"Using a Hop-constrained Model to Generate Alternative Communication Network Designs

A. Balakrishnan, K. Altinkemer

MIT Sloan School Working Paper #3233-91-MSA

Revised: December 1990

"Using a Hop-constrained Model to Generate Alternative Communication Network Designs

A. Balakrishnan, K. Altinkemer

MIT Sloan School Working Paper #3233-91-MSA

Revised: December 1990

Using a Hop-constrained Model to Generate Alternative Communication Network Designs

Anantaram Balakrishnan[*]
Sloan School of Management
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Kemal Altinkemer
Krannert Graduate School of Management
Purdue University
West Lafayette Indiana 47907

Revised: December 1990

[*] Supported in part by grant DDM-8996120 from the National Science Foundation and a grant from the AT&T Foundation

Abstract

Designing the link topology and selecting capacities in a backbone network of a communication system involves complex tradeoffs between investment and operating costs and service considerations such as network reliability and vulnerability, delays, and blocking. Incorporating all these design criteria simultaneously in a comprehensive model results in a large-scale, non-linear, discrete optimization problem that is intractable. In this paper, we propose an alternate optimization-based methodology that generates several cost-effective backbone network designs with varying cost and performance characteristics. Network planners can use the method in conjunction with detailed performance evaluation techniques to assess the cost impact of different service requirements, and select a design that achieves the proper balance between conflicting objectives. To generate different configurations, the method parametrically varies a set of hop constraints that restrict the number of links over which messages can be transmitted. Reducing the maximum number of hops increases the number of alternate routes but incurs higher total cost for the communication system. For a given set of hop constraints, we develop a Lagrangian-based algorithm to identify a cost-minimizing network design that satisfies all internode traffic requirements. We report results for extensive computational tests of the algorithm using several randomly generated test problems. Our results demonstrate that, even for relatively large problems, the method identifies good heuristic solutions and tight lower bounds that confirm the near-optimality of the selected designs. Using a 25-node example, we illustrate how the model can be used to evaluate the cost versus performance tradeoff.

Keywords: Communication network design, cost-performance tradeoff, Lagrangian relaxation

1. Introduction

This paper presents a methodology to support topological design and link capacity planning decisions for the backbone network interconnecting geographically dispersed gateway nodes and switching centers in a telecommunication system. Topological design decisions are very important in backbone network planning because of the enormous capital investments required for transmission and switching facilities, and the significant impact of network configuration choices on service levels. In selecting the configuration and capacities for a backbone network, planners face conflicting objectives of reducing the total investment and operating costs while ensuring adequate service levels, expressed in terms of network performance measures such as reliability and vulnerability, alternate routing capabilities, blocking probabilities, and queueing and transmission delays. With increasing competition and service diversity in the communications sector, these service considerations are becoming increasingly important.

Because of the multiple, conflicting objectives, non-linearities in service criteria, and discrete design choices, formulating and solving a comprehensive optimization model to support backbone network planning decisions is impractical. Indeed, even with a single objective such as cost minimization, the design task is very complex and challenging because the decision model must (i) incorporate binary network design decisions to capture the significant fixed costs, and (ii) simultaneously account for the close linkages between topological design, link capacity, and routing decisions.

In this paper we propose an alternate approach that uses an optimization-based method to generate several alternative network designs with varying cost-performance characteristics. Our approach is motivated by several observations regarding network design practice. First, network planners must often accomodate various implicit and qualitative design considerations that cannot be adequately represented in optimization models; hence, they prefer a decision support system that identifies a limited set of effective designs rather than a single 'optimal' solution for an imprecise model. Second, in most practical contexts we cannot completely characterize network performance (reliability, delays, blocking) in a convenient form for mathematical modeling, except under very restrictive assumptions about traffic patterns and network behaviour. Therefore, to accurately assess the network's performance, we require detailed evaluative techniques such as simulation and queueing network analysis. Third, the total number of possible network designs grows exponentially with the dimensions of the network. Since detailed performance evaluation of each design is a time-consuming process, planners require a principled method to prune the list of all possible network configurations to a few cost-effective designs with varying performance characteristics. Finally, to keep the model tractable, even a single objective (cost-minimization) model must necessarily make some approximations. In terms of computational complexity, most (deterministic) discrete choice network design problems in telecommunication planning are NP-hard. Therefore, we must rely on optimization-based heuristic procedures that provide provably good solutions.

Thus, to effectively use decision support models, we propose a hierarchical framework for telecommunication network planning. The approach consists of the following four steps:

(1) Generate, using an optimization-based model, a small set of good 'base' designs that span a wide range of cost and service levels.
(2) Refine each base design (e.g., selectively add buffer capacities) and modify it to meet implicit or qualitative design requirements.
(3) Evaluate the true cost and performance of each design using detailed analytical or simulation models.
(4) Select an appropriate design that achieves the desired tradeoff between cost and performance.

