A Decomposition Algorithm for Expanding Local Access Telecommunications Networks

Anantaram Balakrishnan, Thomas L. Magnanti and Richard T. Wong

OR 244-91 April 1991

A Decomposition Algorithm for Expanding Local Access Telecommunications Networks †

Anantaram Balakrishnan *
Thomas L. Magnanti

Sloan School of Management
M. I. T.
Cambridge, MA

Richard T. Wong

AT&T Bell Laboratories
Holmdel, NJ

April 1991

† This research was initiated through a grant from GTE Laboratories Incorporated, Waltham, Massachusetts

* Supported in part by a research award from the AT&T Foundation

Abstract

Growing demand and increased diversity of telecommunication services, combined with recent advances in transmission and switching technologies, are revolutionizing the telecommunications industry. As part of the changes in the industry, telecommunication companies are rapidly expanding and modernizing the communication network hierarchy. This paper develops and tests a decomposition methodology to generate cost-effective expansion plans, with performance guarantees, for one major component of this hierarchy—the local access network connecting customers to the local switching center. The model captures essential tradeoffs between installing concentrators and expanding cables to accomodate demand growth; it also addresses economies of scale in facility costs. By exploiting the special structure of the expansion planning problem, our solution method integrates two major solution strategies from mathematical programming—the use of valid inequalities, obtained by studying a problem's polyhedral structure, and dynamic programming, which can be used to solve an uncapacitated version of the local access network expansion planning problem. The computational results for three actual test networks demonstrate that this enhanced dynamic programming algorithm, when embedded in a Lagrangian relaxation scheme (with problem preprocessing and local improvement), is very effective in generating good upper and lower bounds: implemented on a personal computer, the method was able to generate solutions guaranteed to be within 1.2 - 7.2% of optimality. Apart from developing a successful solution methodology for a practical problem, this paper illustrates the possibility of effectively combining decomposition methods and polyhedral approaches.

Keywords: *Integer programming decomposition, concentrator location, telecommunications planning, polyhedral methods*

1. Introduction

The telecommunication industry is changing rapidly. In particular, recent advances in digital switching and transmission technologies combined with deregulation of the industry have created many opportunities for telephone companies to diversify their basic services, from traditional voice transmission to data, video, telemetry and other communication services. To exploit the increased demand that these opportunities will stimulate, telephone companies are actively modernizing and expanding their transmission facilities. Since these investments are so very expensive, network expansion planning is a prime candidate for analysis via optimization-based decision support systems. Because the significant fixed costs for installing switching and transmission resources necessitate discrete choice decisions, network planning is a complex and challenging task involving large-scale combinatorial optimization. This paper develops and tests an optimization-based methodology for developing a cost-effective expansion strategy for one portion of the overall telecommunication system, the local access network.

The local access network connects customer nodes to a local switching center which, in turn, communicates with other switching centers via the inter-office and backbone networks. Most existing local access networks have a tree structure, i.e., a unique path connects each customer node to its assigned switching center. The U. S. telephone system contains over 10,000 local access networks, and approximately 60% of the 100 billion dollars invested in U. S. telecommunication facilities resides in these networks. Thus, effective local network design and expansion planning can have a significant economic impact.

Local access networks are not as technologically advanced as other components (the inter-office and long-distance networks) of the telecommunication system. For instance, although the telecommunication industry has moved rapidly to invest in digital transmission, electronic switching, and fiber optic transmission in long-distance networks, over 80% of the local networks continue to use only analog transmission over copper cables. However, the large anticipated growth in demand and increasing diversity of services is accelerating the introduction of newer technologies in the local network such as electronic multiplexers, remote switches, and digital transmission. These technologies provide alternative means to increase capacity and complement the conventional expansion strategy of installing more copper cables. Correspondingly, network planners now require more sophisticated decision support tools to incorporate the additional capacity expansion options and the spatial couplings that the new devices introduce. An earlier survey paper (Balakrishnan, Magnanti, Shulman, and Wong [1991a]) reviews the evolution of local access networks, and motivates various approaches for modeling contemporary network expansion planning problems.

This paper describes an optimization methodology for planning local access network expansion, and reports on computational experience using this methodology. In a companion paper (Balakrishnan, Magnanti, and Wong [1991b]), we proposed an integer programming model for planning capacity expansion in local access networks, and partially characterized the polyhedral structure of this formulation. The model captures the essential tradeoffs between concentrator location and cable expansion, and accomodates economies of scale in investment and operating costs. We identified several different classes of valid inequalities that strengthen the linear programming relaxation of the problem formulation. In developing these inequalities, our aim was to use stronger problem formulations to achieve improvements in algorithmic performance. The current paper builds upon these modeling results to develop a decomposition algorithm for solving the local access network planning problem. Our method combines Lagrangian relaxation with a dynamic programming algorithm that incorporates some of the valid inequalities identified in our previous study. Using data derived from three actual local access networks, we demonstrate the effectiveness of the valid inequalities in generating good upper and lower bounds.

This paper makes three contributions: (i) it develops an effective method for solving the important practical problem of local access network design; (ii) it adds to the growing literature demonstrating the usefulness of polyhedral methods for solving difficult, large-scale combinatorial problems; and (iii) unlike other cutting plane methods that use general purpose linear programming codes to solve successively stronger relaxations, this research demonstrates how to tailor a dynamic programming algorithm to directly incorporate valid inequalities.

The rest of this paper is organized as follows: Section 2 presents a formal definition of the local access network planning problem, reviews our modeling assumptions, and describes a basic mixed-integer programming formulation. Section 3 describes the Lagrangian relaxation scheme for the basic model, develops a dynamic programming algorithm to solve the main subproblem, and outlines a procedure to construct heuristic expansion plans from the Lagrangian solution. In Section 4 we describe two algorithmic enhancements—a problem preprocessing procedure to eliminate variables, and a coefficient reduction method to strengthen the problem formulation. Section 5 shows how to modify the dynamic program to incorporate several classes of valid inequalities. Section 6 describes our implementation, and presents computational results for three test problems obtained from real data. We illustrate how the valid inequalities dramatically improve the lower bounds (by about 80%) relative to the basic model, and we study the robustness of the method to changes in demand and cost parameters. Our results show that the combination of Lagrangian relaxation, dynamic programming, and polyhedral methods permits us to efficiently find solutions guaranteed to be within

about 4% of optimality, and using a personal computer. Section 7 identifies directions for further work.

2. The Basic Local Access Network Expansion Model

This section presents a f rmal definition of the problem and reviews our modeling assumptions. Balakrishnan et al. [1991b] provide more detailed discussions and justification of the model and its assumptions.

2.1 Problem description

The local access network connects customer nodes (in telecommunication parlance called control or distribution points) to the switching center. Each customer node is a "collection point" for 20 to 200 individual customers connected via a subsidiary distribution network. Each switching center serves all the customer nodes (ranging from 10 to more than 50) in a predetermined geographical region. In the basic network without remote switches, all communications (both within and outside the local switching region) to and from each customer node flow through the assigned switching center; a unique circuit, i.e., an electrical transmission path, connects each customer to the switching center. We associate a **demand** with each customer node. This demand represents the required number of circuits from the customer node to the switching center; it depends on the number and type (e.g., residential or commercial) of customers assigned to that customer node. The local access network can "satisfy" this demand in two ways: either provide a dedicated cable (from the customer node to the switching center) for each required circuit, or route the circuits through a traffic compression device called a **concentrator**. Concentrators are electronic devices that compress incoming traffic, i.e., they combine incoming signals on multiple lines into a single composite (higher frequency) signal that requires only one outgoing line. In practice, a variety of devices can perform traffic compression: multiplexers (time division or frequency division), concentrators (which dynamically allocate output lines to incoming circuits), remote switches, and fiber optic terminals. For convenience, we collectively refer to all these devices as concentrators since they perform essentially equivalent functions.

As demands at the customer nodes increase (for example, due to new construction, customer movement, or new services), the existing cables and concentrators can no longer accomodate the required number of circuits from each customer node. The expansion planning problem then consists of strategically locating new concentrators, selectively expanding cable capacities, and rerouting traffic from customer nodes via concentrators in order to satisfy demand at minimum total network expansion cost. Observe that by routing traffic through

concentrators we reduce the downstream cable requirements. This tradeoff between installing concentrators and expanding cable capacities is central to the local access network expansion problem. We next introduce some notation and formally describe the expansion planning model and its assumptions.

Problem parameters

Our capacity expansion model applies to local access telecommunication networks that have a tree topology. Let the root node of the tree represent the switching center. All other nodes of this tree represent customer nodes and/or potential concentrator locations. The edges of the tree correspond to physical cable sections connecting adjacent nodes; by joining cables in successive sections, we can create an electrical path between non-adjacent nodes. Let **T** denote the given (undirected) rooted tree over which the local access network expansion problem is defined. We index the set of nodes **N** of this tree from 0 to n, with the **root node 0** representing the switching center. Every customer node i of the network has a known projected **demand**, denoted as d_i, which represents the number of circuits from node i to the switching center required at the end of the planning horizon. For simplicity, all our subsequent discussions *assume that the existing network does not contain any concentrators,* even though our method extends easily to problems with existing concentrators. We treat the number of currently existing cables (e.g., twisted wire pairs) connecting adjacent nodes i and j as the existing **capacity** B_{ij} of the edge (i,j) in the tree.

Consider a "conventional" network that does not employ any concentrators. Let P_{ij} denote the (unique) **path** connecting any pair of nodes i and j. To provide one circuit from node i to the switching center, we must allocate one cable belonging to each intermediate section along the path P_{i0}; joining the selected cable at each intermediate node creates the required (physical) circuit. Since the existing network does not have adequate capacity to meet the projected demand, one or more sections in the curent network must have projected *exhaust*, i.e., the number of available cables in that section is less than the total demand for all nodes communicating through that section to the switching center. To meet this demand, the network planner can either (i) expand cable capacities on sections with projected exhaust, or (ii) install concentrators at selected locations to relieve the exhaust. Observe that we permit cable expansion only along currently existing sections, i.e., the expansion plan must preserve the tree structure. In practice, creating new cable sections is very expensive because of the costs of acquiring land and building needed infrastructure.

Cost structure

To make tradeoffs between concentrator location and cable expansion, our model accounts for three types of costs: ***cable*** costs, ***concentrator*** costs, and node-to-concentrator ***connection*** or ***rerouting*** costs. *Cable expansion* costs might vary by section (typically, the expansion cost depends on the length and location of the

section), and consist of fixed as well as variable components. The **fixed cable cost** G_{ij} (also called the *cable installation cost*) for section (i,j) represents the expenses for digging trenches, laying pipes, and other costs incurred between locations i and j. The **variable cable cost** e_{ij} is proportional to the number of cables that are added; this cost might consist of the investment in cables as well as operating expenses for maintenance. *Concentrator* costs also consist of fixed and variable components, and might vary by location. The **fixed concentrator cost** F_j models land acquisition and infrastructure investments at node j, while the **variable** (i.e., throughput-dependent) **cost** c_j reflects the purchase price and operating expenses of electronic concentrator modules. Our solution procedure can accomodate a more general piecewise-linear, concave cost structure as shown in Figure 1 that might represent, for instance, alternative technologies or economies of scale in cable and concentrator costs. (This figure represents the cost structure for a cable or concentrator with four alternate technologies, each with its own fixed plus variable cost.) For expositional ease in describing the model formulation and solution algorithm, we will assume the simpler fixed plus variable cost structure for both cable expansion and concentrator location. We indicate how to extend the solution procedure to the more general cost structure, and our implementation solves test problems with concave concentrator costs. Observe that the true cost of, say, installing a concentrator might be a step function of the required capacity since concentrator modules are available only in discrete units. Thus, even the concave cost function might only approximate the true cost. However, our discussions with network planners and some preliminary data analysis of actual cost functions suggested that the concave approximation is adequate for long-term capacity planning purposes.

In addition to the cable and concentrator costs, our model also incorporates a **connection** or **rerouting** cost A_{ij} for every node pair i, j. We incur this connection cost if the network expansion strategy reroutes the circuits from node i through a concentrator located at node j. This cost includes the cost of disconnecting the current circuit (from the switching center) and the cost of reconnecting the circuit through node j. The cost might vary from node to node because of varying demands and circuit distances. We can also use this cost parameter to prohibit certain routing patterns (for instance, to avoid deterioration in transmission quality) by setting the corresponding connection cost to a very high value.

2.2 Modeling assumptions

To reduce the complexity of managing and maintaining the local access telecommunication network, planners often impose several restrictions on the permissible expansion options and routing patterns. To keep the problem tractable, we also make an additional assumption (assumption A4) regarding the cost for transmitting concentrated traffic. Discussions with planners in industry suggest that the following four modeling assumptions reflect or adequately approximate current

practice. Balakrishnan et al. [1991b] discuss these modeling assumptions in greater detail.

Assumption A1: Single-level concentration
Traffic originating at any node of the network is concentrated at most once before reaching the switching center.

Assumption A2: Non-bifurcated routing
Traffic from each node follows a unique route to the switching center, i.e., all circuits from each customer node use the same cable sections, and a common concentrator if necessary.

Assumption A3: Contiguity restriction
Every concentrator serves a contiguous region surrounding it, i.e., if node i's traffic is compressed by a concentrator located at node j, then this concentrator also compresses traffic from all the nodes lying on the path P_{ij} connecting nodes i and j (including node j itself).

Assumption A4: Transmission cost for concentrated traffic
The (fixed and capacity-dependent) cost of transmitting concentrated traffic (from the concentrator to the switching center) is separable by concentrator location, i.e., concentrated traffic either consumes a negligible amount of existing cable capacity (and, hence, entails negligible transmission cost), or uses an umbilical (dedicated) connection to the switching center.

Notice that, since the local access network has a tree structure, assumptions A1 and A2 together imply that selecting a concentrator location j for each node i completely specifies the routing pattern in the network. We say that **node i homes on node j** if a concentrator located at node j processes node i's traffic; in this case, the route from node i to the switching center consists of the unique path P_{ij} connecting node i to node j, followed by path P_{0j} connecting node j to the switching center. Any node whose traffic is not concentrated is said to home on the switching center. The contiguity assumption (A3) reduces the concentrator assignment (and, hence, routing) decision to a problem of decomposing (and covering) the tree into subtrees, and selecting one concentrator location within each subtree to serve its traffic requirements. We use this assumption to efficiently solve an uncapacitated subproblem using dynamic programming. Assumption A4 enables us to include the cost of transmitting concentrated traffic from any node j to the switching center in the total concentrator cost for node j.

Observe that, even with these restrictions, the final plan might involve complex routings. For instance, we assume that cables are bidirectional, and we permit ***backfeed*** or flow away from the switching center. Thus, to avoid cable

expansion, node i may home on a node j that does not lie on the path P_{i0} connecting node i to the switching center (node 0). Next, we present the basic mixed-integer programming formulation for the local access network expansion problem.

2.3 Basic Integer Programming Formulation

Given the projected demand at each customer node, the existing cable capacity in each section, and the costs for rerouting, for adding cables, and for installing concentrators at each location, the local network expansion planning task consists of deciding:

- where to locate concentrators, and with what capacity;
- which cable sections to expand, and by how much; and,
- how to route the traffic from each node to the switching center.

As we noted in Section 2.2, our assumptions reduce the traffic routing decision to selecting the homing node for each customer node (i.e., choosing a concentrator or the switching center to process each node's traffic). Consequently, our formulation uses binary decision variables, called *assignment variables*, to assign homing nodes. In addition, it uses binary variables to determine concentrator locations (*concentrator location variables*), and binary and continuous variables for adding cables (*cable installation* and *cable expansion variables*, respectively). We define these variables as follows:

Assignment variable	x_{ij}	=	1 if node i homes on node j, 0 otherwise;
Concentrator Location variable	y_j	=	1 if we install a concentrator at node j, 0 otherwise;
Cable Installation variable	z_{ij} (z_{ji})	=	1 if we expand cable capacity from node i to node j (node j to node i), 0 otherwise; and,
Cable Expansion variable	s_{ij} (s_{ji})	=	number of cables added from node i to node j (j to i).

Observe that we have used directed variables for modeling cable installation and expansion. Therefore, on any cable section connecting nodes i and j, we distinguish between expansion in the i-to-j direction from expansion in the j-to-i direction even though cables are bidirectional (i.e., transmission is permitted in either direction on

existing cables). Using undirected cable installation and expansion variables gives an equivalent mixed-integer formulation with fewer decision variables. However, as we shall see later, the use of directed cable addition variables enables us to strengthen the model's linear programming relaxation. To emphasize the direction of flow, we will consider two directed *arcs*, denoted as <i,j> and <j,i>, corresponding to each original undirected *edge* (i,j), and we redefine P_{ij} as the *directed path* from node i to node j in the tree.

