.03%	0.00%
20	60	[0,350]	2.00%	0.44%	0.07%	52.57%	46.61%	0.00%
30	90	[0,350]	2.05%	0.49%	0.09%	43.85%	36.22%	0.00%
40	120	[0,350]	2.05%	0.54%	0.09%	49.42%	43.90%	0.00%
50	150	[0,350]	2.03%	0.51%	0.08%	36.98%	31.07%	0.00%
10	30	[0,100]	7.10%	1.66%	0.25%	68.88%	61.52%	0.00%
20	60	[0,100]	6.99%	1.50%	0.33%	52.12%	46.51%	0.00%
30	90	[0,100]	7.17%	1.95%	0.34%	43.21%	34.95%	0.00%
40	120	[0,100]	7.17%	1.75%	0.30%	49.39%	43.71%	0.00%
50	150	[0,100]	7.08%	1.80%	0.26%	40.67%	35.57%	0.00%
10	30	[0,10]	69.01%	20.67%	4.43%	68.87%	61.01%	0.00%
20	60	[0,10]	67.84%	15.41%	3.27%	53.71%	47.47%	0.00%
30	90	[0,10]	69.70%	18.64%	3.35%	43.96%	34.94%	0.00%
40	120	[0,10]	69.52%	17.06%	2.89%	49.83%	41.79%	0.00%
50	150	[0,10]	68.98%	16.89%	3.26%	43.90%	35.89%	0.00%
Real-world problems								
10	14	[13,302]	1.42%	0.47%	0.30%	6.96%	5.65%	0.00%
11	23	[16,81]	5.98%	1.82%	1.17%	7.30%	5.12%	0.16%
53	79	[0,94]	7.71%	3.73%	3.28%	4.77%	3.37%	0.09%

Table 2. Computational results for MCP problems with cost scale factor $d = 0.6$

Problem Size			% from optimal LP value					% from optimal IP value
Nodes	Edges	Flow Range	Upper Bound on LP round-up	LP round-up cost	LP round-up + CR cost	CL cost	CL + CR cost	Best heuristic cost
Random problems								
10	15	[0,350]	6.9%	1.5%	1.3%	8.1%	6.8%	0.03%
20	30	[0,350]	6.9%	1.6%	1.4%	7.3%	5.6%	≤ 0.22%
30	45	[0,350]	6.8%	1.7%	1.5%	6.1%	4.5%	
40	60	[0,350]	7.1%	1.8%	1.6%	5.7%	4.6%	
50	75	[0,350]	7.1%	1.8%	1.6%	6.0%	4.6%	
10	15	[0,100]	24.0%	6.1%	5.5%	11.8%	9.6%	0.03%
20	30	[0,100]	24.1%	6.0%	5.1%	10.6%	8.9%	≤ 0.29%
30	45	[0,100]	23.6%	5.7%	4.7%	9.8%	7.5%	
40	60	[0,100]	24.8%	5.8%	5.1%	8.8%	7.5%	
50	75	[0,100]	24.7%	6.3%	5.3%	10.1%	8.4%	
10	15	[0,10]	234.5%	61.4%	60.7%	63.1%	60.8%	0.28%
20	30	[0,10]	235.9%	129.8%	61.7%	136.8%	133.8%	
30	45	[0,10]	231.2%	67.7%	65.3%	70.7%	64.6%	
40	60	[0,10]	242.2%	66.3%	62.6%	65.9%	63.6%	
50	75	[0,10]	241.2%	67.9%	64.0%	69.1%	66.6%	
10	30	[0,350]	12.0%	4.7%	3.2%	39.0%	33.5%	≤ 0.08%
20	60	[0,350]	11.8%	3.7%	2.5%	33.4%	27.0%	
30	90	[0,350]	12.1%	4.0%	2.9%	34.0%	28.1%	
40	120	[0,350]	12.1%	4.3%	3.0%	36.5%	30.7%	
50	150	[0,350]	12.0%	4.2%	3.0%	36.8%	30.4%	
10	30	[0,100]	41.9%	15.8%	11.7%	46.6%	40.0%	≤ 3.44%
20	60	[0,100]	41.3%	13.5%	10.4%	40.1%	34.9%	
30	90	[0,100]	42.4%	14.3%	10.1%	42.1%	36.0%	
40	120	[0,100]	42.4%	14.2%	10.5%	45.2%	38.9%	
50	150	[0,100]	41.9%	15.4%	11.4%	36.7%	30.1%	
10	30	[0,10]	407.9%	126.6%	98.4%	145.0%	132.9%	
20	60	[0,10]	401.0%	111.6%	89.6%	128.4%	114.6%	
30	90	[0,10]	412.0%	119.8%	91.6%	133.7%	120.4%	
40	120	[0,10]	411.0%	117.5%	90.5%	139.8%	123.7%	
50	150	[0,10]	407.8%	118.3%	92.7%	134.1%	115.8%	
Real-world problems								
10	14	[13,302]	8.39%	1.46%	1.43%	10.24%	3.56%	0.50%
11	23	[16,81]	35.36%	11.04%	8.75%	21.20%	14.95%	
53	79	[0,94]	45.59%	10.01%	8.73%	15.88%	13.36%	