The designs generated in Step 1 serve as the basis for marginal user-directed enhancements in Step 2. Each base design specifies an underlying network topology, the preferred routes between various origin-destination pairs, and a corresponding 'rough-cut' capacity plan for switching and transmission resources. The base designs should account for tradeoffs and interrelationships between topological design and routing issues, between fixed and variable (traffic dependent) costs, and between total costs and connectivity (e.g., the number of alternate routes in the network). In Step 2, the planner might refine each base design (perhaps, using a decision support model) by selectively adding buffer capacities to improve network performance, and making minor changes to the topology and routes to accomodate context-specific constraints. Step 3 consists of detailed evaluation of each candidate design, and Step 4 involves choosing a design that balances the planners' preferences for cost versus service. The four steps in this approach are closely interrelated, and might be performed iteratively with, say, Step 2 providing feedback to Step 1 and so on.

In this paper, we focus on a model to generate alternative network designs in Step 1. We also describe model extensions to incorporate some of the design refinements — allocating buffer capacities, and creating alternate routes — contained in Step 2 of the hierarchical planning approach. Since the base designs incorporate all major topological decisions, they largely determine the possible levels of service.

In particular, a base design that is sparse (i.e., contains few links) cannot accomodate alternative routes and is, therefore, inherently unreliable, while a dense design provides routing flexibility. Thus, in order to generate a spectrum of base designs with varying cost-performance characteristics, we must identify topologies that differ in their connectivity levels. For this purpose, we propose a deterministic model that seeks cost-effective designs to meet internode traffic requirements subject to a class of routing restrictions which we call *hop constraints*. For each origin-destination pair, the hop constraint limits the number of links (or hops) on the interconnecting route. We refer to the upper limit on the number of hops between each origin and destination as the route's *Maximum Hop Parameter* (or, *hop parameter* in brief). By systematically changing these hop parameters, we can generate topologies with varying arc densities and costs. In particular, reducing the hop parameter values increases the number of alternate paths, and impacts delays while increasing total cost. Thus, hop constraints serve as a convenient surrogate for service restrictions to generate alternate base designs. In addition, limiting the number of hops for each message also simplifies the task of managing traffic over the network (see, for example, Ash, Murray and Cardwell (1981) and Monma and Sheng (1986)). Recently, LeBlanc and Reddoch (1990) independently developed a hop-constrained model with explicit delay costs for reliable topology/capacity design and routing in backbone networks.

For a given set of hop constraints, we formulate the design problem as a mixed integer program, and propose a Lagrangian relaxation method to solve it approximately, but with performance guarantees. We then propose a systematic method to vary the hop parameters in order to generate a variety of design solutions. The network designer can then apply detailed evaluative models to these candidate designs in order to accurately assess the cost impact of different service requirements.

The rest of this paper is organized as follows. In Section 2, we formally describe the design problem, state our modeling assumptions, and present the mathematical programming formulation. Section 3 describes our Lagrangian-based solution approach to generate upper and lower bounds on the cost of the optimal solution, for a given set of hop constraints. In Section 4 we present extensive computational results for several randomly generated test problems. Our first set of computational tests focuses on evaluating the effectiveness of the proposed algorithm for various problem sizes, cost structures, and hop restrictions. These results demonstrate that the method constructs near-optimal solutions even for relatively large problems with upto 30 nodes, 420 edges, and 420 commodities. Our second set of computational experiments illustrates how the model can be used to gain insights about the tradeoffs between cost and performance by varying the hop restrictions. Section 5 describes some variants and enhancements of the basic hop-constrained network design model to allocate buffer capacities, and provide for alternate routes. Section 6 summarizes the paper and outlines directions for further research.

2. Problem Definition and Formulation

The backbone network design problem involves selecting a subset of links to include in the topology, determining the capacity on each selected link, and routing all internode traffic requirements at minimum total investment and operating cost. We emphasize that the three decisions — topological design, capacity planning, and routing — are closely interrelated, and we seek an integrated model that addresses all three decisions simultaneously.

Many authors have proposed methods to separately solve two components of the backbone design problem — the capacity assignment problem and the routing problem — assuming that the network topology is prespecified. The *capacity assignment* problem (see, for example, Bonucelli (1981), Chou et al. (1978), Maruyama and Tang (1976)) assumes that traffic routes (and, hence, the network topology) are given, and selects the best capacity for each link from a discrete set of available link capacities in order to minimize total cost subject to delay restrictions. On the other hand, the *routing* or *flow assignment* problem starts with a given choice of link topology and capacities, and seeks to identify primary routes between various origin-destination pairs in order to minimize the average or maximum message delay (see, for example, Ahuja (1979), Bertsekas (1980, 1984), Cantor and Gerla (1974), Courtois and Semal (1981), Frank and Chou (1971), Gerla (1973), Tymes (1981), and Yum (1981)).