In terms of the decision variables x, y, z, and s, the Local Access Network Expansion Planning Problem has the following basic mixed-integer programming formulation:

Network Expansion Planning Model [LAN1]

$$\text{minimize} \quad \sum_{i \in N} \sum_{j \in N} A_{ij} x_{ij} + \sum_{j \in N} F_j y_j + \sum_{i \in T} \sum_{j \in T} (d_i c_j) x_{ij}$$

$$+ \sum_{(i,j) \in T} G_{ij} (z_{ij} + z_{ji}) + \sum_{(i,j) \in T} e_{ij} (s_{ij} + s_{ji}) \qquad (2.1)$$

subject to

Assignment constraints

$$\sum_{j \in N} x_{ij} = 1 \qquad \text{all } i \in N, \qquad (2.2)$$

Concentrator location constraints

$$y_j = x_{jj} \qquad \text{all } j \in N, \qquad (2.3)$$

Contiguity restrictions

$$x_{ij} \leq x_{k_{ij}j} \qquad \text{all } i,j \in N, \qquad (2.4)$$

Cable capacity constraints

$$\sum_{k,l \in OD_{ij}} d_k x_{kl} \leq B_{ij} + s_{ij} + s_{ji} \qquad \text{all } (i,j) \in T, \qquad (2.5)$$

Cable installation-forcing constraints

$$s_{ij} \leq M_{ij} z_{ij} \qquad (2.6a)$$

$$s_{ji} \leq M_{ji} z_{ji} \qquad \text{all } (i,j) \in T, \qquad (2.6b)$$

Arc orientation constraints

$$z_{ij} + z_{ji} \leq 1 \qquad \text{all } (i,j) \in T, \text{ and} \qquad (2.7)$$

Integrality/Nonnegativity constraints

$$y_j\ x_{ij},\ z_{ij},\ z_{ji} = 0 \text{ or } 1 \qquad \text{all } j \in N,\ (i,j) \in T, \text{ and} \qquad (2.8)$$

$$s_{ij},\ s_{ji} \geq 0 \qquad \text{all } (i,j) \in T. \qquad (2.9)$$

In this formulation,

k_{ij}	is the node adjacent to node i on the path P_{ij} from node i to node j,
OD_{ij}	is the set of node pairs k,*l* whose path P_{kl} contains edge (i,j), and
M_{ij} (M_{ji})	is an upper bound on the maximum required cable expansion on section (i,j) in the i-to-j (j-to-i) direction.

The objective function (2.1) seeks to minimize the sum of the node-to-concentrator connection costs, the fixed and variable concentrator costs, and the cable installation and expansion costs. Observe that, although we use the same fixed and variable cable costs (G_{ij} and e_{ij}) in both directions on each edge, the model and our solution approach permit different costs in the two directions. Constraints (2.2) specify that the solution must assign each node i to exactly one concentrator or to the switching center (the latter decision corresponds to setting $x_{i0} = 1$). Equation (2.3) specifies that node i must contain a concentrator ($y_j = 1$) if and only if this node homes on itself (i.e., $x_{jj} = 1$). This restriction, in conjunction with the contiguity constraints (2.4), ensures that node j contains a concentrator whenever any other node homes on this node j. Constraints (2.4) state that if node i homes on node j, then node i's immediate neighbor k_{ij} on the path P_{ij} (connecting node i to node j) must also home on j. Including these constraints for all node pairs i,j ensures that each concentrator serves a (surrounding) contiguous subtree. We model the cable capacity constraints as inequalities (2.5). The left-hand side of the capacity restriction for section (i,j) corresponds to the total flow on edge (i,j) expressed in terms of the assignment variables x_{kl} for all node pairs k,*l* $\in OD_{ij}$ that communicate via this edge.

Constraints (2.6) are forcing constraints that relate the binary cable installation variables z_{ij} to the continuous expansion variables s_{ij}. If the amount of cable expansion s_{ij} (in the i-to-j direction) is positive, this constraint forces the installation variable z_{ij} to assume a value of 1, thus absorbing the fixed cable expansion cost G_{ij} in the objective function. The parameter M_{ij} (M_{ji}) in the right-hand side of this constraint represents the maximum cable expansion that section (i,j) can possibly require in the i-to-j (j-to-i) direction in any feasible solution. Constraint (2.7) specifies that section (i,j) can be expanded either in the i-to-j or j-to-i direction, but not both.

We refer to M_{ij} as the ***cable expansion bound*** from node i to node j. To compute this parameter, consider the following naive method: When we remove edge (i,j), the original tree T decomposes into two subtrees, say, T' and T''. Suppose

T' contains node i and T" contains node j, and let D' and D" represent the sum of nodal demands in the respective subtrees. Then, the maximum possible flow in the i-to-j (j-to-i) direction is D' (D"); thus, we can set M_{ij} = Max {0, (D'-B_{ij})} and M_{ji} = Max {0, (D"-B_{ij})}. Later (in Section 4.2), we indicate how to obtain tighter values for M_{ij} and M_{ji}.

Formulation [LAN1] is a large-scale mixed-integer program, whose size (number of variables and constraints) increases quadratically with the number of nodes. Like many other design problems, this problem is NP-complete (Balakrishnan et al. [1991b]). Thus, finding the optimal local network expansion plan is a computationally difficult task. However, as we show in Section 3, a dynamic programming algorithm will permit us to solve the expansion problem in polynomial time if the network does not contain any existing cable capacities. We propose a decomposition approach to exploit this observation. In the next three sections, we describe the basic solution procedure for the problem and several enhancement techniques for improving algorithmic performance.

3. Decomposition Algorithm for the Basic Local Access Network Planning Model

This section outlines our approach for solving the basic [LAN1] model. The method consists of three components: (i) preprocessing and coefficient reduction, i.e., performing some prior analysis to reduce the problem size and strengthen the formulation; (ii) solving the Lagrangian subproblems to generate lower bounds on the optimal cost; and (iii) generating good heuristic solutions from the Lagrangian subproblem solutions. We describe the second and third components in this section. In subsequent sections, we describe the preprocessing and coefficient reduction methods, and propose various additional formulation and algorithmic enhancements to improve the method's performance.

3.1 The Lagrangian Relaxation Scheme

The uncapacitated local access network planning problem, which assumes that the given network has no current cable capacity, is easy to solve optimally using dynamic programming. To exploit this property, we use a Lagrangian relaxation scheme that dualizes the cable capacity constraints (2.5) using *Lagrange multipliers* μ_{ij} for all edges (i,j) $\in$ T (see, for instance, Fisher [1981] for a review of the Lagrangian relaxation method). The resulting Lagrangian problem becomes:

$$\text{minimize} \quad \sum_{i \in N} \sum_{j \in N} A_{ij}(\mu)\, x_{ij} + \sum_{j \in N} F_j\, y_j + \sum_{(i,j) \in T} G_{ij}\,(z_{ij} + z_{ji})$$

$$+ \sum_{(i,j) \in T} (e_{ij} - \mu_{ij})\,(s_{ij} + s_{ji}) - \sum_{(i,j) \in T} \mu_{ij}\, B_{ij}$$

subject to: constraints (2.2) - (2.4) and (2.6) - (2.9).

In this formulation, we have consolidated all the cost coefficients of the assignment variable x_{ij} into an *assignment cost* $A_{ij}(\mu)$ defined as:

$$A_{ij}(\mu) = A_{ij} + d_i \Big(\sum_{(k,l) \in P_{ij}} \mu_{kl} \Big) + d_i\, c_j \qquad \text{for all } i, j \in N. \qquad (3.1)$$

This cost represents a fixed charge for assigning node i to a concentrator at node j. Intuitively, the cost μ_{kl} represents an imputed variable or marginal cost for using capacity on arc (k,*l*). Thus, the second term in the right-hand side of equation (3.1) represents an imputed incremental cable cost when we route node i's demand to a concentrator at node j. Similarly, we account for the variable concentrator cost incurred when node j serves the demand at node i by including the third term $d_i\, c_j$ in the i-to-j homing cost $A_{ij}(\mu)$. The transformation in equation (3.1) thus converts all the cable expansion and concentrator variable costs into equivalent fixed assignment costs.

Note that the Lagrangian problem decomposes into two subproblems: an uncapacitated local access network expansion subproblem [ULAN1(μ)] containing only the x and y variables, and an easily solved cable expansion subproblem [CES(μ)] containing the s and z variables. The sum of the optimal values of these two Lagrangian subproblems provides a lower bound on the optimal cost of [LAN1]. We use subgradient optimization (see, for instance, Held, Wolfe, and Crowder [1974] or Fisher [1981]) to heuristically maximize the Lagrangian lower bound by finding a near-optimal set of Lagrange multipliers. We note that both our Lagrangian subproblems satisfy Geoffrion's [1970] integrality property (i.e., the linear programming relaxations of both subproblems have integer optimal solutions); therefore, the best possible Lagrangian lower bound cannot exceed the optimal value of the linear programming relaxation of formulation [LAN1]. We next discuss efficient methods to solve the two Lagrangian subproblems at each subgradient iteration.

3.1.1 The Cable Expansion Subproblem

Given a set of Lagrange multipliers $\{\mu_{ij}\}$, the *cable expansion subproblem,* denoted as *[CES(μ)]*, determines the optimal values of the cable installation and expansion variables, z_{ij} and s_{ij}, for all arcs <i,j>. This subproblem has the following form:

[CES(μ)]

$$\text{minimize} \quad \sum_{(i,j)\in T} G_{ij}\,(z_{ij} + z_{ji}) \;+\; \sum_{(i,j)\in T} (e_{ij} - \mu_{ij})\,(s_{ij} + s_{ji}) \tag{3.2}$$

subject to:

$$s_{ij} \le M_{ij}\, z_{ij}$$
$$s_{ji} \le M_{ji}\, z_{ji} \qquad \text{all } (i,j) \in T, \tag{3.3}$$
$$z_{ij} + z_{ji} \le 1 \qquad \text{all } (i,j) \in T, \text{ and} \tag{3.4}$$
$$z_{ij}, z_{ji} = 0 \text{ or } 1$$
$$s_{ij}, s_{ji} \ge 0 \qquad \text{all } (i,j) \in T. \tag{3.5}$$

Observe that this subproblem decomposes by edge, and is easy to solve. For each edge (i,j), we can solve the problem in two stages. First, we set:

$$s_{ij} = s_{ji} = 0 \qquad \text{if } (e_{ij} - \mu_{ij}) \ge 0, \text{ and}$$
$$s_{ij} = M_{ij}\, z_{ij},\ s_{ji} = M_{ji}\, z_{ji} \qquad \text{if } (e_{ij} - \mu_{ij}) < 0.$$

If we make these substitutions for s_{ij} and s_{ji}, the problem reduces to a formulation in only the installation variables z_{ij} and z_{ji}. The cost coefficients for z_{ij} and z_{ji} become:

$$G_{ij}(\mu) \equiv G_{ij} + M_{ij} \min\{e_{ij} - \mu_{ij}, 0\}, \text{ and}$$
$$G_{ji}(\mu) \equiv G_{ij} + M_{ji} \min\{e_{ij} - \mu_{ij}, 0\}.$$

If both these coefficients are nonnegative, we set $z_{ij} = z_{ji} = 0$. Otherwise, we set $z_{ij} = 1$ and $z_{ji} = 0$ if $G_{ij}(\mu) \le G_{ji}(\mu)$, and $z_{ij} = 0$ and $z_{ji} = 1$ if $G_{ji}(\mu) < G_{ij}(\mu)$. As a consequence, the objective function value of the subproblem [CES(μ)] is

$$\sum_{(i,j)\in T} \min\{0, G_{ij}(\mu), G_{ji}(\mu)\}.$$

Let us briefly indicate how to incorporate piecewise-linear, concave cable expansion cost functions (as shown in Figure 1) instead of the simple fixed plus linear cable expansion cost we have assumed so far. Suppose the cost function consists of M linear segments, with increasing intercepts G_{ijm} and decreasing slopes

e_{ijm} for m = 1,2,...M. To capture the different fixed and variable costs (G_{ijm} and e_{ijm}) for each segment, we replace z_{ij} and s_{ij} with disaggregate cable installation and expansion variables z_{ijm} and s_{ijm} for each range m = 1,2,...,M. The binary variable z_{ijm} is 1 if the solution chooses segment m on arc <i,j>, and 0 otherwise, while s_{ijm} denotes the amount of cable expansion (its value must lie in the range corresponding to the m^{th} segment). Accordingly, we modify the cable capacity constraints (2.5), the cable forcing constraints (2.6), and the arc orientation constraints (2.7) (the revised form of constraint (2.7) selects at most one segment in either direction on every edge). Because the cable expansion cost function is concave, we do not require any additional constraints (for example, variable lower bounds on the s_{ijm} variables). Clearly, these changes in the formulation affect only the cable expansion subproblem. For each edge (i,j) $\in$ T, this subproblem must now decide whether to set $z_{ijm} = s_{ijm} = 0$ for all m = 1,2,...,M (i.e., do not expand any cables) or select exactly one range m*, and set $z_{ijm^*} = 1$, $s_{ijm^*} = M_{ijm^*}$ for this range (with $z_{ijm} = s_{ijm} = 0$ for all $m \neq m^*$.) The optimal choice depends on the value of

$$G_{ijm}(\mu) = G_{ijm} + M_{ijm} \min \{e_{ij} - \mu_{ij}, 0\}.$$

If $G_{ijm}(\mu) \geq 0$ for all m = 1,2,...M, then $z_{ijm} = s_{ijm} = 0$ is the optimal solution to the cable expansion subproblem. Otherwise, we select the index m* with the most negative value of $G_{ijm}(\mu)$, and set $z_{ijm^*} = 1$, $s_{ijm^*} = M_{ijm^*}$. Thus, incorporating piecewise-linear, concave cable expansion cost functions in the Lagrangian subproblem is relatively easy.

3.1.2 Uncapacitated Local Access Network Expansion Subproblem

The uncapacitated local access network expansion subproblem, denoted as [ULAN1(μ)], determines the values of the assignment variables x_{ij} and the concentrator location variables y_j, assuming that the network does not contain any existing cables or concentrators (the model still restricts cable installation to the edges of the given tree network).

Let us consider a more general uncapacitated tree location (UTL) model with fixed concentrator costs (F_j), assignment costs (a_{ij}), and fixed cable installation costs (b_{ij}). Subproblem ULAN1(μ) is a special case of UTL with zero fixed cable installation costs. The dynamic programming algorithm we use to solve ULAN1(μ) applies to the general formulation (and, in fact, we use the more general model in Section 5.4 when we consider enhancements to improve the algorithm's performance). The uncapacitated tree location model has the following formulation:

[UTL]

$$\text{minimize} \quad \sum_{j \in N} F_j y_j + \sum_{i \in N} \sum_{j \in N} a_{ij} x_{ij} + \sum_{(i,j) \in T} b_{ij} (z_{ij} + z_{ji}) \qquad (3.6)$$

subject to:

Node assignment constraints (2.2),
Concentrator location constraints (2.3),
Contiguity restrictions (2.4),

Arc installation-forcing constraints:

$$x_{ij} \leq z_{ik} \quad \text{for all } i \in N, \text{ all } k \in P_{ij}, \qquad (3.7)$$

$$y_j, x_{ij} = 0 \text{ or } 1 \quad \text{for all } i, j \in N, \text{ and}$$

$$z_{ij} = 0 \text{ or } 1 \quad \text{for all } (i,j) \in T.$$

The arc installation-forcing constraints (3.7) capture the fixed costs b_{ij} for all edges $(i,j) \in T$. In the uncapacitated Lagrangian subproblem ULAN1(μ), the assignment cost a_{ij} equals $A_{ij}(\mu)$, as defined by equation (3.1), and $b_{ij} = 0$; so we can set every $z_{ij} = 1$, and remove the arc installation-forcing constraints.

3.2 Solving the Uncapacitated Local Access Network Planning Problem

To solve subproblem UTL for a tree with n nodes, we employ a dynamic programming algorithm that requires $O(n^2)$ operations. The method exploits and relies heavily on the tree structure and the contiguity property. This algorithm is a modest extension of methods previously proposed by Kariv and Hakimi [1979] for the p-median problem on a tree, and by Barany, Edmonds and Wolsey [1986] for optimally covering a tree by subtrees. (These two previous algorithms are themselves closely related and use a similar dynamic programming solution strategy.) We first describe the method's underlying principle before presenting it formally.