Table 3. Computational results for MCP problems with cost scale factor *d = 0.1*

Problem Size			% from optimal LP value					% from optimal IP value
Nodes	Edges	Flow Range	Upper Bound on LP round-up	LP round-up cost	LP round-up + CR cost	CL cost	CL + CR cost	Best heuristic cost
Random problems								
10	15	[0,350]	11.8%	3.2%	2.5%	8.6%	7.2%	0.00%
20	30	[0,350]	11.9%	3.4%	3.0%	9.6%	7.8%	≤ 0.98%
30	45	[0,350]	11.6%	3.3%	2.6%	7.6%	5.3%	
40	60	[0,350]	12.3%	3.3%	2.4%	6.3%	5.0%	
50	75	[0,350]	12.2%	3.3%	2.5%	7.3%	5.6%	
10	15	[0,100]	41.3%	10.4%	7.3%	15.9%	13.6%	0.00%
20	30	[0,100]	41.4%	12.2%	9.1%	13.3%	12.5%	≤ 1.09%
30	45	[0,100]	40.6%	11.0%	8.3%	13.4%	11.2%	
40	60	[0,100]	42.7%	11.4%	9.4%	12.7%	11.2%	
50	75	[0,100]	42.5%	11.7%	9.3%	12.6%	10.6%	
10	15	[0,10]	403.0%	131.6%	88.8%	113.7%	85.9%	2.2%
20	30	[0,10]	405.6%	228.6%	198.7%	212.0%	165.9%	
30	45	[0,10]	397.4%	135.8%	111.1%	118.5%	94.6%	
40	60	[0,10]	416.3%	140.5%	114.9%	122.9%	99.6%	
50	75	[0,10]	414.6%	143.5%	122.7%	126.7%	109.3%	
10	30	[0,350]	20.7%	9.1%	4.8%	40.6%	31.9%	≤ 0.28%
20	60	[0,350]	20.3%	7.4%	3.9%	32.9%	27.4%	
30	90	[0,350]	20.9%	7.4%	3.9%	33.7%	27.6%	
40	120	[0,350]	20.9%	7.8%	4.2%	35.2%	32.0%	
50	150	[0,350]	20.6%	8.1%	4.4%	32.7%	30.5%	
10	30	[0,100]	72.1%	29.5%	13.9%	48.2%	41.1%	≤ 0.33%
20	60	[0,100]	71.0%	27.6%	13.8%	40.3%	32.4%	
30	90	[0,100]	72.9%	27.1%	14.4%	42.4%	36.3%	
40	120	[0,100]	72.9%	28.1%	14.8%	43.9%	35.2%	
50	150	[0,100]	72.0%	29.7%	13.9%	40.0%	34.1%	
10	30	[0,10]	701.3%	428.7%	321.8%	224.2%	180.6%	
20	60	[0,10]	689.4%	351.3%	279.9%	238.8%	204.4%	
30	90	[0,10]	708.3%	386.1%	294.0%	247.0%	194.8%	
40	120	[0,10]	706.6%	391.5%	297.6%	236.2%	197.9%	
50	150	[0,10]	701.0%	398.6%	310.8%	250.3%	202.4%	
Real-world problems								
10	14	[13,302]	14.4%	3.84%	3.18%	15.46%	6.42%	0.00%
11	23	[16,81]	60.79%	19.99%	12.87%	22.61%	16.85%	
53	79	[0,94]	78.37%	18.17%	14.26%	26.00%	22.90%	

Table 4. Computational results for random [0,1] SCP problems

Problem Size			% from optimal LP value				% from optimal IP value
Nodes	Edges	Flow Range	LP round-up cost	LP round-up + CR cost	CL cost	CL + CR cost	Best heuristic cost
10	20	[0,1]	110%	50%	65%	35%	0%
20	40	[0,1]	44%	25%	56%	25%	5%
30	60	[0,1]	99%	27%	40%	27%	4%
40	80	[0,1]	133%	41%	57%	41%	12%
50	100	[0,1]	134%	34%	48%	25%	2%