Combined approaches that iterate between capacity and flow assignment decisions have been discussed by Gerla et al. (1974), Gerla and Kleinrock (1977), and Maruyama, and Fratta and Tang (1977). Gavish and Neuman (1986,1987), Gavish and Altinkemer (1987) and Pirkul and Narasimhan (1987,1988) describe integrated optimization methods for simultaneous capacity assignment and routing. These papers also account for traffic delay (either as a constraint or as a 'cost' in the objective function), node and link failures, and/or alternate routing. Incorporating practical connectivity constraints in fixed-charge network design models is difficult. Eswaran and Tarjan (1976) show that the problem of finding the least cost enhancement of existing networks to satisfy even some special types of connectivity restrictions is NP-hard.

LeBlanc and Reddoch (1990) introduce two combined topology/ capacity planning and routing models for reliable backbone network design. These models are similar to our approach, but differ in two respects: they (i) include an explicit (non-linear) delay cost in the objective function, and (ii) assume continuous capacities, and linear capacity costs (our model includes an additional fixed cost for installing links). The first model considers traffic with different priorities, and finds the least (delay + capacity) cost design containing alternate routes for single link failures. The second model uses hop constraints as a surrogate for reliability. Both models permit bifurcated routing. The authors exploit the linear capacity cost structure to solve problems with up to 100 nodes using the flow deviation algorithm.

Next, we introduce some notation, state our modeling assumptions, and present a mixed-integer programming formulation for the backbone network design problem with hop constraints.

2.1 *Notation and Modeling Assumptions*

The hop-constrained network design problem is defined over an undirected network G:(N,E) whose vertices N represent the backbone nodes and intermediate switches; the edges in E correspond to possible direct connections. For every pair of nodes p, q in the network, let d_{pq} represent the projected peak internode traffic (or *demand*) from p to q. We treat the traffic between each pair of nodes as a separate *commodity*, with <p,q> denoting the commodity that flows from node p to node q. Let K denote the set of all commodities, i.e., $K = \{<p,q> : p,q \in N \text{ with } d_{pq} > 0\}$. For each commodity <p,q>, the designer specifies an upper limit, denoted as h_{pq}, on the number of hops for messages originating at p and destined for q. We refer to h_{pq} as the (maximum) hop parameter for commodity <p,q>. Section 2.3 describes a method for selecting these hop parameters.

To establish transmission capacity on each link {i,j} of the network, we incur two types of (investment and operating) costs: a *fixed cost*, denoted as F_{ij}, and a *variable cost* c_{ij} that represents the cost per unit of capacity from i to j. The fixed cost might consist of investments for acquiring land, building the infrastructure, and installing/operating the communication link, while the variable cost component approximates expenses for acquiring and maintaining transmission and switching equipment. In Section 5 we indicate how these fixed and variable costs can also account for fixed or proportional buffer capacities on each link. Our model also accommodates economies of scale and volume discounts in the form of piecewise-linear, concave costs (see Figure 1); we assume the simpler fixed plus variable cost structure for expositional ease. As LeBlanc and Reddoch (1990) note, even though transmission cables and switches can be purchased and leased only in discrete sizes, telecommunication companies often have the option of using public telecommunication facilities (and even leasing fractional capacities) for overflow traffic; hence, piecewise linear, concave functions adequately represent capacity costs.

Our model includes an explicit design constraint specifying that the selected network topology must be connected. This restriction, though not essential for our algorithm, strengthens the problem formulation and improves the lower bound. For certain problem contexts, when the commodity flow pattern does not necessitate a connected solution, the following network transformation ensures connectedness. Consider a "commodity graph" containing the original nodes, and edges $\{p,q\}$ for every commodity $<p,q>$ with $d_{pq} > 0$. Let v be the number of connected components in this graph. If $v = 1$, we say the demand pattern is 'complete'; in this case, every feasible design must necessarily be connected, and our connectedness restriction is valid. When demand is not complete (i.e., if $v > 1$), we can augment the original network as follows to create a complete demand pattern: introduce a dummy node, say, node 0, and add v dummy edges $\{0,i_r\}$ connecting node 0 to one selected node i_r in every component $r = 1,2,...,v$ of the commodity graph. Each dummy edge has zero fixed cost and a very high variable cost. Add v dummy commodities $<0,i_r>$, each with unit demand and a hop restriction of 1. Observe that the commodity graph for the augmented network has a single component; and, the optimal design for the original problem together with the dummy edges $\{0,i_r\}$ is optimal for the transformed problem. Thus, augmenting the network makes the connectedness assumption valid. Finally, the network transformation is unnecessary if the planner wishes to explicitly impose connectedness (for instance, to facilitate alternate routing) even with incomplete demand; in this case, our algorithm generates a higher lower bound relative to the 'unrestricted' lower bound obtained using the augmented network. For our computational tests, a random choice of commodity origins and destinations resulted in complete demand for all test networks (even with sparse demand patterns); we, therefore, did not augment the networks.