First, let us introduce some notation and conventions. We define the *level* of a node i as the number of edges lying on the path P_{i0} connecting that node to the switching center. Thus, the root node (node 0) has level 0, its immediate successors have level 1, and so on. For convenience, we index the nodes in increasing order of their levels. For notational simplicity, *we assume, without loss of generality, that each node of the given tree T either has exactly two successors or is a leaf node (i.e., the node has no successors).* Splitting nodes with more than two successors, or adding a dummy node to each single-successor node satisfies this assumption (and adds no more than n dummy nodes). For any node i ($i \neq 0$) in the tree, let p_i denote its *predecessor;* if node i is not a leaf node, let l_i be its *left successor,* and r_i its *right successor.* (By our node indexing convention, $p_i < i < l_i, r_i$.) Let *T(i)* denote the *subtree rooted at node i* when we delete edge (p_i,i) from tree T.

Starting with node n at the bottom of the tree, the dynamic programming algorithm sequentially considers nodes in decreasing order of node index, and hence in decreasing order of node levels. For each node i, the procedure recursively calculates the optimal total (cable expansion + concentrator + connection) *cost of covering all nodes within subtree T(i)* (i.e., serving the subtree's demand) using only concentrators that are located within this subtree. We denote this total cost as *TC(i)* (*TC* denotes *Tree Cost*); it includes the cost of locating concentrators within the subtree T(i), the cost of assigning all the nodes in the subtree to their respective concentrators, and the cost of expanding cables to provide adequate transmission capacity between each node and its assigned concentrator. The value TC(0), corresponding to the switching center (i.e., the root node 0), gives the optimal cost of UTL.

To calculate TC(i), we must first determine where node i should home within its subtree T(i). Let *HC(i,j)*, standing for homing cost or HC, denote the *cost of covering all nodes within subtree T(i), assuming node i homes on some node j* (for this definition, node j need not necessarily belong to T(i)). Then,

$$TC(i) = \underset{j \in T(i)}{\text{minimum}} HC(i,j), \qquad (3.8)$$

i.e., the cost TC(i) of covering all nodes in subtree T(i) as a stand-alone tree is the smallest value of HC(i,j) over all homing nodes within T(i). Within the dynamic programming algorithm, we will iteratively (from the bottom of the tree to the top) compute the HC(i,j) values. To calculate HC(i,j) we exploit the following two observations, both stemming from the contiguity property:

Observation 1: *Converting fixed concentrator and cable costs to Homing costs*
The UTL model has three types of costs: fixed assignment costs (a_{ij}), fixed concentrator costs (F_j), and fixed cable installation costs (b_{ij}). We can simplify this cost structure by including the fixed cable and concentrator costs in appropriate homing costs. First, if we assume all fixed cable costs are positive, then the contiguity property implies that any optimal solution to UTL installs cable on arc $<i,k_{ij}>$ if and only if node i homes on node j (recall that k_{ij} is node i's neighbor on the path P_{ij} connecting node i to node j). Thus, we can include the fixed arc cost $b_{ik_{ij}}$ in the i-to-j homing cost. Since we assign node i to exactly one node, we avoid double counting the fixed arc costs. Similarly, if concentrator costs are positive, the optimal solution installs a concentrator at node j if and only if some node homes on node j; and, by contiguity, node j must home on itself before any other node can home on it. Thus, we can include the fixed concentrator cost F_j in the j-to-j homing cost.

Because of these two simplifications, UTL contains only a single cost type, namely, an ***adjusted i-to-j homing cost*** d_{ij} for each node pair i,j that is defined as follows:

$$d_{ij} = \begin{cases} a_{ij} + b_{ik_{ij}} & \text{if } i \neq j, \text{ and} \\ a_{jj} + F_j & \text{if } i = j. \end{cases} \quad (3.9)$$

The total cost of any uncapacitated tree expansion solution equals the sum of the adjusted homing costs for all the node assignments selected by that solution. Recall that, for the local access subproblem ULAN1(μ), the assignment cost a_{ij} in (3.9) equals $A_{ij}(\mu)$ and includes the connection cost, variable concentrator cost, and variable cable expansion cost (see equation (3.1)). As we show next, transforming all the fixed and variable cable and concentrator costs into node-to-concentrator homing costs for the uncapacitated problem enables us to decompose and recursively compute the tree covering costs TC(i) and HC(i,j).

Observation 2: *Decomposition of Tree Covering costs*
If node i homes on node j, the contiguity property restricts the possible homing patterns for node i's successors. In particular, consider the possible homing node for the left successor l_i of node i. If node j belongs to the *left subtree* $T(l_i)$, then by the contiguity property, node l_i must also home on node j since path P_{ij} contains l_i; in this case, we must include the (previously computed) cost $HC(l_i,j)$ in HC(i,j). Otherwise, (if $j \notin T(l_i)$) node l_i must either home on node j or on an internal node within its own subtree $T(l_i)$. In this case, the best choice for l_i's homing node depends on whether $HC(l_i,j)$ is less than $TC(l_i)$ (the optimal cost of covering all nodes in $T(l_i)$ using only internal concentrators). We can apply a similar argument to determine the best homing node for the right successor r_i: if $j \in T(r_i)$, node r_i must also home on j and we include the value of $HC(r_i,j)$ in HC(i,j); otherwise, r_i homes on j or on an internal node within the *right subtree* $T(r_i)$ depending on whether $HC(r_i,j)$ is smaller than or exceeds $TC(r_i)$.

This decomposition of homing patterns suggests that we can recursively calculate HC(i,j) for intermediate nodes i using the following equations:

$$HC(i,j) = d_{ij} + \min\{HC(l_i,j), TC(l_i)\} + \min\{HC(r_i,j), TC(r_i)\} \quad \text{if } j=i \text{ or } j \notin T(i), \quad (3.10a)$$

$$HC(i,j) = d_{ij} + HC(l_i,j) + \min\{HC(r_i,j), TC(r_i)\} \quad \text{if } j \in T(l_i), \text{ and} \quad (3.10b)$$

$$HC(i,j) = d_{ij} + \min\{HC(l_i,j), TC(l_i)\} + HC(r_i,j) \quad \text{if } j \in T(r_i). \quad (3.10c)$$

If node i is a leaf node of T, HC(i,j) equals d_{ij}.

Equations (3.8) and (3.10) are the underlying recursive equations for the UTL dynamic programming algorithm. To ensure that the required quantities on the right-hand side of equations (3.10) are available when needed, we must properly sequence the computations of HC(i,j). As mentioned before, the decreasing order of node indices (and hence decreasing level of the nodes) defines an appropriate bottom-to-top sequence.

The DP algorithm, thus, consists of (n+1) stages, one corresponding to each node k in the tree. At stage k, for k = n, (n-1), ..., 0, we consider the subtree T(k) rooted at node k. In this stage, we first compute HC(i,j) for every node pair i,j $\in$ T(k) *that communicate via node k*, i.e., with either nodes i and j lying in opposite subtrees T(l_k) and T(r_k), or i = k and j $\in$ T(k), or i $\in$ T(l_k) $\cup$ T(r_k) and j = k. To compute the values of the homing costs HC(i,j), we consider nodes i in order of decreasing index; we can consider target nodes j in any order. At the end of stage k, the procedure has computed HC(i,j) for all i,j $\in$ T(k). Using this information, we can apply equation (3.8) to calculate TC(k), the optimal cost of serving all nodes of subtree T(k) using only internal concentrators belonging to this subtree. The final value TC(0) that we calculate at stage 0 gives the optimal value of the UTL problem. (By convention, at the final stage k = 0, we assume that node 0 can home only on itself, i.e., the connection cost A_{0j} is set equal to a very large value for all j $\neq$ 0.) The usual dynamic programming backtracking procedure gives the optimal concentrator location and node assignment strategy. A formal description of the dynamic programming algorithm follows:

DP Algorithm for the Basic UTL model: [DP1]

For k = n,n-1,...,0

if k is a leaf node,

HC(k,j) $\leftarrow$ d_{kj} for all j $\in$ N

TC(k) $\leftarrow$ HC(k,k) = d_{kk}

else

for i = n, n-1,...,k+1, with i $\in$ T(k)

for j = n, n-1,...,k, with j $\in$ T(k) and k $\in$ P_{ij}

compute HC(i,j) using equation (3.10a)

next j;

next i;

for j = n, n-1,...,k-1, with j $\in$ T(k)

compute HC(k,j) using equation (3.10b) if j $\in$ T(l_k)

compute HC(k,j) using equation (3.10c) if j $\in$ T(r_k)

next j;

compute HC(k,k) using equation (3.10a)

$$TC(k) \leftarrow \underset{j \in T(k)}{\text{Minimum}}\ HC(k,j)$$

next k;

The method examines each node pair i,j exactly once to calculate HC(i,j). Since each HC(i,j) calculation requires constant time, the overall computational complexity of the dynamic programming algorithm is $O(n^2)$ operations. We note that, after our cost transformation (3.9), the dynamic program is identical to Barany et al.'s [1986] method for covering a tree with subtrees. Their method, in turn, could be viewed as an adaptation of Kariv and Hakimi's [1979] p-median algorithm. In the UTL context, if $|N(i)|$ is the number of nodes in the rooted subtree T(i), the Kariv-Hakimi algorithm recursively evaluates the tree cost value TC(i,v) for $v = 1, 2, \ldots, |N(i)|$ which is defined as the total cost of covering all nodes in T(i) using exactly v internal concentrators. Thus, we can use the Kariv-Hakimi algorithm to solve UTL by setting:

$$TC(i) = \min \{TC(i,v): v = 1, 2, \ldots, |N(i)|\}.$$

Note that this algorithm requires $O(n^2p)$ operations.

Next, we briefly outline the modest modifications required to the UTL solution method to incorporate multiple concentrator technologies, and so the piecewise-linear, concave concentrator cost function shown in Figure 1.

As before, let M denote the number of linear segments in the piecewise-linear concentrator cost function. Conceptually, we view the piecewise-linear cost function as the lower envelope of a set of M fixed plus linear cost functions, each corresponding to a different concentrator technology. (In reality, different segments might model the same physical technology, say, multiplexers but with different fixed and variable costs). Let F_{jm} and c_{jm} denote, respectively, the fixed cost (or intercept) and variable cost (or slope) of the m^{th} technology for m = 1,2,...,M. F_{jm} increases and c_{jm} decreases as m increases; consequently, when we minimize total cost, the model will automatically select the correct technology corresponding to the required concentrator throughput.

To account for this concave concentrator cost function, we model the problem with M alternative concentrators at every node j, each with different fixed and variable costs. Correspondingly, we define M different assignment costs $A_{ijm}(\mu)$ (using the variable cost c_{jm} instead of c_j in equation (3.1)) for every pair of nodes $i, j \in N$; we, therefore, obtain M adjusted homing costs d_{ijm}. For every node j, the adjusted homing cost d_{jjm} includes the fixed cost F_{jm} of the type m concentrator (see

equation (3.9)). At each stage, the DP algorithm now recursively computes HC(i,j,m), the total cost of covering all nodes of T(i), using equation (3.10) assuming node i homes on a type m concentrator at node j, for m = 1,2,...,M. We now define TC(i) as min {HC(i,j,m): $j \in T(i)$, m = 1,2,...,M}. As before, TC(0) gives the optimal value of UTL. If M is the maximum number of segments in the concentrator cost function over all nodes of the network, these enhancements increase the computational complexity of the DP algorithm to $O(n^2M)$.

3.3 Lagrangian-based Heuristic

By iteratively adjusting the Lagrange multipliers using subgradient optimization (see, for example, Fisher [1981]), and solving the Cable Expansion and Uncapacitated Tree Location Lagrangian subproblems at each iteration, we generate lower bounds on the optimal network expansion cost. The Lagrangian relaxation scheme also generates good starting solutions for a heuristic improvement procedure. Our Lagrangian-based heuristic procedure uses the optimal solution to the uncapacitated tree location subproblem to generate a feasible starting solution. The method then iteratively improves this starting solution by interchanging node assignments.

Recall that the UTL subproblem determines the optimal node-to-concentrator assignments (i.e., values for the x_{ij} variables) and the corresponding concentrator location strategy (y_j variables), assuming that the network contains no existing cable capacities. To obtain a heuristic expansion plan, we complete the UTL solution by determining the actual number of additional cables required to accomodate the selected node-to-concentrator assignments.

We then apply a myopic improvement strategy called the *Greedy Reassignment Heuristic* to the starting solution. This method attempts to iteratively reassign nodes to concentrators, one at a time, without violating the contiguity condition. Note that, to preserve contiguity at every stage, each node has at most three alternate homing nodes, i.e., the homing nodes of its neighbors. At each iteration, the greedy heuristic: (i) evaluates the cost impact of all "feasible" changes in node-to-concentrator assignments (i.e., those that preserve the contiguity property); and, (ii) performs the best change, i.e., the reassignment that gives the greatest reduction in total cost (concentrator + cable expansion cost). If all feasible reassignments increase total cost, the local improvement procedure terminates.

To reduce computational time, instead of improving the Lagrangian-based starting solution at every iteration of the subgradient optimization solution procedure, our implementation applies the greedy method only intermittently (e.g., when the current starting solution has lower cost than the previous best starting

solution). We also use the greedy heuristic to generate an **initial upper bound,** before initiating the subgradient procedure. For this purpose, we consider two different starting solutions—the *centralized* solution that homes all demand nodes on the root node (ie., this solution employs only cable expansion to satisfy projected demand), and the *distributed* solution that locates a concentrator at each node. After applying the greedy reassignment heuristic to each of these two solutions, we choose the solution with the lower cost as the initial upper bound.

In summary, this section has described the algorithm to generate upper and lower bounds for the basic problem formulation [LAN1]. The next two sections describe methods for improving the lower bounds in several ways: through problem preprocessing, tightening the constraints of formulation [LAN1], and adding new inequalities to strengthen the Lagrangian relaxation.

4. Modeling and Algorithmic Enhancements I: Variable Elimination and Coefficient Reduction

Our preliminary computational experience with the Lagrangian relaxation algorithm of Section 3 for the basic [LAN1] model suggested that, while the heuristic method generates very good solutions, the Lagrangian lower bounds are very weak, as evidenced by large gaps between the upper and lower bounds on the optimal objective value. Section 6.2 summarizes these computational results. To improve the lower bounds, we developed various modeling and algorithmic enhancements. This section describes two types of improvements: problem preprocessing to eliminate certain assignment variables, and reducing the values of the cable expansion bounds in order to tighten the forcing constraints (2.6) in formulation [LAN1].

4.1 Variable Elimination by Problem Preprocessing

To reduce the size of problem [LAN1], we employ a tradeoff analysis that identifies, based on the given problem data (i.e., demands, costs, and capacities), node-to-concentrator assignments that do not occur in an optimal solution. We prohibit such assignments by eliminating the corresponding assignment variables x_{ij} from the problem formulation. This variable elimination process not only reduces the problem size, and hence the computational effort, but might also improve the lower and upper bounds.

For convenience, we will assume in the following discussion that, for each node i, the connection cost A_{ij} is the same for all homing nodes j. The preprocessing method extends easily to problems with varying connection costs. To identify suboptimal node-to-concentrator assignments, we employ the following principle:

Consider any node pair i,j, and suppose we wish to prove, if possible, that node i does not home on node j in an optimal expansion plan. For this purpose, we compute a *lower bound* L_{ij} on the incremental cost of assigning node i to node j. (By incremental cost, we mean the cable expansion and concentrator cost that can be attributed solely to the i-to-j asssignment.) We then compare this lower bound to an upper bound U_{ii} on the cost of locating a concentrator at node i (and homing node i on this concentrator). If $L_{ij} > U_{ii}$, the i-to-j assignment is provably suboptimal, and we can eliminate the assignment variable x_{ij} from formulation [LAN1]. Next, we describe a simple method to estimate L_{ij} and U_{ii}.