Appendix A - Random Graph Generation

Given the desired number of nodes, number of arcs, and minimum and maximum potential working flow, the random graph generator creates a random doubly-connected graph using the following procedure:

Random Graph Generation Procedure

- Create a random spanning tree.
- While number of edges < target number of edges
  - Choose a new edge randomly as follows:
    - Select a current edge (*i*,*j*) whose endpoints are not doubly connected.
    - Temporarily remove (*i*,*j*), splitting the graph into two disconnected subgraphs.
    - Randomly select nodes $\hat{i}$ and $\hat{j}$, one from each subgraph.
    - If the selected nodes are *i* and *j*, repeat the node selection step.
    - Otherwise, add edge $(\hat{i}, \hat{j})$.
    - Return edge (*i*,*j*) to the graph.
- Update the list of edges whose endpoints are not doubly connected.
- If the endpoints of some remaining edges are not doubly connected, restart procedure.
- Otherwise, for each edge, randomly choose a working flow in the specified range.

The procedure generates random numbers using a linear congruential generator first proposed by Lewis, Goodman, and Miller [18].

Every edge we add to the initial spanning tree will necessarily create (and thus be part of) a cycle. Thus, the only edges that might not be part of a cycle (and thus whose endpoints might not be doubly connected) are the edges of the initial spanning tree. At each iteration, we select our current edge randomly from these candidates. When we create the initial spanning tree, we simultaneously create a table listing the first edge in the tree-arc path between every two nodes. This table serves two purposes. First, when we remove edge (*i*,*j*) at each iteration, we use the table to quickly define the disconnected subgraphs. Any node *l* such that the *i*,*l*th table entry is *j* is in one subgraph, and all other nodes are in the other subgraph. Second, when we add edge $(\hat{i}, \hat{j})$ to the graph, we use the table to quickly reconstruct the tree-arc paths from *i* to $\hat{i}$ and from *j* to $\hat{j}$. All nodes along these paths are now doubly connected, and so in the update step, we remove all edges on these paths, as well as edge (*i*,*j*), from the edge candidate list. Although procedure does not guarantee producing a doubly-connected graph on the first attempt (except when $m \geq 2n-3$), we found that it succeeds very often, and in any case requires very little computational time even with “second-tries”.

References

[1] Ahuja, R., T. L. Magnanti, and J. Orlin. Network Flows: Theory, Algorithms, and Applications. Prentice-Hall, Englewood Cliffs, N.J., 1993.

[2] Balakrishnan, A., T. L. Magnanti, and P. Mirchandani. "A Dual-based Algorithm for Multi-level Network Design." Management Science. Volume 40, Number 5 (1994). pp. 567-81.

[3] Balakrishnan, A., T. L. Magnanti, J. S. Sokol, and Y. Wang. "Spare Capacity Assignment for Line Restoration Using a Single Facility Type." Working paper, Department of Management Science & Information Systems, Pennsylvania State University, University Park, PA (2000).

[4] Bienstock, D. and G. Muratore. "Strong Inequalities for Capacitated Survivable Network Design Problems." Working Paper, Columbia University, New York, 1997.

[5] Chujo, T., H. Komine, K. Miyazaki, T. Ogura, and T. Soejima. "Distributed Self-healing Network and its Optimum Spare-capacity Assignment Algorithm." Electronics and Communications in Japan, Part 1. Volume 74, Number 7 (1991). pp. 1-9.

[6] Dahl, G. and M. Stoer. "A Cutting Plane Algorithm for Multicommodity Survivable Network Design Problems." INFORMS Journal on Computing, volume 10, number 1, pp. 1-11, 1998.

[7] Dunn, D. A., W. D. Grover, and M. H. MacGregor. "Comparison of k-shortest Paths and Maximum Flow Routing for Network Facility Restoration." 1994 IEEE JSAC Special Issue on Integrity of Public Telecommunications Networks. Volume 12, Number 1. pp. 88-99.

[8] Fahim, F. and J. C. Shah. Private Correspondence. September 1994 - May 1995.

[9] Garcia, V., and P. Ritchi. Private Conversation. Fall 1996.

[10] Groetschel, M., C. Monma, and M. Stoer. "Design of Survivable Networks." Handbooks of OR and MS, volume 7, 1995.