2.2 *Mathematical Programming Formulation*

Our mixed-integer programming formulation for the backbone network design problem distinguishes the direction of flow on each edge of the original undirected network G. We consider two *directed* **arcs**, denoted as (i,j) and (j,i), corresponding to each original *undirected* **edge** {i,j}. The original set of undirected edges is denoted as E; let A represent the corresponding set of directed arcs. Our formulation uses two sets of binary decision variables, y_{ij} and z_{ij}^{pq}, defined as follows:

$$y_{ij} = \begin{cases} 1 & \text{if edge } \{i,j\} \text{ is included in the design,} \\ 0 & \text{otherwise, and} \end{cases}$$

$$z_{ij}^{pq} = \begin{cases} 1 & \text{if commodity } \langle p,q \rangle \text{ is routed on arc } (i,j), \\ 0 & \text{otherwise.} \end{cases}$$

The y variables model the *edge selection* decisions, while the z variables represent the (primary) *routing* decisions for each commodity.

The backbone design problem can now be represented mathematically as the following integer formulation called [HCDP] (for *hop-constrained design problem*):

$$\text{minimize} \quad \sum_{\{i,j\}} F_{ij}\, y_{ij} + \sum_{(i,j)} \sum_{\langle p,q \rangle} c_{ij}\, d_{pq}\, z_{ij}^{pq} \tag{2.1}$$

subject to:

$$\sum_{j \in N} z_{ij}^{pq} - \sum_{j \in N} z_{ji}^{pq} = \begin{cases} 1 & \text{if } i = p \\ -1 & \text{if } i = q \\ 0 & \text{if } i \neq p,q \end{cases} \quad \text{for all } \langle p,q \rangle \in K, \tag{2.2}$$

$$\sum_{(i,j)} z_{ij}^{pq} \leq h_{pq} \quad \text{for all } \langle p,q \rangle \in K, \tag{2.3}$$

$$z_{ij}^{pq} + z_{ji}^{pq} \leq y_{ij} \quad \text{for all } \langle p,q \rangle \in K, \text{ all } \{i,j\} \in E \tag{2.4}$$

$$y \in \tau \text{, and} \tag{2.5}$$

$$y_{ij},\ z_{ij}^{pq},\ z_{ji}^{pq} = 0 \text{ or } 1 \quad \text{for all } \langle p,q \rangle \in K, \text{ all } \{i,j\} \in E, \tag{2.6}$$

where τ = the set of all connected subnetworks of the original network G.

The objective function consists of minimizing the sum of the fixed design costs and the variable capacity costs. Note that we permit the variable costs c_{ij} to be asymmetric (i.e., different for the two directions on each edge); in fact, our subsequent solution algorithm applies even when these costs vary by commodity.

Constraints (2.2) ensure that, for each commodity <p,q>, the arcs (i,j) with $z_{ij}^{pq} = 1$ form a route from p to q. Constraint (2.3) represents the hop restriction for commodity <p,q>; it specifies that <p,q> must be routed over at most h_{pq} arcs. Inequalities (2.4) are forcing constraints that relate the routing and edge selection variables: commodity <p,q> can flow in either direction on edge {i,j} only if this edge is included in the topological design. Finally, constraint (2.5) specifies that the selected design must be a connected subnetwork of the original network G. Our solution method does not require an explicit mathematical representation of this constraint.

To incorporate piecewise linear, concave capacity cost functions, we replace each edge {i,j} of the original network with r_{ij} parallel edges connecting i and j, where r_{ij} denotes the number of linear segments in the expansion cost function for edge {i,j} (see Figure 1). Let F_{ij}^r and c_{ij}^r denote, respectively, the y intercept and the slope of the r^{th} segment; the r^{th} parallel edge in our augmented network carries a fixed charge of F_{ij}^r and a variable cost of c_{ij}^r. Because the cost function is concave, the model does not require explicit capacity constraints to enforce the upper limit of flow for each cost range.

2.3 *Setting the Maximum Hop Parameters*

The backbone network design model's primary role is to generate various cost minimizing solutions as the hop restrictions are parametrically changed. We propose one possible scheme, called the *Demand-based Hop Restriction Method*, to systematically vary the hop restrictions. For each commodity <p,q>, this method selects the hop parameter h_{pq} from a user-specified window or range of permissible hop parameter values. Let h_1 and h_2 ($\geq h_1$) denote, respectively, the smallest and largest desired hop parameter. Within the range $[h_1, h_2]$, our method selects lower h_{pq} values for high-demand commodities, and vice versa. In particular, we set $h_{pq} = h_1$ for the commodity <p,q> with the largest demand (among all commodities), and $h_{pq} = h_2$ for the commodity with the lowest demand. For commodities with intermediate demand values, the hop parameters are calculated by linear interpolation (rounded down to the next lower integer values) in the range $[h_1, h_2]$. By imposing more stringent hop restrictions for commodities with higher demand, the hop restriction method gives greater importance to these commodities in determining the network configuration. In practice, designers may use other criteria and methods to select and systematically vary the hop parameters for each commodity.

Observe that as h_1 and h_2 decrease, the hop constraints become tighter, and the resulting network designs should become more dense. Also, when $h_1 = h_2 = h_0$, say, the method selects the same hop parameter value h_0 for all commodities. We refer to this special case as the *Uniform Hop Restriction Method.* Section 4.3, presents computational results to illustrate the effect of various hop parameter settings using the general (commodity-dependent) and uniform hop restriction methods. Next, we discuss a Lagrangian-based algorithm for generating good upper and lower bounds on the optimal value of [HCDP].