The lower bound L_{ij} on the incremental cost of assigning node i to node j consists of two components: an incremental *cable expansion* cost, and an incremental *concentrator* cost. To calculate the incremental cable expansion cost, consider the path P_{ij} connecting node i to node j, and let <k,*l*> be any arc on this path. By the contiguity property, if node i homes on node j, then every node on the subpath P_{ik} (from i to k) must also home on node j. Thus, the total demand, say, D_{ik} of all nodes on the path P_{ik} (including node i's and node k's demand) must flow through arc <k,*l*> if i homes on j. If D_{ik} is less than or equal to B_{kl}, the existing capacity of edge (k,*l*), then we do not incur any cable expansion cost on edge (k,*l*). Otherwise, we incur both a fixed cost G_{kl} and a per unit cost e_{kl} for each unit of flow exceeding B_{kl}. Let $\phi_{kl} = \text{Max}\ \{ [D_{ik}-B_{kl}], 0\}$ denote the excess flow on edge (k,*l*). Then, node i's demand contributes a cost of at least $\delta_{kl} = e_{kl}\ \text{Min}\{\phi_{kl}, d_i\}$ to the total cable cost on edge (k,*l*). Adding the incremental costs δ_{kl} for all edges (k,*l*) of path P_{ij} gives the cable expansion cost component of the lower bound L_{ij}. The concentrator cost component is equal to $c_j\, d_i$, the demand at node i times the variable concentrator c_j at node j. Thus, the lower bound L_{ij} on the incremental cost of assigning node i to node j is

$$L_{ij} \quad = \quad \sum_{(k,l)\in P_{ij}} \delta_{kl} \; + \; c_j\, d_i \qquad \text{for all } i,j \in N. \tag{4.1}$$

If node i is a leaf node of T, and if the current capacity of the edge (i,k_{ij}) (incident to node i on the path P_{ij}) is less than node i's demand, we can improve the lower bound L_{ij} by adding the fixed cost $G_{i,k_{ij}}$ to the right-hand side of (4.1).

The upper bound U_{ii} on the incremental cost when node i homes on itself is easy to compute; we set

$$U_{ii} \quad = \quad F_i \; + \; c_i\, d_i \qquad \text{for all } i \in N. \tag{4.2}$$

If $L_{ij} > U_{ii}$, we prove suboptimality of the i-to-j assignment by contradiction. Consider any optimal expansion plan, and suppose node i homes on node j in this plan. First, consider the case when node i is a leaf node of the subtree t_j induced by the nodes that the concentrator at node j serves. Clearly, canceling the i-to-j

assignment will save at least L_{ij}, while installing a concentrator at node i (and homing node i on this concentrator) will cost exactly U_{ii}. Thus, if $L_{ij} > U_{ii}$, we can improve the given solution, contradicting its optimality. Now, suppose that node i is an interior node of subtree t_j. In the given solution, let N(i,j) be the subset of nodes (including node i) assigned to node j that communicate via node i. Observe that if we cancel the k-to-j assignment for every node $k \in N(i,j)$, the total saving in cable expansion and concentrator costs must be at least $|N(i,j)| L_{ij}$, while $|N(i,j)| U_{ii}$ overestimates the incremental cost of homing all these nodes to a new concentrator at node i. (Here, we exploit the concavity of concentrator costs.) Thus, if $L_{ij} > U_{ii}$, we can again improve the given solution, implying that this solution is not optimal.

This preprocessing technique of prohibiting all i-to-j assignments with $L_{ij} > U_{ii}$ extends easily to piecewise-linear, concave cost functions for cable expansion and concentrators. The preprocessing method not only reduces the problem size (e.g., the number of variables in the integer programming formulation) but also strengthens it by decreasing the maximum possible flows through certain arcs <i,j>, thus reducing the cable expansion bound M_{ij}. Next, we describe another method to reduce this parameter.

4.2 Tightening the Cable Forcing Constraints by Coefficient Reduction

To increase the relaxation lower bounds, we first attempted to improve formulation [LAN1] by tightening the cable installation-forcing constraints (2.6a) and (2.6b). Recall that these forcing constraints relate the cable installation and cable expansion variables, and have the following form:

$$\begin{aligned} s_{ij} &\leq M_{ij} z_{ij}, \text{ and} \\ s_{ji} &\leq M_{ji} z_{ji} \end{aligned} \qquad \text{for each edge } (i,j) \in T.$$

In Section 2.3, we showed how to find valid values for the cable expansion bounds M_{ij} and M_{ji} by considering the total demand that can enter or leave the subtree formed by deleting edge (i,j). Let us refer to these bounds as *demand-based* cable expansion bounds, and denote them as M_{ij}^d and M_{ji}^d.

Since these demand-based bounds represent the worst-case cable expansion requirements, their values are typically much larger than the actual flows routed on each arc. Consequently, the cable installation variables z_{ij} often take small fractional values in the optimal linear programming (or Lagrangian) solution. Our coefficient reduction method uses a tradeoff between concentrator costs and cable expansion costs to identify a tighter *cost-based* upper limit M_{ij}^c on cable expansion on every arc <i,j>. We then set the right-hand side coefficient M_{ij} in the forcing constraint (2.6a) equal to min $\{M_{ij}^d, M_{ij}^c\}$.

We obtain the cost-based upper limit M_{ij}^c by comparing the cable expansion cost on arc <i,j> with the concentrator cost at node i. Assume, for simplicity, that the connection cost A_{kl} is the same for all homing nodes l. Suppose an optimal expansion plan routes f_{ij} units of traffic from i to j, and suppose $f_{ij} > B_{ij}$, edge (i,j)'s current cable capacity. This solution incurs an expansion cost of at least $CE_{ij}(f_{ij}) = \{G_{ij} + (f_{ij}-B_{ij})*e_{ij}\} + f_{ij}\, c_{min}$; in this expression, c_{min} is the smallest variable concentrator cost taken over all nodes at or beyond node j. Consider the alternate solution obtained by installing a concentrator at node i, and rehoming all the traffic that previously flowed through arc <i,j> on this concentrator. This solution incurs a concentrator cost of $CC_i(f_{ij}) = \{F_i + f_{ij}*c_i\}$. Clearly, if $CE_{ij}(f_{ij})$ exceeds $CC_i(f_{ij})$, then installing a concentrator at node i improves the given solution.

Let U_{ij} denote the flow value at which the cost functions $CE_{ij}(f)$ and $CC_i(f)$ intersect (assuming $G_{ij} < F_j$ and $e_{ij} > c_i$), i.e., for flow values above the threshold value U_{ij}, $CE_{ij}(f)$ exceeds $CC_i(f)$. Figure 2 shows this intersection. Our previous argument implies that routing more than U_{ij} units of flow on arc <i,j> is suboptimal. Thus, the cost-based cable expansion bound is $M_{ij}^c = (U_{ij} - B_{ij})$. Similarly, by comparing the cost of locating a concentrator at node j with the cost of expanding arc <j,i>, we can compute a cost-based bound M_{ji}^c and strengthen the forcing constraint (2.6b). With minor modifications, the method to calculate cost-based upper limits also extends to problems with varying node-to-concentrator connection costs, and piecewise-linear, concave cable expansion and concentrator costs.

5. Modeling and Algorithmic Enhancements II: Incorporating Valid Inequalities

This section describes further modeling and algorithmic enhancements to improve the performance of our Lagrangian-based solution method for the local access network expansion problem. We obtain these improvements by adding certain valid inequalities, or cuts, to the original problem formulation so that we reduce the feasible region for the Lagrangian (and linear programming) relaxation of the integer formulation without eliminating the optimal integer solution. Since the relaxation for the improved formulation has a smaller feasible region, the lower bound it provides might possibly be better than the basic model (i.e., it might have a larger optimal value). For a review of some of the basic ideas and successful applications of this "polyhedral combinatorics" solution approach, the reader might consult Hoffman and Padberg [1985] or Nemhauser and Wolsey [1988].

In Balakrishnan et al. [1991b], we identified several classes of valid inequalities for the local access network expansion problem, and proved results concerning their polyhedral properties. In particular, under certain conditions, these inequalities constitute facets of the integer programming polytope. In this paper, we focus on a subset of our original valid inequalities; the inequalities that we chose were easy to incorporate (with some modifications) in our dynamic programming algorithm for the Lagrangian subproblem. These constraints attempt to relate the assignment variables x_{ij} and the cable installation and expansion variables (z_{ij} and s_{ij}) without appreciably increasing the computational complexity of the Lagrangian subproblem. Adding these valid inequalities to the problem formulation results in a single, comprehensive Lagrangian subproblem that simultaneously determines homing assignments, concentrator locations, and cable additions.

We first describe and motivate (in Sections 5.1 to 5.3) the three classes of valid inequalities that we implemented. Section 5.4 describes requisite modifications to the dynamic programming method needed to accomodate these three types of inequalities. Our computational experience indicates that the subset of valid inequalities that we chose is very effective in reducing the gap between the Lagrangian lower and upper bounds.

Throughout this discussion, recall that T(i) is the subtree of our given tree T rooted at node i. Let D_i denote the total demand in subtree T(i) (summed over all nodes in T(i)). We let p_i denote the predecessor of node i, and if node i is not a leaf node, we let r_i and l_i denote its left and right successors. Recall that, by adding dummy nodes if necessary, we are assuming that all intermediate nodes (i.e., nodes other than the root and leaf nodes) have exactly two successors.

5.1 Assignment-forcing Arc Installation Inequalities

Our first class of valid inequalities exploits the contiguity property to relate the assignment variables x_{ij} to the binary cable installation variables z_{ij}. The inequalities are based on the following observation: Assuming that arc expansion costs are positive, any optimal local network expansion plan expands arc $<i,p_i>$ only if node i homes on some node $l \notin$ T(i) via its predecessor p_i. Similarly, a plan that expands arc $<p_i,i>$ must assign node p_i to some concentrator within subtree T(i). This relationship between the assignment and cable installation variables motivates the following 2(n-1) ***assignment-forcing arc installation inequalities***:

$$\sum_{l \notin T(i)} x_{il} \quad \geq \quad z_{ip_i}, \text{ and} \tag{5.1a}$$

$$\sum_{l \in T(i)} x_{jl} \quad \geq \quad z_{p_i i} \qquad \text{for all } (i,j) \in T. \qquad (5.1b)$$

Balakrishnan et al. [1991b] generalize these constraints to cutsets of the tree T.

5.2 Bottleneck-Arc Installation and Expansion Inequalities

Our next class of inequalities relate the concentrator location decisions to the cable installation and expansion decisions. We refer to arc $<i,p_i>$ as a ***BOTTLENECK ARC*** if the total demand D_i for all nodes in subtree T(i) exceeds the arc's current capacity B_{ip_i}.

Let I_B be the set of *bottleneck nodes*, i.e., for each node $i \in I_B$, the total demand in the corresponding rooted subtree T(i) exceeds the capacity of the incident arc $<i,p_i>$. We can then add the following valid bottleneck-arc installation and expansion inequalities to the problem formulation:

$$\sum_{k \in T(i)} y_k + z_{ip_i} \quad \geq \quad 1 \qquad \text{for all } i \in I_B, \text{ and} \qquad (5.2)$$

$$(D_i - B_{ip_i}) \{ \sum_{k \in T(i)} y_k \} + s_{ip_i} \quad \geq \quad D_i - B_{ip_i} \qquad \text{for all } i \in I_B. \qquad (5.3)$$

Constraint (5.2) states that the expansion plan must either install at least one concentrator within the subtree T(i) or expand arc $<i,p_i>$ (or both). Constraint (5.3) forces the amount of cable expansion s_{ip_i} to be at least $(D_i - B_{ip_i})$ if T(i) does not contain any concentrator. (Since the cable installation-forcing constraint (2.6) of the original formulation [LAN1] imposes an upper bound of $M_{ip_i} = D_i - B_{ip_i}$ on s_{ip_i}, if T(i) has no concentrators, constraints (5.3) and (2.6) together ensure that $s_{ip_i} = D_i - B_{ip_i}$.) Observe also that, if arc $<i,p_i>$ is *not* a bottleneck arc, then we can eliminate the installation and expansion variables z_{ip_i} and s_{ip_i} from the problem formulation.

Stronger version of Bottleneck-Arc Installation Inequalities

If node i is a bottleneck node, and its rooted subtree T(i) does not contain any concentrator, constraint (5.2) forces the solution to expand only the incident arc $<i,p_i>$. However, if T(i) does not contain any concentrators, then neither does any subtree T(k) within T(i), for all nodes $k \in T(i)$. Therefore, we must also expand *all* other bottleneck arcs $<k,p_k>$ within subtree T(i). Let $A_B(i)$ denote the set of bottleneck arcs within subtree T(i); for notational convenience, we also include arc $<i,p_i>$ in this set. We then have the enhanced bottleneck-arc installation inequalities:

$$\sum_{k \in T(i)} y_k + \{ \sum_{(k,p_k) \in A_B(i)} z_{kp_k} / |A_B(i)| \} \geq 1 \quad \text{for all } i \in I_B. \qquad (5.4)$$

Observe that aggregating the constraints (5.2) for all arcs $<k,p_k> \in A_B(i)$ gives a weaker constraint than (5.4). Balakrishnan et al. [1991b] generalize constraints (5.4) to apply to any arbitrary (bottleneck) subtree of T.

5.3 Subtree-splitting Arc Installation and Expansion Inequalities

The bottleneck-arc installation and expansion inequalities exploit one opportunity for determining the exact arc flow, namely, if T(i) does not contain any concentrator, then all its nodes home outside T(i), and the total flow on arc $<i,p_i>$ must *exactly equal* the total demand D_i within that subtree. To identify additional valid inequalities, we consider the following extension of the bottleneck-arc inequalities' underlying principle. Suppose, instead of knowing the exact flow value, we can compute either upper or lower bounds on the flow on arc <i,j> for certain assignment and concentrator location patterns. If the upper bound (lower bound) is less (more) than arc <i,j>'s capacity B_{ij}, then we can specify that solutions that contain the special patterns *do not* (*do*) expand arc <i,j>. This principle suggests several new classes of valid inequalities which we call the *Subtree-splitting Arc Installation* and *Expansion Inequalities*. This class of inequalities exploits the natural decomposition of homing patterns (stemming from the contiguity property) used in the dynamic programming algorithm for the UTL subproblem. The decomposition splits each subtree into its left and right components, motivating the name for this constraint class.

For any intermediate node k of T, let $dmin_k$ denote the ***smallest nodal demand among all leaf nodes of the rooted subtree T(k)***. Observe that, if T(k) contains one (or more) concentrator, the concentrator throughput must be at least $dmin_k$ since the non-bifurcated routing assumption (assumption A2) implies that the concentrator must serve the entire demand for at least one node, and by the contiguity property, at least one leaf node homes on this concentrator. Consequently, at most $(D_k - dmin_k)$ units of traffic can flow out of subtree T(k).

Consider some intermediate node i. We wish to generate upper and lower bounds on the flow emanating from this node for various possible homing patterns. Let k be one of the three nodes, p_i, r_i, or l_i, adjacent to node i. We will denote the ***upper and lower bounds on flow*** on arc <i,k> as $fmax_{ik}$ and $fmin_{ik}$, respectively.

To systematically consider various possible node assignment and concentrator location decisions, we first broadly partition the alternative homing patterns into four cases:

Case 1: Node i homes on a node j outside subtree T(i);
Case 2: Node i homes on a node within its right subtree $T(r_i)$;
Case 3: Node i homes on a node within its left subtree $T(l_i)$; and,
Case 4: Node i homes on itself, i.e., node i contains a concentrator.

The upper and lower bounds on arc flow for Case 2 also apply to Case 3; and, for Case 4, the flow on all arcs emanating from node i must be zero. Our subsequent discussions will, therefore, focus on Cases 1 and 2.

We use the contiguity property to further partition the possible homing patterns in each case. For example, consider the situation, Case 1, in which node i homes on a node j that is outside the rooted subtree T(i). By contiguity, each successor of node i, say, r_i must either home on node j or home on a node within its rooted subtree $T(r_i)$. This type of observation leads to the subcase entries given in Table 1 which are a natural partition of the the possible homing patterns in subtree T(i).

We next illlustrate a representative case for computing upper and lower bounds on the flow from node i. Table 1 summarizes the bounds for all the possible cases.

Case 1.2.2: Node i homes on a node j outside T(i); r_i and l_i also home on node j; $T(r_i)$ does not contain any concentrator; $T(l_i)$ contains at least one concentrator.

In this case, we wish to compute upper and lower bounds ($fmax_{ip_i}$ and $fmin_{ip_i}$) on the flow on arc $<i,p_i>$. First, observe that arc $<i,p_i>$ must carry at least the demands at nodes i, r_i, and l_i. Furthermore, since $T(r_i)$ does not contain any concentrator, the demands for all nodes within $T(r_i)$ must also necessarily flow via arc $<i,p_i>$ (since all these nodes also home on node j, outside T(i), by the contiguity property). Thus, the lower bound for the flow on arc $<i,p_i>$ is

$$fmin_{ip_i} = d_i + d_{l_i} + D_{r_i}. \qquad (5.5a)$$

(Recall that D_{r_i} is the total demand for all nodes within the right subtree $T(r_i)$, and includes the demand at node r_i.) To obtain an upper bound $fmax_{ip_i}$ on the flow, we observe that the throughput of the concentrator(s) within the left subtree $T(l_i)$ must be at least $dmin_{l_i}$ (the smallest leaf node demand in subtree $T(l_i)$). Therefore, at most $(D_{l_i} - dmin_{l_i})$ units can flow out of $T(l_i)$ and via arc $<i,p_i>$. Thus,

$$fmax_{ip_i} = d_i + D_{r_i} + (D_{l_i} - dmin_{l_i}). \qquad (5.5b)$$

Using similar arguments, we can determine the upper and lower bounds shown in Table 1 for the remaining cases. Notice that, because we have expressed all the bounds in terms of the given node demands, we can compute the values of $fmax_{ik}$ and $fmin_{ik}$ at the outset, prior to solving the problem.