[11] Grover, W. D., T. D. Bilodeau, and B. D. Venables. "Near Optimal Spare Capacity Planning in a Mesh Restorable Network." 1991 IEEE Global Telecommunications Conference. pp. 2007-12.

[12] Herzberg, M. "A Decomposition Approach to Assign Spare Channels in Self-healing Networks." 1993 IEEE Global Telecommunications Conference. pp. 1601-5.

[13] Herzberg, M., S. J. Bye, and A. Utano. "The Hop-limit Approach for Spare Capacity Assignment in Survivable Networks." IEEE/ACM Transactions on Networking. Volume 3, Number 6 (1995). pp. 775-84.

[14] Iri, M. "How to Generate Realistic Sample Problems for Network Optimization." Lecture Notes in Computer Science. Number 650 (1992). pp. 342-250.

[15] Kennington, J. and J. E. Whitler. "An Efficient Decomposition Algorithm to Optimize Spare Capacity in a Telecommunications Network." INFORMS Journal on Computing, volume 11, number 2, pp. 149-160, 1999.

[16] Lee, L. and H. W. Chun. "An ANN Approach to Spare Capacity Planning." 1992 IEEE Region 10 International Conference. pp. 891-5.

[17] Lee, Y. Computation Analysis of Network Optimization Algorithms. Ph.D. Thesis, Massachusetts Institute of Technology, 1993.

[18] Lewis, P. A. W., A. S. Goodman, and J. M. Miller. "A Pseudo-random Number Generator for the System 360." IBM System Journal. Volume 8 (1969). pp. 136-146.

[19] Lisser, A., R. Sarkissian, and J. Vial. "Survivability in Transmission Telecommunication Networks." Technical Report 1995.3, CNET, Paris, France, 1995.

[20] Lisser, A., R. Sarkissian, and J. Vial. "Optimal Joint Syntheses of Base and Spare Telecommunication Networks." International Sympousion on Mathematical Programming, 1997.

[21] Magnanti, T. L., P. Mirchandani, and R. Vachani. "Modeling and Solving the Two-facility Capacitated Network Loading Problem." Operations Research. Volume 43, Number 1 (1995). pp. 142-57.

[22] Magnanti, T.L. and Y. Wang. "Polyhedral Properties of the Network Restoration Problem – With the Convex Hull of a Special Case." Working Paper OR 323-97, Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA, 1997.

[23] Magnanti, T. L. and L. Wolsey. "Optimal Trees." Handbooks in Operations Research and Management Science, Volume 7. M. O. Ball et al., Eds. Elsevier Science B.V., 1995.

[24] Meshkovskiy, K. A. and A. Y. Rokotyan. "Restoration of Communications Network Connectivity Following the Failure of Transmission Junctions and Lines." Telecommunications and Radio Engineering. Volume 47, Number 1. January 1992. pp. 1-5.

[25] Minoux, M. "Network Synthesis and Optimum Network Design Problems." Networks, volume 19, number 3, pp. 313-360, 1989.

[26] Sakauchi, H., Y. Nichimura, and S. Hasegawa. "A Self-healing Network with an Economical Spare Channel Assignment." 1990 IEEE Global Telecommunications Conference. pp. 438-43.

[27] Stoer, M. and G. Dahl. "A Polyhedral Approach to Multicommodity Survivable Network Design." Numerische Mathematik. Volume 68, Number 1, pp. 149-167, 1994.

[28] Vachani, R. and P. Kubat. "Design of Survivable Telecommunications Networks: Models and Algorithms." ORSA/TIMS Joint National Conference, Phoenix, 1994.

[29] Veerasamy, J., S. Venkatesan, and J. C. Shah. "Effect of Traffic Splitting on Link and Path Restoration Planning." 1994 IEEE Global Telecommunications Conference. pp. 1867-71.

[30] Veerasamy, J., S. Venkatesan, and J. C. Shah. "Spare Capacity Assignment in Telecom Networks Using Path Restoration." MASCOTS 1995. pp. 370-75.

[31] Venables, B. D. Algorithms for the Spare Capacity Design of Mesh Restorable Networks. Master of Science Thesis, University of Alberta. Fall 1992.

[32] Venables, B. D., W. D. Grover, and M. H. MacGregor. "Two Strategies for Spare Capacity Placement in Mesh Restorable Networks." 1993 IEEE International Conference on Communications. pp. 267-71.

[33] Yang, C. and S. Hasegawa. "FITNESS: Failure Immunization Technology for Network Service Survivability." 1988 IEEE Global Telecommunications Conference. pp. 1549-54.