3. Solving the Hop-constrained Network Design Problem

The hop-constrained design problem generalizes the fixed-charge network design problem which itself is known to be NP-hard (Johnson, Lenstra and Rinnooy Kan (1978)). Finding the optimal hop-constrained solution is, therefore, a difficult computational task. In this section, we develop a Lagrangian-based solution method that simultaneously generates good feasible solutions as well as lower bounds on the optimal cost, for any given set of hop parameters. These bounds provide performance guarantees so that the user can assess the quality of the heuristic solutions.

3.1 Lagrangian Relaxation Scheme

The Lagrangian relaxation method has been successfully applied to several difficult discrete optimization problems (see, for example, Geoffrion (1974), Fisher (1981)). The method exploits special structure in the problem formulation by dualizing the complicating constraints, and solving the resulting subproblems using efficient, specialized algorithms. For a given set of Lagrange multipliers, the cost of the optimal Lagrangian solution serves as a lower bound on the optimal cost of the original problem. The Lagrangian subproblem solutions also provide useful information to construct feasible designs for the original problem. The gap between the Lagrangian lower bound and the cost of the best feasible solution measures the maximum possible cost differential between the optimal and heuristic solutions. To generate good lower bounds, the Lagrange multipliers are changed iteratively using techniques such as subgradient optimization or dual ascent (see, for example, Fisher (1981), Balakrishnan et al. (1989)).

For the hop-constrained design problem, we consider a Lagrangian relaxation scheme that dualizes the forcing constraints (2.4) using nonnegative Lagrangian multipliers μ_{ij}^{pq} for all edges {i,j} and all commodities <p,q>. Observe that when constraints (2.4) are removed from formulation [HCDP], the problem decomposes into two main subproblems: a ***Routing*** subproblem denoted as [RSP(μ)], and an ***Edge Selection*** subproblem denoted as [ESP(μ)]. The routing subproblem determines the optimal values of the route selection variables z_{ij}^{pq}, while the edge selection subproblem calculates the values of the design variables y_{ij}. The routing subproblem further decomposes by commodity; we denote the subproblem corresponding to commodity <p,q> as [RSP$_{pq}$(μ)]. The next two sections describe these subproblems in greater detail, and discuss methods to solve them.

3.2 Routing Subproblem

For any given set of Lagrange multipliers $\mu = \{\mu_{ij}^{pq}\}$, we define an *adjusted* variable cost for each commodity <p,q> and every arc (i,j) as follows:

$$c_{ij}^{pq} \quad \triangleq \quad c_{ij} + \mu_{ij}^{pq}/d_{pq} \qquad \text{for all } (i,j) \in A, \text{ all } <p,q> \in K. \quad (3.1)$$

Then, the routing subproblem [RSP$_{pq}$(μ)] corresponding to commodity <p,q> has the following formulation:

[RSP$_{pq}$(μ)]

$$L_{pq}(\mu) \quad = \quad \min \quad d_{pq} \sum_{(i,j)\in A} c_{ij}^{pq} z_{ij}^{pq} \qquad (3.2)$$

subject to

$$\sum_{j \in N} z_{ij}^{pq} - \sum_{j \in N} z_{ji}^{pq} \quad = \quad \begin{cases} 1 & \text{if } i = p, \\ -1 & \text{if } i = q, \text{ and} \\ 0 & \text{otherwise}, \end{cases} \qquad (3.3)$$

$$\sum_{i} \sum_{j} z_{ij}^{pq} \quad \leq \quad h_{pq}, \text{ and} \qquad (3.4)$$

$$z_{ij}^{pq} \quad = \quad 0 \text{ or } 1 \quad \text{for all } (i,j) \in A. \qquad (3.5)$$

Observe that subproblem [RSP$_{pq}$(μ)] essentially seeks the shortest directed path from node p to node q containing a maximum of h_{pq} arcs; the adjusted variable costs c_{ij}^{pq} serve as arc lengths for this *hop-constrained shortest path problem.* The optimal value $L_{pq}(\mu)$ of the routing subproblem equals d_{pq} times the length of the shortest hop-constrained path.

We solve the hop-constrained shortest path subproblem using a truncated version of the method of successive approximations (Lawler (1976)). Let u_j^h denote the length of the shortest path from node p (the origin) to node j containing *at most* h arcs. Initially, $u_j^1 = c_{pj}^{pq}$ if $(p,j) \in A$; otherwise $u_j^1 = \infty$. At the end of stage h, values of u_j^h are known for all nodes j in the network. During stage (h+1), we use the following expression to compute u_j^{h+1} for all $j \in N$:

$$u_j^{h+1} = \min \{u_j^h, \min [\, u_i^h + c_{ij}^{pq} : i \in N, (i,j) \in A \,] \}. \tag{3.6}$$

The first term in the right-hand side of equation (3.6) is the length of the shortest path from p to j containing at most h arcs. The second term represents the length of the (h+1)-arc shortest path from p to j, which must consist of a h-arc shortest path from p to i, for some intermediate node i, plus the arc from i to j.