Thus far we have discussed bounds on the flows and homing patterns for only intermediate nodes. For leaf nodes, we need to consider only two cases:

(i) node i homes on itself: The flow on arc $<i,p_i>$ must be zero in this case; therefore, $fmax_{ip_i} = fmin_{ip_i} = 0$; or

(ii) node i homes on a downstream node: Arc $<i,p_i>$ carries exactly d_i units in this case, i.e., $fmax_{ip_i} = fmin_{ip_i} = d_i$.

Suppose, using Table 1 and our previous observations, we calculate the upper and lower bounds ($fmax_{ik}$ and $fmin_{ik}$) on the flow on every arc $<i,k>$ for each homing pattern. Then, depending on the values of $fmax_{ik}$ and $fmin_{ik}$ relative to the arc capacity B_{ik}, we can impose one of the following three conditions whenever the optimal solution selects the corresponding homing pattern:

(i) if $B_{ik} < fmin_{ik}$, (Arc $<i,k>$ is *IN*)
$z_{ik} = 1$, and
$\max\{0, fmin_{ik}-B_{ik}\} \le s_{ik} \le \min\{M_{ik}, fmax_{ik}-B_{ik}\}$; and, (5.6a)

(ii) if $B_{ik} \ge fmax_{ik}$, (Arc $<i,k>$ is *OUT*)
$z_{ik} = s_{ik} = 0$; and, (5.6b)

(iii) if $fmin_{ik} < B_{ik} < fmax_{ik}$, (Arc $<i,k>$ is *FREE*)
$z_{ik} = 0$ or 1, and
$s_{ik} \le \min\{M_{ik}, fmax_{ik}-B_{ik}\}$. (5.6c)

We could mathematically represent these additional conditions as a set of valid inequalities relating the assignment decision variables x_{ij}, x_{r_ij}, and x_{l_ij}, the concentrator location variables y_k for all $k \in T(l_i)$ and $T(r_i)$, and the cable installation and expansion variables z_{ip_i} and s_{ip_i}. These constraints strengthen the original problem formulation, thus potentially improving the Lagrangian (and LP) lower bounds. We will not attempt to formulate and exhaustively list all the subtree-splitting constraints since our dynamic programming approach does not require an explicit mathematical representation of these cuts. Instead, as shown in Section 5.4, we modify the algorithm so that it implicitly accounts for the inequalities in the recursive calculations at each stage.

We conclude this section with a final note about the subtree-splitting inequalities. These constraints generalize the bottleneck-arc inequalities by distinguishing homing patterns based upon whether each subtree in the next level below node i (i.e., the left and right subtrees) contains at least one concentrator. We can further refine this partition and obtain sharper valid inequalities by examining the number of concentrators within subtrees that are two levels, three levels, and so on below node i. However, incorporating these inequalities in the dynamic program adds to the algorithmic complexity of the solution approach. Our implementation incorporates only the first level subtree-splitting inequalities.

5.4 Modifying the Dynamic Programming Algorithm to Accomodate the Valid Inequalities

Adding the three classes of inequalities—the assignment-forcing inequalities, the bottleneck-arc installation and expansion constraints, and the subtree-splitting cuts—to the original formulation introduces additional linkages between the assignment, concentrator location, and cable addition variables. Consequently, when we dualize the cable capacity constraints (2.5) using multipliers $\{\mu_{ij}\}$, the resulting Lagrangian subproblem, which we denote as [ULAN2(μ)], combines the previous uncapacitated network expansion problem [ULAN1(μ)] and the cable expansion subproblem [CES(μ)]. This section and Appendix 1 describe how we modify the UTL dynamic programming approach [DP1] described in Section 3.2 to solve the new, integrated Lagrangian subproblem.

To incorporate the new valid inequalities, we exploit the dynamic program's capability to account for arc fixed costs. Recall, from Section 3, how we transformed cable and concentrator fixed and variable costs into equivalent, adjusted homing costs d_{ij} for every node pair $i,j \in N$. In particular, we argued that, for the uncapacitated problem, the optimal solution must install arc $<i,k_{ij}>$ if it asssigns node i to a concentrator at node j. Consequently, we included the arc fixed cost $b_{ik_{ij}}$ in the adjusted homing cost d_{ij} (see equation (3.9)). Observe that this transformation automatically satisfies the assignment-forcing cable installation inequalities (5.1), i.e., the Lagrangian subproblem solution does not install arc <i,k> if node i does not home via node k.

How do we select a value for the arc fixed cost b_{ik}? To address this issue, let us first understand the common underlying strategy for the bottleneck-arc inequalities and the subtree-splitting constraints. Observe that both these classes of inequalities attempt to use the demand parameters to determine if arc <i,k> must necessarily be ***included*** or ***excluded*** from the design, for various homing patterns. Thus, for a given homing pattern, one of the three cases described in (5.6a), (5.6b) or

(5.6c) must hold. Correspondingly, as equations (5.6) indicate, we say that arc <i,k> is restricted to be either *IN* or *OUT* of the solution, or is *FREE* to be either in or out. For each of these three cases we will define an appropriate arc fixed cost b_{ik} that captures the Lagrangian objective function coefficients of both z_{ik} and s_{ik}. With the additional inequalities described in this section, the enhanced Lagrangian subproblem [ULAN2(μ)] has the following objective function coefficients:

- z_{ik} has G_{ik} (the original cable installation cost) as objective function coefficient; and,
- s_{ik} has the coefficient - μ_{ik} in the Lagrangian objective function.

To account for these Lagrangian objective coefficients, we define the equivalent arc fixed cost as follows:

(i) if arc <i,k> is forced to be IN,

$$b_{ik}^{IN} \equiv G_{ik} + \min \{ (e_{ik}-\mu_{ik}) [fmin_{ik}-B_{ik}], (e_{ik}-\mu_{ik}) \hat{M}_{ik} \}; \quad (5.7a)$$

(ii) if arc <i,k> is OUT,

$$b_{ik}^{OUT} \equiv 0; \text{ and} \quad (5.7b)$$

(iii) if arc <i,k> is FREE,

$$b_{ik}^{FREE} \equiv \min \{0, G_{ik} + (e_{ik} - \mu_{ik}) \hat{M}_{ik}\}. \quad (5.7c)$$

In these expressions, $\hat{M}_{ik} = \min \{M_{ik}, fmax_{ik}-B_{ik}\}$. Essentially, to determine the arc fixed cost b_{ik} we solve the cable expansion subproblem (defined in Section 3.1) corresponding to arc <i,k> with one of the three additional constraints (5.6a), (5.6b), or (5.6c), and set the fixed cost b_{ik} equal to the optimal value of this subproblem. Note that, since $fmin_{ik}$ and $fmax_{ik}$ vary depending on the homing pattern, the arc fixed cost could also vary for the different cases.

We now complete the specification of the modified DP algorithm by describing how to distinguish between various possible homing patterns. Recall that our DP algorithm recursively calculates a cost value HC(i,j) which is the total cost (connection+concentrator+cable cost) to serve the demand for all nodes within subtree T(i), assuming that node i homes on node j. We then obtained the value TC(i), defined as the total cost of serving all nodes of T(i) using only concentrators located within T(i), as the minimum value of HC(i,j) over all homing nodes $j \in T(i)$. In the basic algorithm DP1, to compute HC(i,j) we did not further distinguish the homing patterns for successors r_i and l_i of node i. However, as we have just noted, the additional valid inequalities introduce arc fixed costs that vary with the assignment pattern for node i's successors. We, therefore, divide the computation of the cost HC(i,j) into several subcases.

Let us illustrate the revised procedure for computing the cost of covering all nodes in subtree T(i), for some intermediate node i, assuming node i homes on a node j outside T(i). Suppose arc $<i,p_i>$ is a bottleneck arc. First, to account for the bottleneck-arc installation inequalities, we must distinguish between the two broad cases: (i) all nodes of T(i) home outside the subtree (i.e., on node j), or (ii) T(i) contains at least one concentrator. To make this distinction, we define the following two costs EHC(i,j) and PHC(i,j) in place of the single cost HC(i,j) defined for DP1:

EHC(i,j) = Cost of serving all nodes in T(i), assuming <u>all</u> nodes of T(i) home on an *external* node $j \notin T(i)$; and

PHC(i,j) = Cost of serving all nodes in T(i), assuming i homes on node j, and T(i) contains at least one concentrator, i.e., node j serves T(i)'s demand only *partially*.

Observe that EHC(i,j) corresponds to the homing pattern denoted as Case 1.1 in Table 1, while PHC(i,j) includes all other cases.

To compute EHC(i,j), we note that if T(i) does not contain any concentrator, then neither do the two subtrees $T(r_i)$ and $T(l_i)$. Thus, the tree covering cost consists of the adjusted homing cost d_{ij} (including the arc fixed cost b_{ip_i} with $fmax_{ip_i} = D_i$), and the cost of covering all nodes of $T(r_i)$ and $T(l_i)$, with neither subtree containing any concentrator, i.e.,

$$EHC(i,j) = d_{ij} + EHC(r_i,j) + EHC(l_i,j). \tag{5.8}$$

Computing PHC(i,j) is more complex since we must consider various possibilities for the homing patterns and number of concentrators within subtrees $T(r_i)$ and $T(l_i)$. In particular, when the homing node j does not belong to subtree T(i), we will compute the total cost of covering the demand for nodes in T(i) under each of the subcases corresponding to Cases 1.2 through 1.5; we denote the cost under, say, subcase 1.2.1 as $PHC_{1.2.1}(i,j)$. The value of PHC(i,j) is then the minimum cost over all the subcases, i.e.,

$$\begin{aligned} PHC(i,j) = \min \{ & PHC_{1.2.1}(i,j), PHC_{1.2.2}(i,j), PHC_{1.2.3}(i,j), PHC_{1.3.1}(i,j), \\ & PHC_{1.3.2}(i,j), PHC_{1.4.1}(i,j), PHC_{1.4.2}(i,j), PHC_{1.5}(i,j) \}. \end{aligned} \tag{5.9}$$

Likewise, we can compute PHC(i,j) when j belongs to $T(r_i)$ (using Cases 2.1.1, 2.1.2, and 2.2) or $T(l_i)$. Appendix 1 presents the modified DP algorithm in its entirety.

To summarize, this section has described several classes of valid inequalities that strengthen the local access network planning problem formulation. We also

discussed modifications to the basic Lagrangian solution procedure needed to incorporate these cuts. The next section presents computational results comparing the performance of the decomposition method, with and without the cuts and other model enhancements, for three test networks.

6. Computational Results

Our research is best characterized as applications-driven algorithmic development, with the objective of developing an effective solution method for a practical problem. Consequently, computational testing was a central element throughout the research project. Indeed, as the discussions in Sections 4 and 5 suggest, we followed an iterative process of computational testing and algorithmic enhancement—testing the performance of the algorithm's current version, gaining insights about its shortcomings, and devising techniques (preprocessing, coefficient reduction, valid inequalities) to address these deficiencies. To perform these computational tests, we used three test problems derived from actual networks. Section 6.1 describes some broad characteristics of our test networks. We then present computational results using: (i) the basic model as is, and (ii) the enhanced model with reduced cable expansion bounds and valid inequalities. To obtain a point of comparison for assessing the computational time of our algorithm and for gauging the quality of the solutions it generates, we also attempted to solve the mixed-integer formulation and the linear programming relaxation for all three problems using a general purpose mathematical programming software package (LINDO). Section 6.2 reports the IP and LP results, and our initial experience with the basic [LAN1] model. Section 6.3 shows the dramatic improvement in computational performance resulting from the modeling and algorithmic enhancements. Section 6.4 reports on several computational tests with parametrically-scaled demand and cost values for the three network configurations.

6.1 Test Problems

Our computational tests employed three test problems—a 27-node network (***Problem 1***), a 25-node network (***Problem 2***), and a 41-node network (***Problem 3***). Each of the three networks represents an existing local access configuration connecting customer nodes to a switching center; the capacity B_{ij} of each edge (i,j) in the tree corresponds to the existing number of cables in the section connecting nodes i and j. The configuration for Problem 2 contained two nodes with three successors each, and Problem 3 had one node with three successors. We split each of these three-successor nodes into two dummy nodes in order to reduce the number of successors to two. The networks also contained several intermediate nodes with only one successor each. Instead of adding dummy nodes to create two-successor intermediate nodes (as we had assumed in Section 3), we adapted our

implementation to handle both single-successor and two-successor intermediate nodes.

For each network, telecommunication planners provided us with information on the projected demand at every customer node. Figure 3 shows the 41 node network configuration for Problem 3, the projected demand at each node, the number of existing cables in each section, and the cable sections with projected exhaust (i.e., sections (i,j) with existing capacities B_{ij} less than the total demand D_i in subtree T(i)). To obtain expansion cost estimates, we examined the actual, prevailing costs for various transmission and concentration technologies. We then applied a simple regression procedure to use this data to approximate the actual cost functions for our model.

Cable expansion costs vary by (i) construction type (aerial, buried, or underground), and (ii) cable gauge (22, 24 or 26). The expansion on each section (edge) was limited to the type and gauge currently in use on that section. Using information on the expansion cost ($ per unit distance) for various package sizes (i.e., different available cable capacities), we performed regression analyses (for each construction type and gauge) to determine appropriate fixed and variable cable expansion costs per unit distance. For all cable types, the correlation coefficient between the actual and approximated cost was 0.98 or larger.

To determine the concentrator cost parameters, we examined prevailing cost data for 8 different concentration options (multiplexers, concentrators, and remote switches). Each option had its own fixed cost, variable cost per unit of capacity, and upper bound on the possible throughput. We considered 8 different concentrator options because the same physical technology (say, multiplexers) is available in different size ranges (three for multiplexers) with varying fixed and variable costs. We superimposed the 8 cost curves (as a function of throughput), and approximated the lower envelope of these curves with a piecewise-linear, concave cost function (similar to Figure 1) containing three segments. Effectively, these three segments (or concentrator "types") corresponded to three different technologies, and provided a reasonably close approximation to the actual cost envelope (the actual cost envelope contained some minor discontinuities at the breakpoints due to capacity constraints). For the test problems, we used the same concentrator cost structure at all nodes (though our algorithm can handle differential costs) except the root node (we set the concentrator costs at this node to 0). Our test problems did not contain any special node-to-concentrator *connection* costs (i.e., $A_{ij} = 0$ for all node pairs i and j).

6.2 Initial Computational Results

6.2.1 Lagrangian results for the Basic Model [LAN1]

We initially implemented the Lagrangian/dynamic programming algorithm for the basic model [LAN1] in FORTRAN on an IBM 3083. The implementation incorporated: (i) the problem preprocessing method described in Section 4.1, and (ii) the local improvement heuristic described in Section 3.3. This initial approach did not incorporate our coefficient reduction method (of Section 4.2), or any of the valid inequalities described in Section 5. For all our tests, we initialized all Lagrange multipliers to zero, used an initial step size multiplier value (see, for example, Held et al. [1974]) of 2.0, and permitted a maximum of 100 subgradient iterations (the procedure might terminate earlier if the percentage gap between the upper and lower bounds reduces to a very small fraction).

Table 2 summarizes the computational results of this initial solution approach for the three test problems. (The upper and lower bounds reported in this table are scaled; for each test problem, we express the bounds as a % of the optimal LP value for the basic model reported in Table 3.) We measured *preprocessing effectiveness* by the proportion of assignment variables x_{ij} that the preprocessing technique described in Section 4.1 was able to eliminate (by discovering that node i cannot home on node j in any optimal solution). Note that the total number of possible assignments indicated in Table 2 excludes root node assignments (the root node always homes on itself), and assignments to dummy nodes (obtain by splitting nodes with more than two successors). To evaluate the performance of the Lagrangian algorithm, we used a *% gap* statistic, defined as the difference between the best upper and lower bounds as a percentage of the lower bound. The CPU times reported in Table 2 correspond to the total computational time (in seconds) on the IBM 3083, including the time required for input and initialization, preprocessing, subgradient optimization, and the local improvement heuristic.