To solve subproblem $[RSP_{pq}(\mu)]$, we terminate the algorithm after h_{pq} stages; the method's computational complexity is $O(n^2 h_{pq})$. The value u_q^h obtained at the final stage (i.e., with $h = h_{pq}$) gives the length of the shortest hop-constrained path from p to q. We can trace the actual path for commodity <p,q> by performing the usual backtracking procedure.

3.3 Edge Selection Subproblem

Let us now consider the second Lagrangian subproblem involving the edge selection variables y_{ij}. Given a set μ of Lagrangian multipliers, let

$$\tilde{F}_{ij} \quad \triangleq \quad F_{ij} - \mu_{ij}^{pq} \qquad\qquad \text{for all } \{i,j\} \in E, \tag{3.7}$$

represent the *adjusted* fixed cost for edge {i,j}. The edge selection subproblem involves selecting a subset of edges that connects all nodes of the network at minimum total (adjusted fixed) cost. We solve this subproblem as follows: First, select all edges with negative adjusted fixed costs, and arrange the remaining edges in order of nondecreasing adjusted fixed cost. At each stage we refer to the subnetwork defined by the currently selected edges as the *current* network. Consider each edge {i,j} in sequence from the sorted list of unselected edges. If edge {i,j} connects two different components of the current network, select it and update the current network by merging the two components; otherwise, discard edge {i,j}. Repeat the procedure for all edges in the list. At termination, the final network must be connected. The sum of the adjusted fixed costs in this network gives the optimal value of the edge selection subproblem. The computational effort required to solve this subproblem is dominated by the edge sorting procedure which requires

$O(|E| \log |E|)$ operations. Note that, if formulation [HCDP] does not contain the connectedness condition (2.5), the edge selection subproblem is easily solved as follows: set $y_{ij} = 1$ if $\tilde{F}_{ij} \leq 0$, and 0 otherwise, for all edges $\{i,j\} \in E$; this solution may have a lower value than the connected subproblem solution.

The optimal value $\underline{Z}(\mu)$ of the complete Lagrangian subproblem is the sum of (i) the edge selection subproblem value, and (ii) the optimal values $L_{pq}(\mu)$ of the route selection subproblems for all commodities $<p,q>$. For any given vector μ of nonnegative multipliers, $\underline{Z}(\mu)$ is a lower bound on the optimal value of the original subproblem. To obtain good lower bounds, we use a subgradient method (see, for instance, Held, Wolfe, and Crowder (1974) or Fisher (1981)) to iteratively adjust the Lagrange multipliers.

We note that our Lagrangian relaxation scheme does not satisfy the *integrality* property (Geoffrion (1974)); for example, the optimal solution to the linear programming relaxation of the routing subproblem might possibly be fractional. Therefore, the best lower bound obtained using our relaxation scheme may exceed the linear programming lower bound for formulation [HCDP].

3.4 Lagrangian-based Heuristic Procedure

At each subgradient iteration, the solution to the routing subproblem provides a set of feasible routes (that satisfy the hop restrictions) for all commodities. Note that the edge selection subproblem may include some edges that do not belong to any of the selected routes and/or may omit some edges belonging to these routes. We, therefore, ignore the solution to the edge selection subproblem, and use only the routing solution to construct an initial feasible design. Our Lagrangian-based initial design consists of all edges belonging to the routes chosen in the routing subproblem. For this design, we solve the hop-constrained shortest path problem for each commodity using the *original* variable costs c_{ij} as arc lengths. The sum of the fixed costs for the selected arcs and the (true) variable costs for all commodities gives the cost of the heuristic solution.

We then attempt to improve this starting solution by applying a *Drop* heuristic (Billheimer and Gray (1973)). The drop procedure is a local improvement method that iteratively eliminates existing edges from the current design in order to reduce the total cost. When we drop an edge, the total fixed cost decreases while the variable costs might increase since some commodities must now flow on more expensive alternate routes. At every stage, the procedure evaluates the net savings

Δ_{ij} obtained by dropping each edge {i,j} from the current design. If the net savings is zero or negative for all edges, the procedure terminates. Otherwise, it drops an edge with positive net savings, updates the current design and commodity routings, and decreases the upper bound.

Evaluating the revised routing costs when an edge is dropped entails solving a hop-constrained shortest path problem for each commodity that previously used this edge. Since this computation can be time-consuming, we do not evaluate the savings for every edge at each iteration. Instead, we maintain and iteratively update a *candidate list* of edges to evaluate. Initially, this list contains all edges that yield net savings in the first iteration. In subsequent iterations, we only evaluate the savings for edges that belong to this list. The edge with the maximum savings is dropped from the current design and removed from the list; edges that do not give any net savings are also deleted from the list. When the list becomes empty, we reinitialize it by evaluating the savings for all edges in the current design.

The local improvement heuristic can be applied at the end of each subgradient iteration. However, to reduce computation time, we only apply the drop procedure intermittently, say, once every 100 iterations or if the current starting design has a lower cost than the previous best starting design. The next section describes our computer implementation of the Lagrangian-based solution procedure, and presents extensive computational results for several randomly generated test problems.

4. Computational Results

We implemented the Lagrangian-based algorithm for the hop-constrained design problem in standard FORTRAN on an IBM 3083 (model BX) computer, and tested the algorithm using several randomly generated problems. This section first describes some features of our implementation, and the method we used to generate random test problems. In all, we applied the algorithm to over 65 problem instances. Our computational tests (discussed in Sections 4.3 and 4.4) address two different performance aspects. The first set of tests focuses on evaluating the performance of the algorithm, in terms of computation time and quality of the Lagrangian-based heuristic solutions, as a function of the problem size, cost structure, and hop restrictions. We then focus on a single problem instance in order to assess the impact on cost and performance of systematically varying the hop restrictions. This exercise demonstrates the effectiveness of our approach for generating several alternative base designs.

4.1 Implementation Details

In addition to the features described in Section 3, our implementation of the Lagrangian relaxation algorithm incorporates: (i) a method to generate an initial upper bound before initiating the subgradient procedure, and (ii) a non-standard multiplier initialization method.

Generating an Initial Heuristic Solution

To generate an initial upper bound, we first construct a feasible design using a method which we call the **BUILD** heuristic, and subsequently apply the drop procedure to improve this design. The Build heuristic first sorts all commodities in decreasing order of demand; it builds a feasible design by iteratively considering each commodity in the sorted order. At the beginning of stage k, let $E^{(k-1)}$ denote the set of edges in the current design. This design contains feasible paths (satisfying the respective hop constraints) for each of the first (k-1) commodities. During stage k, the algorithm augments this design to ensure that the k^{th} commodity, say, commodity <p,q> has a good feasible path. For this purpose, we solve a hop-constrained shortest path problem from node p to node q with arc lengths l_{ij} defined as follows:

$$\begin{aligned} l_{ij} &= c_{ij} && \text{if } (i,j) \in E^{k-1}, \text{ and} \\ &= c_{ij} + F_{ij}/d_{pq} && \text{if } (i,j) \notin E^{k-1}. \end{aligned} \qquad (4.1)$$

Essentially, if edge {i,j} already belongs to the current design, we assign only the variable cost c_{ij} to commodity <p,q>. Otherwise, we add the per unit cost F_{ij}/d_{pq} to the variable cost c_{ij}, effectively forcing commodity <p,q> to absorb the entire fixed cost of edge {i,j}. Let P^k denote the set of edges belonging to the shortest hop-constrained path from p to q using arc lengths l_{ij}. The current design is then augmented by adding to $E^{(k-1)}$ all the new edges of P^k, i.e., we set $E^k = E^{(k-1)} \cup P^k$. Thus, at the end of stage k, the current design contains feasible paths for the first k commodities in the sorted list. By considering commodities in decreasing order of demand, the Build heuristic gives greater importance to high-demand commodities (which contribute more to the variable cost) in developing the design. When the procedure terminates, the set of edges $E^{|K|}$ is a feasible design. As a final step, the algorithm reroutes all commodities over the respective shortest hop-constrained paths in $E^{|K|}$ using the original variable costs c_{ij} as arc lengths. We then apply the drop procedure (described in Section 3.4) to the starting solution constructed by the Build method. The improved solution provides the initial upper bound.

Initializing the Lagrange Multipliers

Instead of initializing the Lagrange multipliers to zero, we use the initial heuristic solution to determine the starting values for the multipliers. Let x_{ij} denote the total flow on each edge {i,j} of the initial heuristic solution. The multiplier μ_{ij}^{pq} is then set equal to F_{ij}/x_{ij} if commodity <p,q> uses edge {i,j} in the initial solution; otherwise, μ_{ij}^{pq} is initialized to zero. Thus, for each edge {i,j} in the initial design, the multiplier initialization method completely allocates the fixed cost F_{ij} to the Lagrange multipliers for all commodities that flow on this edge.

For all computational tests, our subgradient procedure: (i) initializes the step size factor (see, for instance, Held et al. (1974)) to 2; (ii) halves the step size factor if the lower bound does not improve for 15 consecutive iterations; and (iii) terminates after 250 subgradient iterations (or earlier if the gap between the current best upper and lower bounds reduces to a very small value).

4.2 *Random Problem Generation*

We implemented a random problem generator to construct a wide range of test problems for the Lagrangian relaxation algorithm. The problem generator can create test problems with different sizes, edge and commodity densities, cost structures and hop restrictions. To generate a test problem, the user must first specify the following parameters: (i) seed for random number generator; (ii) **network size** information: number of nodes n, edges m, and commodities $|K|$; (iii) **cost structure** information: ***weight*** w for the random component in edge costs, and ***fixed-to-variable cost ratio*** r (these parameters are explained later); (iv) average **demand** $\bar{d}$; and (v) **hop-restriction** information: the lower and upper limits, h_1 and h_2, defining the range of desired hop parameters.