The % gaps range from 64% to 124%, and the algorithm required approximately 8 seconds for the smaller problems (Problems 1 and 2), and 22 seconds for the 41-node problem (Problem 3). To evaluate the effectiveness of preprocessing, we performed a separate set of computational experiments (not reported here) without the provisions for variable elimination. We found that, for Problem 3, the preprocessing procedure was quite effective in reducing the gap between the upper and lower bounds. For this problem, the % gap without preprocessing was around 160%, and with preprocessing the gap was 124% gap; we achieved this reduction in the % gap through improvements in both the upper and lower bounds. For Problems 1 and 2, the improvement in the % gap due to preprocessing was only marginal. For all three test problems, the preprocessing procedure did not require more than 0.1 seconds of CPU time.

The large % gaps between the upper and lower bounds reported in Table 2 suggested that either: (i) we might be able to find much more effective solutions (with costs reduced by as much as half), i.e., our Lagrangian-based heuristic solution procedure was not very effective, or (ii) the heuristic solutions were close to optimal, but the algorithm could not establish their near-optimality because of weak Lagrangian lower bounds for the basic model. To determine the underlying cause of these large gaps, we solved the test problems using a general-purpose linear and integer programming package. Our results confirmed the latter conclusion, i.e., although the heuristic procedure generated good solutions, the basic lower bounds were very weak and did not provide good performance guarantees.

6.2.2 Linear and Integer Programming Solutions for Basic Model

To generate a benchmark for the bounds and computation times, we attempted to solve the integer programming formulations as well as the linear programming relaxations for all three test problems. We used LINDO, a commercial mathematical programming package, running on the IBM 3083 mainframe, to solve the IP and LP relaxations.

The formulation that we used in these tests was a slight variation of the basic model [LAN1]. Recall that each of our test problems considers three possible concentrator types at every node, corresponding to the three segments of the concave concentrator cost function. Correspondingly, our formulation uses disaggregated node-to-concentrator assignment variables x_{ijm} with m=1, 2, or 3; the variable x_{ijm} equals 1 if node i homes on a type m concentrator at node j, and is 0 otherwise. To reduce the number of variables, we used x_{jjm} in place of separate concentrator installation variables y_{jm}; consequently, our formulation did not contain the concentrator location constraints (2.3).

Since we were considering the basic model <u>without the valid inequalities</u> described in Section 5, our formulation used <u>undirected</u> cable installation and expansion variables z_{ij} and s_{ij}; the directed version of these variables strengthens the LP relaxation only when we incorporate the additional assignment-forcing cable installation inequalities. Consequently, the formulation contains only "undirected" cable forcing constraints (2.6), and does not require the arc orientation constraints (2.7). Also, our problem formulation did not contain the reduced cable expansion bounds M_{ij} obtained using the coefficient reduction method described in Section 4.2. However, we used the results of preprocessing to reduce the formulation size. Preprocessing eliminated some assignment variables, some cable installation and expansion variables (e.g., if preprocessing reduced the maximum possible flow on an arc below its current capacity), and some contiguity, cable capacity, and cable forcing constraints (for example, we omitted contiguity constraints (2.4) corresponding to i-

to-j assignments that preprocessing eliminated.) Also, our formulation did not contain assignment variables corresponding to the root node (since the root node must always home on itself).

We solved two versions of the basic model—our original formulation **with backfeed**, and a restricted version **without backfeed**. Prohibiting backfeed reduces the number of assignment variables (and constraints) since a node i can now home only on nodes j lying on the path P_{i0} connecting node i to the root node; this restriction also has the potential of eliminating some cable expansion variables (and constraints) since the maximum possible flow could exceed existing capacity on fewer edges. Consequently, the "without backfeed" problem formulation contained fewer variables and constraints than the original, unrestricted version. Since prohibiting backfeed restricts the space of feasible solutions, we expect the LP lower bound to be better (i.e., larger) without backfeed; furthermore, the best integer solution with backfeed might be cheaper than the optimal without-backfeed solution.

Table 3 contains statistics on the reduced problem dimensions (number of integer and continuous variables, and number of constraints) after preprocessing, with and without backfeed, for each test network. The formulation sizes varied from 364 variables and 320 constraints to 2908 variables and 2830 constraints. For each network, Table 3 reports the LP and IP values for both versions of the formulation. (The optimal LP value of the unrestricted formulation serves as the reference point for scaling all of the lower bounds and solution values for a given network.)

We were able to optimally solve all 6 linear programming relaxations, and 5 out of the 6 integer programs. For Problem 3 with backfeed, the branch-and-bound procedure did not reach optimality even after an overnight run exceeding 10 hours of (elapsed) computer time. As expected, the LP values without backfeed exceed the with-backfeed values—by 12%, 3.6%, and 8%, respectively, for the three problems. However, the optimal integer solutions were the same, i.e., the without-backfeed solutions were also optimal for the with-backfeed formulations. Also, for Problems 1 and 2, the optimal integer solutions were the same as the best Lagrangian-based heuristic solutions (see Table 2); and, for Problem 3, the Lagrangian-based solution was only 2.3% more expensive than the best IP value. Finally, the Lagrangian lower bounds were close to the LP lower bounds (with backfeed). For the three problems, the LP lower bounds exceeded the Lagrangian lower bounds by only 0.75%, 3.85%, and 5.09% (as a % of the LP lower bounds). (Recall that, for our relaxation scheme of the basic model, the Lagrangian lower bound can never exceed the LP lower bound.) However, the computation time to solve even the LP relaxation was, on average, *three* times the CPU time required for the entire Lagrangian procedure. Solving the IP by branch and bound required *thirty five* and *one hundred* times the CPU time

required by the Lagrangian method to generate near-optimal solutions to two problems; and, after at least *three hundred* times the CPU time on the third problem, the branch and bound method did not reach optimality.

The LP and IP results confirm that (i) the Lagrangian-based heuristic generates near-optimal solutions, and (ii) the LP (and, hence, the Lagrangian) lower bounds for the basic model are weak, i.e., these bounds do not provide a reliable measure of the quality of the heuristic solutions. This observation led to our study of various modeling and algorithmic enhancements described in Sections 4.2 and 5 to strengthen the relaxation and generate better lower bounds.

6.3 Improved Computational Results

We implemented the enhanced version of our algorithm in the C programming language on a Macintosh II computer (with a Math co-processor). This final implementation incorporated the following features:

(i) initial heuristic solution, using local improvement on the "centralized" and "completely distributed" starting solutions;
(ii) problem preprocessing to eliminate variables;
(iii) the Lagrangian-based heuristic with local improvement;
(iv) coefficient reduction to reduce the magnitude of the cable expansion bounds;
(v) assignment-forcing arc installation and expansion inequalities;
(vi) bottleneck-arc installation and expansion inequalities; and,
(vii) subtree-splitting arc installation and expansion inequalities.

We used the same subgradient settings described in Section 6.2. Table 4 contains summary statistics on the the initial upper bound (i.e., the lower of the two cost values corresponding to the improved solutions derived from the "centralized" and "distributed" starting solutions), the (final) best upper and lower bounds, the % gap, and the total computation time (elapsed time on Mac II, including time for input/output, initialization, heuristic, and subgradient iterations) for the three test problems. Again, we have scaled all the bounds for each problem with respect to the optimal LP value.

Comparing the results of Table 2 and Table 4 highlights the dramatic reduction in the % gaps resulting from our enhancements—from 64%, 81%, and 124% to 1.2%, 3.2%, and 7.0%, respectively, for Problems 1, 2 and 3. As we mentioned previously, even our basic algorithm heuristically generated the true optimal solution for Problems 1 and 2. The enhanced algorithm also finds these same solutions. For Problem 3, however, the enhancements to the Lagrangian lower bounding procedure have also led to a modest improvement (1.6%) in the best upper

bound; and, this improved Lagrangian-based heuristic solution is only 0.66% more expensive than the best solution found using LINDO.

The order-of-magnitude reduction in the % gaps is mainly due to the vastly improved lower bounds. The new Lagrangian lower bounds are 60.85%, 68.52%, and 95.36% larger than the optimal LP values for the basic [LAN1] model corresponding to the three test problems. These results suggest that, even though the local access network planning formulation has numerous other valid inequalities, the cumulative effect of the few classes of cuts that we selected and implemented far exceeds the potential effectiveness of the remaining, more complex inequalities.

Although the computation times reported in Tables 3 and 4 are not commensurate, we estimate that classical branch and bound methods for solving the integer programs optimally might require several orders of magnitude more computation time than the composite Lagrangian relaxation algorithm. We make two final observations regarding the heuristic procedures:

(i) For Problem 1, our initial heuristic improvement method identified the optimal solution. For Problems 2 and 3, however, the Lagrangian-based heuristic improved upon the initial upper bound significantly (by 19.7% and 15.82%), indicating that, even with weak lower bounds, the Lagrangian relaxation procedure can serve as a valuable heuristic method since it generates useful starting solutions for local improvement.

(ii) Neither the "centralized" nor the "distributed" concentrator location strategy consistently provides better starting points for initial local improvement. For Problems 1 and 3, the centralized strategy produced a better locally improved initial heuristic solution; the distributed strategy provided a better starting point for Problem 2.

6.4 Computational Results with Demand and Cost Variations

Having established the effectiveness of the enhanced Lagrangian-based algorithm in finding good upper and lower bounds for the three test problems, we wished to explore the method's robustness as the demand and cost parameters changed. For this purpose, we applied the algorithm to problem variations created by scaling the input data, holding the topology of the three networks fixed. In particular, we tested the following scaled versions of each network:

(i) **Demand variation**: We considered scenarios with uniformly lower or higher demand values obtained by multiplying the original demand at each node by a common scale factor. We tested three demand scale factors: 0.5, 2.0, and 5.0;

(ii) **Cable expansion cost variation**: To test robustness with respect to cable expansion costs, we uniformly increased or decreased the variable cable expansion cost by a common scale factor for all edges. We tested two values of the variable cost scale factor: 0.5 (lower variable cost), and 2.0 (higher variable cost); and,

(iii) **Concentrator cost variation**: To test robustness with respect to concentrator costs, we increased or decreased the fixed concentrator cost (for every concentrator type) by a fixed amount at all locations. For our tests, we considered a concentrator fixed cost increase (or decrease) equal to approximately 50% of a type 2 concentrator's fixed cost.

Table 5 contains the computational results—% problem reduction, % gap, and computation time—for the 6 variations corresponding to Problem 1, 2, and 3. For ease of comparison, we also report the computational statistics for the *base case* (i.e., the original problem); in terms of our scaling parameters, this case corresponds to a demand scale factor of 1.0, cable expansion variable cost scale factor of 1.0, and no concentrator fixed cost addition or reduction.

The results shown in Table 5 are encouraging; for 19 out of the 21 problem instances, the % gap between the upper and lower bounds is less than 8%. The trends in the best expansion strategies are consistent with our expectations. For instance, as the demand increases, the number of concentrators increases. Because of economies of scale, concentrators are more cost effective at larger throughput values; indeed, at extremely high demand values (relative to the existing cable capacities), we expect a completely "decentralized" strategy with a concentrator at each node. Similarly, as the variable cable cost increases (relative to the fixed cable costs, and to the concentrator costs), the expansion plans tend to replace cable expansions with additional concentrators. The opposite effect occurs as the fixed concentrator costs increase.

Table 5 also illustrates that, as the demand scale factor increases, the % gap initially increases and then declines. Two factors contribute to this behavior: First, at very low demand values, the system requires no expansion (cables or concentrators); therefore, the upper and lower bounds are both zero, resulting in a zero gap. At the other extreme, with very high demands, the existing cable capacities are negligible compared to the nodal demands. Consequently, the total cost of a "new" network, ignoring existing cable capacities, should not differ appreciably from the true optimal cost of the capacitated problem. Thus, the intermediate demand values are the most challenging problems for the Lagrangian relaxation procedure. The second reason for performance improvements at higher demand values is the improved effectiveness of the preprocessing (problem reduction) method. As the

nodal demands increase, because of the economies of scale in concentrator costs, remote homing (i.e., homing i on another node j) becomes less cost-effective, and so the problem reduction procedure becomes more effective. Consequently, the formulation becomes tighter (and smaller)—with fewer cable installation and expansion variables (z_{ij} can be fixed to either 0 or 1 a priori), and lower values of the cable expansion bound M_{ij}—resulting in better algorithmic performance.

In general, Table 5 shows that, as the cable and concentrator costs increase, the % gap decreases. Again, the increasing effectiveness of preprocessing, and the decreasing relative importance of ignoring existing cable capacities explains this behavior.

7. Concluding Remarks

This paper has attempted to integrate three essential ingredients of contemporary applied integer programming research:

(1) an important practical application: Many changes in the telecommunications industry—technological advances, explosive growth, capital intensive facilities, and increasing competition—have raised renewed opportunities for the use of sophisticated, optimization-based decision support techniques for telecommunication network planning;
(2) decomposition methodology: Since its first successful application for solving large-scale traveling salesman problems (Held and Karp [1970],[1971]), Lagrangian relaxation has become a pervasive technique for exploiting special structure and solving difficult, large-scale discrete optimization models; and
(3) model improvements based on polyhedral combinatorics: Motivated by their marked success for solving large-scale traveling salesman and many other problems, polyhedral techniques have become an essential building block for designing state-of-the-art algorithms for solving several classical integer programming problems.

Our challenge was to develop an effective methodology for solving the important, practical problem of planning for local access network expansion. Decomposition and polyhedral techniques were natural choices for developing this methodolgy.

In this paper, we have demonstrated how to exploit the problem's special tree structure, and how to integrate formulation enhancements (i.e., valid inequalities to strengthen the problem formulation) with Lagrangian relaxation. Our computational results establish the effectiveness of this combined approach for

solving practical problem instances. This project also serves to highlight model-building issues such as the tradeoff between model accuracy and solution effectiveness. For instance, telecommunication planners might not explicitly reveal assumptions such as the contiguity property of expansion plans. However, they readily endorsed this assumption once we identified its crucial impact on our algorithmic development. Similarly, analysts and practitioners must evaluate the modeling approximation for concentrator costs (as continuous, piecewise-linear concave functions of throughput) in terms of its potential impact on both the solution quality and the computational burden imposed by a more accurate model.

While we believe that telecommunication planners will find direct application for our model and solution methodology, we also recognize some of its shortcomings. In particular, we have considered a static or single-period model that determines the optimal overall expansion strategy to meet the telecommunication demands at the terminal year of the planning horizon. Our model does not address the issue of how to sequence the expansions, i.e., what facilities to install during each period of the planning horizon. To identify the optimal network "evolution" plan, we must consider a multi-period model with separate facility expansion variables for each period of the planning horizon. This model would use demand and cost information for each period, and make tradeoffs between (i) installing facilities in anticipation of future demands to exploit economies of scale, and (ii) expanding capacities just-in-time to take advantage of decreasing (discounted) costs for future investments. In spite of this shortcoming, we believe that the single period model is practically useful. First, because of its larger dimensionality and temporal linkages, the multi-period model is much more difficult to solve than its single-period counterpart. Furthermore, the single-period model might serve as the first step in a hierarchical planning procedure in which the planner first decides the target network configuration at the end of the planning horizon, and then determines the optimal evolution plan that reaches this eventual target while meeting the demand in each intermediate period. Alternatively, analysts could use the single-period model, possibly in some modified form, to plan progressive facility expansions; for instance, we could apply it sequentially or iteratively to successive time periods of the planning horizon. Finally, we might consider a decomposition approach for the multi-period model, say, a Lagrangian relaxation procedure that dualizes the constraints linking concentrator and cable capacities across successive time periods. In this scheme, the subproblems reduce to single period expansion planning problems that could be solved using the model described in this paper. A comparative evaluation of these and other multi-period planning strategies will be the focus of a future paper.

Modeling multiple services is another potentially fruitful area for investigation, especially in view of the projected increase in the telecommunication industry's diversity of services. Our single service model applies to multi-service

contexts whenever we can express the demands for different services (such as voice, data, and video) in commensurate units, and different services do not impose unique processing requirements or require different transmission media. However, our model cannot accomodate situations when different services have varying transmission and processing requirements. Since practitioners have not yet implemented fiber-optic transmission, video services, and other advanced features (e.g., intelligent routing, local database query, directory assistance) except in a few experimental networks, analysts still face a great deal of uncertainty concerning how a multiple services model might possibly differ from the basic single (aggregate) service model. Finally, with the projected transition from copper to fiber optic-based transmission in the local network, issues of reliability and connectivity might become more dominant than, or at least stand on equal footing with, capacity and cost considerations. In particular, because of the extremely high bandwith and vulnerability of fiber-optic networks, the focus of modeling efforts for the next generation of local access networks might move away from minimizing fixed plus variable expansion cost for a tree network to configuring a reliable network at minimum total fixed cost. Some recent research (e.g., Groetschel and Monma [1988]) has shed some light on these issues.

Acknowledgments: We wish to thank Marcia Helme, Jeffrey Musser, and Alexander Shulman for their valuable insights and modeling contributions, and for providing the linkage with network planners in the field. Much of Richard Wong's work was completed at Purdue University, West Lafayette, Indiana.

References

Balakrishnan, A., T. L. Magnanti, A. Shulman, and R. T. Wong, "Models for Planning Capacity Expansion in Local Access Telecommunication Networks", to appear in *Annals of Operations Research* (1991a)

Balakrishnan, A., T. L. Magnanti, and R. T. Wong, "Local Access Telecommunication Network Expansion: Modeling and Polyhedral Characterization", in preparation (1991b)

Barany, I., J. Edmonds, and L. A. Wolsey, "Packing and Covering a Tree by Subtrees", *Combinatorica*, Vol. 6, pp. 221-233 (1986)

Fisher, M. L., "The Lagrangian Relaxation Method for Solving Integer Programming Problems", *Management Science*, Vol. 27, pp. 1-18 (1981)

Groetschel, M., and C. L. Monma, "Integer Polyhedra Arising from Certain Network Design Problems with Connectivity Constraints", Report No. 104, Institut für Mathematik, Univeritaet Augsburg, Augsburg (1988)

Held, M., and R. M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees", *Operations Research*, Vol. 18, pp. 1138-1162 (1970)

Held, M., and R. M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees: Part II", *Mathematical Programming*, Vol. 1, pp. 6-25 (1971)

Held, M., P. Wolfe, and H. D. Crowder, "Validation of Subgradient Optimization", *Mathematical Programming*, Vol. 6, pp. 62-88 (1974)

Hoffman, K. and M. W. Padberg, "LP-based Combinatorial Problem Solving", *Annals of Operations Research*, Vol. 4, pp. 145-194 (1985)

Kariv, O., and S. L. Hakimi, "An Algorithmic Approach to Network Location Problems II: The p-medians", *SIAM Journal of Applied Mathematics*, Vol. 27, pp. 539-560 (1979)

Nemhauser, G. and L. A. Wolsey, Integer and Combinatorial Optimization, John Wiley & Sons, New York (1988).

Appendix 1

Enhanced Dynamic Program [DP2] for the Uncapacitated Local Access Network Subproblem with Valid Inequalities

Notation

Problem parameters

d_i	=	demand at node i
B_{ij}	=	existing capacity of edge (i,j)
G_{ij}	=	cable fixed cost on edge (i,j)
e_{ij}	=	cable variable cost on edge (i,j)
F_j	=	concentrator fixed cost at node j
c_j	=	concentrator variable cost at node j
A_{ij}	=	connection cost for assigning node i to a concentrator at node j
M_{ij}	=	cable expansion bound for arc <i,j>
μ_{ij}	=	Lagrange multiplier for edge (i,j)'s cable capacity constraint

Network descriptors

T	=	given tree network
i	=	node index, i = 1,2,..n, in increasing order of node level; i = 0 is the root node
P_{ij}	=	path from node i to node j
k_{ij}	=	node i's neighbor on the path P_{ij}
Lev(i)	=	level of node i, i.e., number of arcs on the path P_{i0}
p_i	=	predecessor of node i; $p_i < i$
r_i	=	right successor of node i; $r_i > i$
l_i	=	left successor of node i; $l_i > i$
T(i)	=	subtree rooted at node i after edge (i,p_i) is deleted from T

DP parameters

D_i	=	total demand of all nodes in T(i)
$dmin_i$	=	smallest demand among all leaf nodes of T(i)
$fmin_{ij}$	=	min. flow on arc <i,j>
$fmax_{ij}$	=	max. flow on arc <i,j>

$A_{ij}(\mu)$ = assignment cost for assigning node i to node j in ULAN1(μ)

$$= A_{ij} + d_i \Big(\sum_{(k,l)\in P_{ij}} \mu_{kl} \Big) + d_i c_j \qquad (1)$$

$\hat{M}_{ik}$ = $\min \{M_{ik}, fmax_{ik}-B_{ik}\}$

b_{ik} = equivalent fixed cost for arc <i,k>

$$\triangleq \begin{cases} G_{ik} + \min \{(e_{ik}-\mu_{ik})\,[fmin_{ik}-B_{ik}], (e_{ik}-\mu_{ik})\,\hat{M}_{ik} & \text{if } fmin_{ik} > B_{ik} \\ 0 & \text{if } fmax_{ik} \le B_{ik}; \text{ and,} \\ \text{Min } \{0, G_{ik} + (e_{ik}-\mu_{ik})\,\hat{M}_{ik} & \text{otherwise.} \end{cases} \qquad (2)$$

d_{ij} = Adjusted homing cost for assigning node i to node j

$$= \begin{cases} A_{ij}(\mu) + b_{i,k_{ij}} & \text{if } i \neq j, \text{ and} \\ A_{ij}(\mu) + F_j & \text{if } i = j. \end{cases} \qquad (3)$$

DP recursive values

EHC(i,j) = total (cable + concentrator) cost of serving all nodes in T(i) when all nodes in T(i) home on $j \notin T(i)$

PHC(i,j) = total cost of serving all nodes in T(i) when node i homes on node j, and T(i) contains at least one concentrator

TC(i) = minimum total cost of serving all nodes in T(i) using only internal concentrators, i.e., node i homes within T(i)

The Enhanced Dynamic Program [DP2]

For i = n,n-1,...,0
  if i is a leaf node,
    for all $j \in N$,
      if $j = i$,
        set $EHC(i,i) \leftarrow \infty$;
      if $j \neq i$,
        $fmin_{ip_i} = fmax_{ip_i} = D_i$;
        calculate b_{ip_i} and d_{ij} using eqn. (2) & (3);
        set $EHC(i,j) \leftarrow d_{ij}$;
    $PHC(i,i) \leftarrow d_{ii}$
    $TC(i) \leftarrow PHC(i,i)$

  else
    for $k = n, n-1,...,i+1$, with $k \in T(i)$,
      for $j = n, n-1,...,i$, with $j \in T(i)$ and $i \in P_{kj}$,
        $fmin_{kp_k} = fmax_{kp_k} = D_k$;
        calculate b_{kp_k} and d_{kj} using eqn. (2) & (3);
        set $EHC(k,j) \leftarrow d_{kj} + EHC(r_k,j) + EHC(l_k,j)$.
      next j;

      for $j = n, n-1,...,i$, with $j \in T(i)$ and $i \in P_{kj}$ (*Case 1*),
        calculate PHC(k,j):
          for cases 1.2.1, 1.2.2, 1.2.3, 1.3.1, 1.3.2, 1.4.1, 1.4.2, 1.5
          (i.e., possible homing patterns for r_k and l_k),
            compute $fmin_{kp_k}$ and $fmax_{kp_k}$ using Table 1;
            calculate b_{kp_k} and d_{kj} using eqn. (2) & (3);
          Case 1.2: r_k and l_k home on j;
            $PHC_{1.2.1}(k,j) \leftarrow d_{kj} + PHC(r_k,j) + PHC(l_k,j)$;
            $PHC_{1.2.2}(k,j) \leftarrow d_{kj} + EHC(r_k,j) + PHC(l_k,j)$;
            $PHC_{1.2.3}(k,j) \leftarrow d_{kj} + PHC(r_k,j) + EHC(l_k,j)$;
          Case 1.3: r_k homes on j; l_k homes within $T(l_i)$;
            $PHC_{1.3.1}(k,j) \leftarrow d_{kj} + EHC(r_k,j) + TC(l_k)$;
            $PHC_{1.3.2}(k,j) \leftarrow d_{kj} + PHC(r_k,j) + TC(l_k)$;
          Case 1.4: r_k homes within $T(r_k)$; l_k homes on j;
            $PHC_{1.4.1}(k,j) \leftarrow d_{kj} + TC(r_k) + EHC(l_k,j)$;
            $PHC_{1.4.2}(k,j) \leftarrow d_{kj} + TC(r_k) + PHC(l_k,j)$;
          Case 1.5: r_k and l_k home within $T(r_k)$ and $T(l_k)$;

$PHC_{1.5}(k,j) \leftarrow d_{kj} + TC(r_k) + TC(l_k)$;

$PHC(k,j) \leftarrow$ Min {$PHC_{1.2.1}(k,j)$, $PHC_{1.2.2}(k,j)$, ..., $PHC_{1.4.2}(k,j)$, $PHC_{1.5}(k,j)$}.

next j;

next k;

for k = i,

for j = n, n-1, ..., i+1, with j ∈ $T(r_i)$ (***Case 2***),

calculate PHC(i,j):

for cases 2.1.1, 2.1.2, 2.2

(i.e., possible homing patterns for l_i),

compute $fmin_{ir_i}$ and $fmax_{ir_i}$ using Table 1;

calculate b_{r_i} and d_{ij} using eqn. (2) & (3);

Case 2.1: l_i homes on j;

$PHC_{2.1.1}(i,j) \leftarrow d_{ij} + PHC(r_i,j) + EHC(l_i,j)$;

$PHC_{2.1.2}(i,j) \leftarrow d_{ij} + PHC(r_i,j) + PHC(l_i,j)$;

Case 2.2: l_i homes within $T(l_i)$;

$PHC_{2.2}(i,j) \leftarrow d_{ij} + PHC(r_i,j) + TC(l_i)$;

$PHC(i,j) \leftarrow$ Min {$PHC_{2.1.1}(i,j)$,$PHC_{2.1.2}(i,j)$,$PHC_{2.2}(i,j)$}.

next j;

for j = n, n-1, ..., i+1, with j ∈ $T(l_i)$ (***Case 3***),

calculate PHC(i,j):

for cases 3.1.1, 3.1.2, 3.2

(i.e., possible homing patterns for r_i),

compute $fmin_{il_i}$ and $fmax_{il_i}$ using Table 1;

calculate b_{l_i} and d_{ij} using eqn. (2) & (3);

Case 3.1: r_i homes on j;

$PHC_{3.1.1}(i,j) \leftarrow d_{ij} + EHC(r_i,j) + PHC(l_i,j)$;

$PHC_{3.1.2}(i,j) \leftarrow d_{ij} + PHC(r_i,j) + PHC(l_i,j)$;

Case 3.2: r_i homes within $T(r_i)$;

$PHC_{3.2}(i,j) \leftarrow d_{ij} + TC(r_i) + PHC(l_i,j)$;

$PHC(i,j) \leftarrow$ Min {$PHC_{3.1.1}$,$PHC_{3.1.2}$,$PHC_{3.2}$}.

next j;

for j = i (***Case 4***),

calculate PHC(i,i):

Case 4.1: r_i and l_i home on i; $T(r_i)$ and $T(l_i)$ each contain either 0 or ≥ 1 conc.

$PHC_{4.1.1}(i,i) \leftarrow d_{ii} + EHC(r_i,i) + EHC(l_i,i);$

$PHC_{4.1.2}(i,i) \leftarrow d_{ii} + EHC(r_i,i) + PHC(l_i,i);$

$PHC_{4.1.3}(i,i) \leftarrow d_{ii} + PHC(r_i,i) + EHC(l_i,i);$

$PHC_{4.1.4}(i,i) \leftarrow d_{ii} + PHC(r_i,i) + PHC(l_i,i);$

Case 4.2: r_i homes on i; l_i homes within $T(l_i)$; $T(r_i)$ contains 0 or ≥ 1 conc.

$PHC_{4.2.1}(i,i) \leftarrow d_{ii} + EHC(r_i,i) + TC(l_i);$

$PHC_{4.2.2}(i,i) \leftarrow d_{ii} + PHC(r_i,i) + TC(l_i);$

Case 4.3: r_i homes within $T(r_i)$; l_i homes on i; $T(l_i)$ contains 0 or ≥ 1 conc.

$PHC_{4.3.1}(i,i) \leftarrow d_{ii} + TC(r_i) + EHC(l_i,i);$

$PHC_{4.3.2}(i,i) \leftarrow d_{ii} + TC(r_i) + PHC(l_i,i);$

Case 4.4: r_i and l_i home within $T(r_i)$ & $T(l_i)$

$PHC_{4.4}(i,i) \leftarrow d_{ii} + TC(r_i) + TC(l_i);$

$PHC(i,i) \leftarrow \text{Min } \{PHC_{4.1.1}(i,i), PHC_{4.1.2}(i,i), \ldots\ldots, PHC_{4.4}(i,i)\}.$

$TC(i) \leftarrow \underset{j \in T(i)}{\text{Minimum}} PHC(i,j)$

next i;

Table 1

Minimum and Maximum Flow Parameters for various Homing patterns in the Enhanced Dynamic Program

Case #	*Homing node for*		*No. of Conc. in*			*Min. and Max flow on arc incident from node i*
	r_i	l_i	i	r_i	l_i	
Case 1: Node i homes on node j ∉ T(i)						
1.1	j	j	0	0	0	$fmin_{ip_i} = D_i$ $fmax_{ip_i} = D_i$
1.2.1	j	j	≥ 1	≥ 1	≥ 1	$fmin_{ip_i} = d_i + d_{r_i} + d_{l_i}$ $fmax_{ip_i} = d_i + (D_{r_i} - dmin_{r_i}) + (D_{l_i} - dmin_{l_i})$
1.2.2	j	j	≥ 1	0	≥ 1	$fmin_{ip_i} = d_i + D_{r_i} + d_{l_i}$ $fmax_{ip_i} = d_i + D_{r_i} + (D_{l_i} - dmin_{l_i})$
1.2.3	j	j	≥ 1	≥ 1	0	$fmin_{ip_i} = d_i + d_{r_i} + D_{l_i}$ $fmax_{ip_i} = d_i + (D_{r_i} - dmin_{r_i}) + D_{l_i}$
1.3.1	j	∈ $T(l_i)$	≥ 1	0	≥ 1	$fmin_{ip_i} = d_i + D_{r_i}$ $fmax_{ip_i} = d_i + D_{r_i}$
1.3.2	j	∈ $T(l_i)$	≥ 1	≥ 1	≥ 1	$fmin_{ip_i} = d_i + d_{r_i}$ $fmax_{ip_i} = d_i + (D_{r_i} - dmin_{r_i})$
1.4.1	∈ $T(r_i)$	j	≥ 1	≥ 1	0	$fmin_{ip_i} = d_i + D_{l_i}$ $fmax_{ip_i} = d_i + D_{l_i}$
1.4.2	∈ $T(r_i)$	j	≥ 1	≥ 1	≥ 1	$fmin_{ip_i} = d_i + d_{l_i}$ $fmax_{ip_i} = d_i + (D_{l_i} - dmin_{l_i})$

Table 1 (continued)

Case #	*Homing node for*		*No. of Conc. in*			*Min. and Max flow on arc incident from node i*
	r_i	l_i	i	r_i	l_i	
1.5	$\in T(r_i)$	$\in T(l_i)$	≥ 1	≥ 1	≥ 1	$fmin_{ip_i} = d_i$ $fmax_{ip_i} = d_i$
Case 2: Node i homes on node j $\in T(r_i)$						
2.1.1	j	j	≥ 1	≥ 1	0	$fmin_{ir_i} = d_i + D_{l_i}$ $fmax_{ir_i} = d_i + D_{l_i}$
2.1.2	j	j	≥ 1	≥ 1	≥ 1	$fmin_{ir_i} = d_i + d_{l_i}$ $fmax_{ir_i} = d_i + (D_{l_i} - dmin_{l_i})$
2.2	j	$\in T(l_i)$	≥ 1	≥ 1	≥ 1	$fmin_{ir_i} = d_i$ $fmax_{ir_i} = d_i$
Case 3: Node i homes on node j $\in T(l_i)$						
Cases 3.1.1, 3.1.2, and 3.2 similar to Cases 2.1.1, 2.1.2, and 2.2, with r_i and l_i interchanged.						
Case 4: Node i homes on itself						
fmax and fmin = 0 for all arcs (i.e., arcs $<i,p_i>$, $<i,r_i>$, and $<i,l_i>$) incident from node i.						

Table 2

Computational Results for the Basic Lagrangian Relaxation Scheme

(with Preprocessing and the Local Improvement Heuristic)

Statistic	**Problem 1**	**Problem 2**	**Problem 3**
Number of nodes	27	25	41
No. of possible assignments*	676	441	1521
% reduction in assignment variables	25%	24%	30%
Best Upper Bound using Lagrangian-based heuristic†	162.72	173.91	212.40
Best Lagrangian Lower Bound†	99.25	96.15	94.91
% gap**	64%	81%	124%
CPU time (seconds on IBM 3083)§	8.4 secs	6.8 secs	22.2 secs

* Total number of possible assignments of nodes to concentrator locations, excludes assignments to dummy nodes and from root node.

† All bounds for each problem are scaled with respect to the optimal value of the LP relaxation (Table 3).

** % gap = (Best Upper Bound - Best Lower Bound) /(Best Lower Bound)

§ CPU time includes time for input and initialization, subgradient iterations, and the local improvement heuristic.

Table 3
Linear and Integer Programming Results using LINDO for the Basic Model [LAN1]

	Problem 1	Problem 2	Problem 3
# Nodes	27	25	41
# Variables[§]: Integer/Continuous	814/17	911/18	2870/38
# Constraints[§]	781	889	2830
Optimal Value[*]			
• LP Relaxation	100.00	100.00	100.00
• Integer Program	162.72	173.91	207.70[†]
CPU secs (on IBM 3083)			
• LP Relaxation	14.2	24.3	72.9
• Integer Program	307.8	691.8	†
No. of *Iterations*			
• LP Relaxation (# pivots)	384	1058	1295
• Integer Prog. (# pivots/branches)	47/15898	89/56239	-
Without Backfeed			
# Variables[§]: Integer/Continuous	421/5	352/12	765/24
# Constraints[§]	376	320	711
Optimal Value[*]			
• LP Relaxation	111.98	103.63	108.02
• Integer Program	162.72	173.91	207.70
CPU secs (on IBM 3083) for LP Relaxation/Integer Program	2.8/36.1	4.2/72.4	6.2/2862.6

§ The number of variables and constraints shown in the table correspond to the compact formulation described in Section 6.2.2, after preprocessing eliminated some assignments.

* Objective function values for each problem are scaled with respect to the optimal value of the LP relaxation with backfeed.

† We terminated the integer program for Problem 3, without reaching optimality, after 10 hours of elapsed time on the IBM 3083. The table represents the value of the best incumbent at termination as the optimal value of the integer program.

Table 4

Computational Results for the Enhanced Lagrangian Relaxation Scheme

Statistic	Problem 1	Problem 2	Problem 3
Number of nodes	27	25	41
% reduction in assignment variables*	25%	24%	30%
Intial Upper Bound †	162.72	208.19	242.16
Best Upper Bound using Lagrangian-based heuristic†	162.72	173.91	209.08
Best Lagrangian Lower Bound†	160.85	168.52	195.36
% gap**	**1.2%**	**3.2%**	**7.0%**
Computation (elapsed) time on Mac II §	285 secs	223 secs	879 secs

* % reduction = No. of assignments eliminated + total no. of possible assignments (excluding assignments to dummy nodes and from root node).

† All bounds for each problem are scaled with respect to the optimal value of the LP relaxation with backfeed (Table 6).

** % gap = (Best Upper Bound - Best Lower Bound) /(Best Lower Bound)

§ Computation (elapsed) time includes time for input and initialization, heuristic, and subgradient iterations.

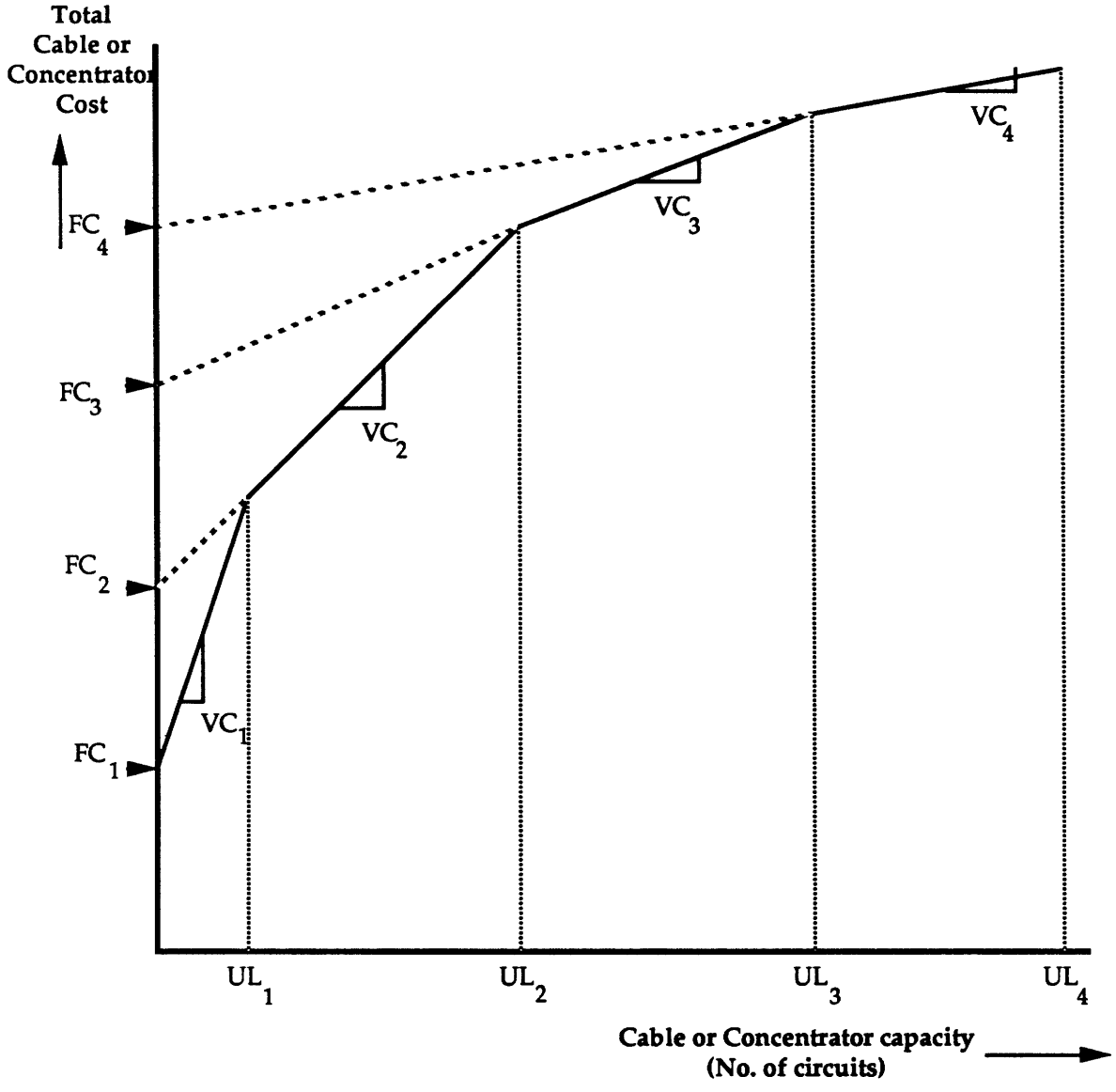
Table 5

Computational Results with Demand and Cost Variations

Parameter Variation	*Problem 1*			*Problem 2*			*Problem 3*		
	% Problem Reducn.	% GAP	Comput. time	% Problem Reducn.	% GAP	Comput. time	% Problem Reducn.	% GAP	Comput. time
Base Case	***25%***	***1.2%***	***285 secs***	***24%***	***3.2%***	***223 secs***	***30%***	***7.03%***	***879 secs***
Demand x 0.5	12%	0.0%	32 secs	12%	0.0%	30 secs	9%	0.0%	82 secs
Demand x 2.0	52%	11.3%	259 secs	53%	5.5%	203 secs	63%	2.71%	648 secs
Demand x 5.0	79%	1.5%	221 secs	80%	0.0%	179 secs	87%	1.1%	546 secs
Var. Cable Cost x 0.5	14%	0.0%	315 secs	14%	3.2%	318 secs	12%	7.5%	1210 secs
Var. Cable Cost x 2.0	40%	3.0%	311 secs	35%	2.8%	234 secs	46%	6.3%	850 secs
Fixed Conc. Cost - 50%†	40%	2.0%	282 secs	36%	5.6%	218 secs	43%	19.9%	836 secs
Fixed Conc. Cost + 50%†	17%	0.5%	292 secs	19%	3.0%	256 secs	20%	5.8%	965 secs

† Increase/decrease in concentrator fixed cost ≈ 50% of type 2 concentrator's fixed cost

Figure 1
Piecewise Linear, Concave Cost Function
for Cable Expansion and Concentrators

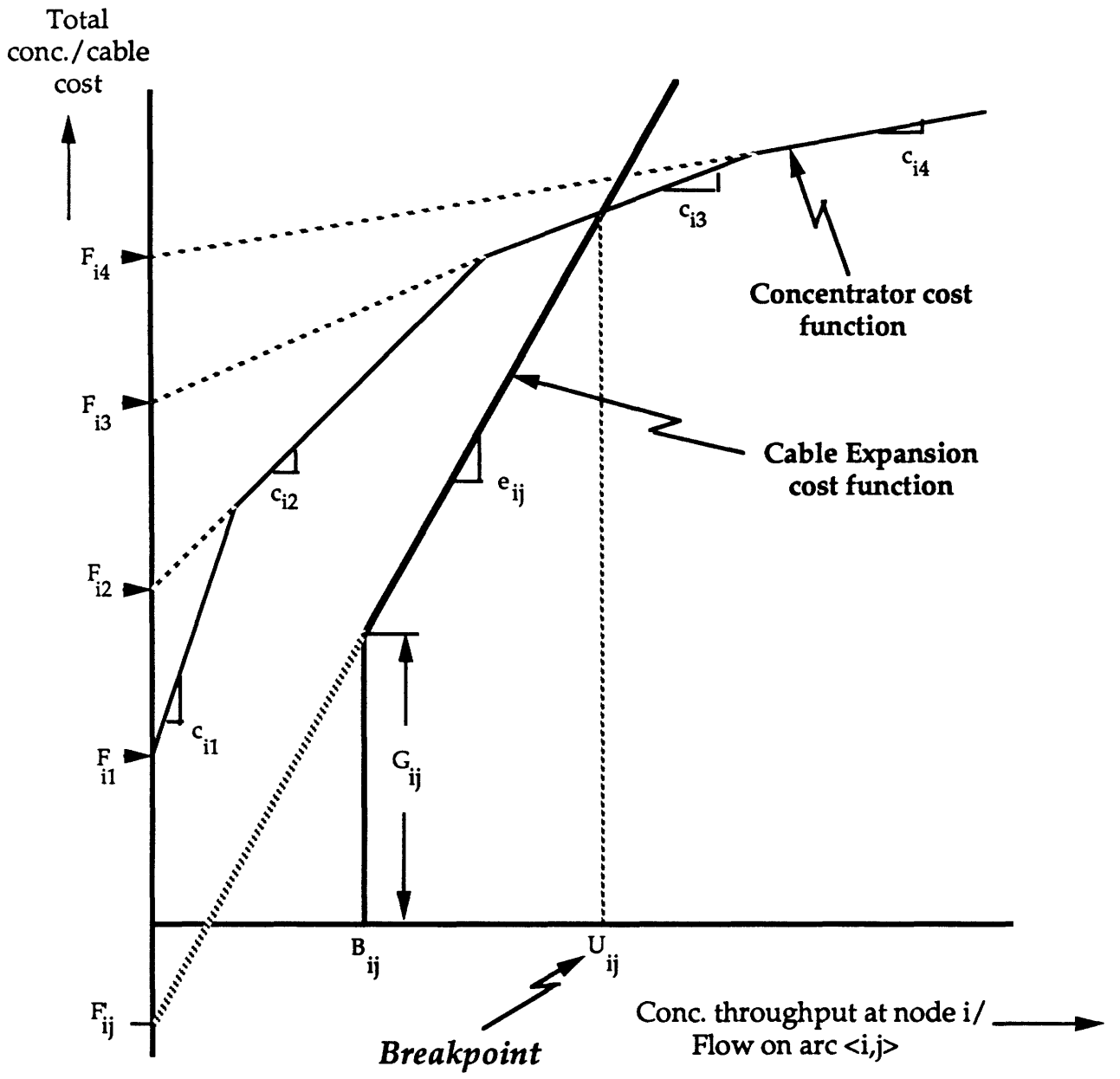


FC_k = Fixed (cable or concentrator) cost for technology k

VC_k = Variable (cable or concentrator) cost for technology k

UL_k = Upper limit on (cable or concentrator) throughput for technology k

Figure 2
Reducing the Maximum Cable Expansion Parameter



F_{im} = Fixed cost of type m concentrator at node i

c_{im} = Variable cost of type m concentrator at node i

G_{ij} = Fixed cost of cable expansion on arc <i,j>

e_{ij} = Variable cost of cable expansion on arc <i,j>

B_{ij} = Existing capacity of arc <i,j>

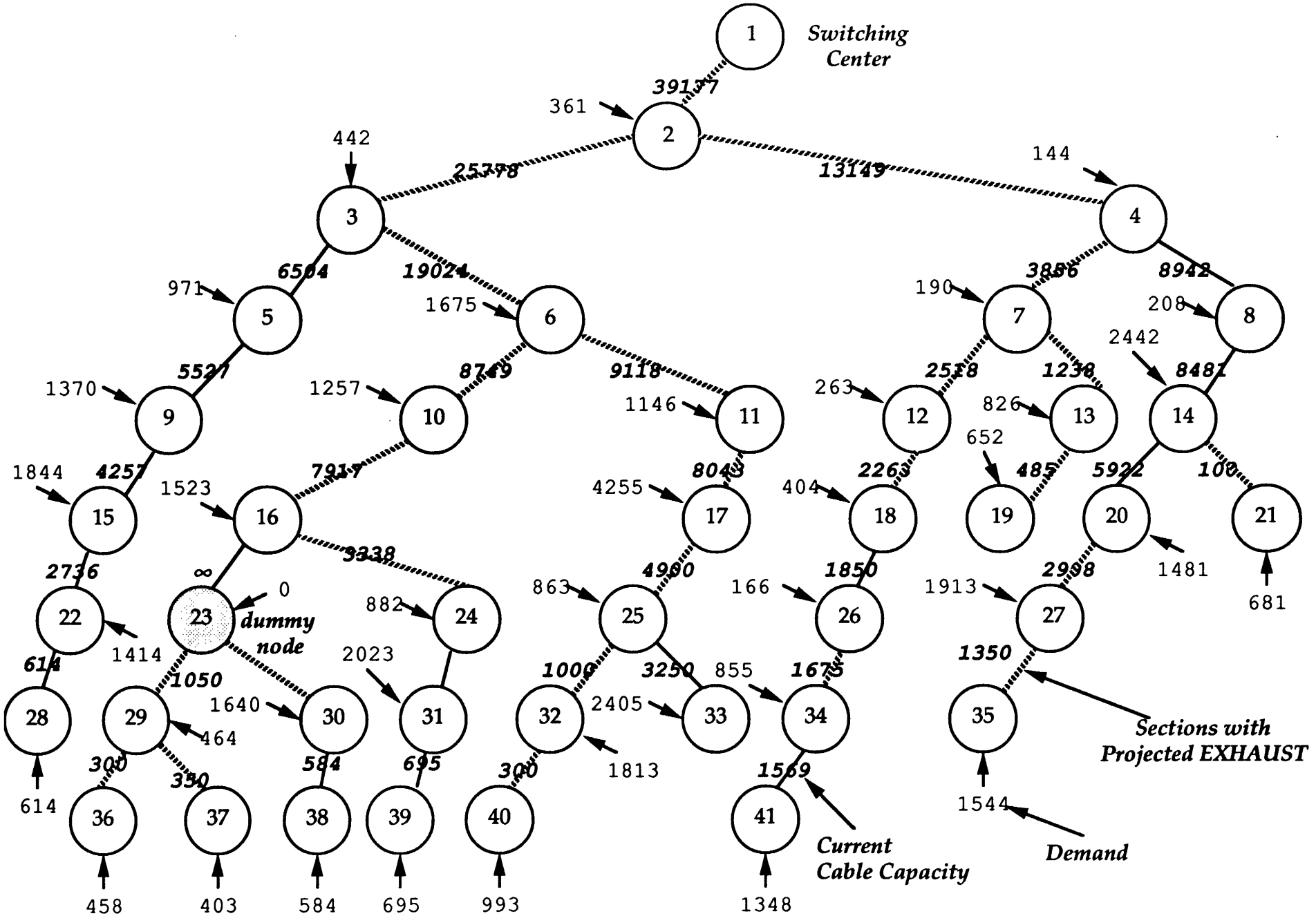


Figure 3: *Network Configuration for Problem 3*