The program first locates the required number of nodes randomly on a 1000 x 1000 grid, and generates a random spanning tree over these nodes (to ensure problem feasibility). It then adds the required number of additional edges (i.e., m-(n-1) edges) randomly to the tree. Having constructed the underlying network G:(N,E), the problem generator calculates the fixed and variable costs for each edge in G. Let e_{ij} denote the Euclidean distance between nodes i and j. The fixed cost F_{ij} of edge {i,j} is then computed as:

$$F_{ij} \quad = \quad (1 - w)\, e_{ij} + w\, \zeta\,, \qquad (4.2)$$

where w is the user-specified weight $(0 \leq w \leq 1)$, and ζ is a random number derived from a uniform distribution with range [0,1000]. Observe that, by varying w, the user can generate cost values that are either completely random (when w = 1) or directly proportional to Euclidean distance (when w = 0); intermediate values of w give a mixture of random and Euclidean costs. We refer to w as the *cost randomness factor*. Note that when w > 0, the fixed costs may not satisfy the triangle inequality.

The variable cost c_{ij} for edge {i,j} is obtained by dividing the fixed cost F_{ij} by the user-specified *fixed-to-variable cost ratio* r. When r is very small, the fixed costs are negligible compared to the variable costs; in this case, the union of the hop-constrained shortest paths (using the variable costs as arc lengths) for all commodities gives the best design. At the other extreme, when r is very large, the variable costs are negligible, and the best solution is a network with the smallest total fixed cost that contains at least one hop-constrained path for every origin-destination pair. In particular, when h_{pq} = (n-1) for all commodities <p,q>, the optimal solution for large values of r is the minimal spanning tree.

After calculating the edge costs, the problem generator randomly identifies the required number (|K|) of origin-destination pairs which define the commodities. For each commodity, the demand d_{pq} is selected from a uniform distribution ranging from 0 to $2\bar{d}$; we set $\bar{d}$ = 5 units in all cases. For all our test problems (even with |K| less than 50% of the maximum possible number of commodities), the random choice of origin-destination pairs resulted in complete demand patterns (as defined in Section 2.1).

Finally, the program generates the hop parameters for each commodity using the demand-based method described in Section 2.3, i.e., by interpolation in the range $[h_1, h_2]$, with high-demand commodities having lower hop parameter values and vice versa. Given a set of hop parameters, our implementation first checks for problem feasibility. If the network does not contain any feasible path for some commodity, the user must either increase the hop parameter(s) or try a different random number seed.

The next two sections describe the results of our computational experiments to assess the effectiveness of the Lagrangian relaxation algorithm, and generate different cost minimizing designs by varying the hop restrictions.

4.3 *Testing the Performance of the Lagrangian Relaxation Algorithm*

We use two performance measures to evaluate the effectiveness of the Lagrangian relaxation algorithm: (i) the *% gap*, defined as the difference between the best upper and lower bounds expressed as a percentage of the best lower bound, which measures the quality of the heuristic solutions, and (ii) the ***computation time*** (in CPU seconds) required for the entire procedure.

Our computational tests attempt to evaluate the effect of three problem characteristics - problem size, cost structure, and hop restrictions - on the algorithmic performance measures (% gap and computation time). *Problem size* depends on the number of nodes in the network, the number (or density) of edges, and the number of commodities. The *cost structure* is determined by two factors: the *cost randomness factor* **w**, and the *fixed-to-variable cost ratio* **r**. The third problem characteristic affecting algorithmic effectiveness is the tightness of the *hop restrictions* measured, for instance, by $(h_1 + h_2)/2$ (the mid-point of the user-specified window). For convenience, in this section we consider only the uniform hop restriction method, with $h_1 = h_2 = h_0$; results for the more general hop restriction method (with $h_1 < h_2$) are reported in Section 4.4.

4.3.1 Effect of Cost Structure and Hop Restrictions

To isolate the effect of each of the three problem characteristics on algorithmic performance, we first consider different cost structures (i.e., different cost randomness factors and fixed-to-variable cost ratios) and hop restrictions for a 20-node network with 180 edges and 180 commodities. For this network size, we generated 3 different problem instances (using different random number seeds) with varying cost randomness factors, namely, w = 0, 0.5, and 1. For each network, we applied the Lagrangian relaxation algorithm with three fixed-to-variable cost ratios, r = 10, 20, and 50, and three different (uniform) hop parameter values, h_0 = 3, 4, and 5 hops. This set of parameters represents a wide spectrum of possible problem structures, ranging from Euclidean costs to completely random costs, and moderate to high fixed costs (relative to the variable costs).

Table I summarizes the results of our analysis. For each combination of w, r, and h_0, this table shows: (i) the fixed cost as a percentage of total cost in the final heuristic solution (denoted as *% FC*); (ii) the number of edges included in the final design (***# of edges***); (iii) the weighted average number of hops $\bar{h}$ (***ave. hop***) over all commodities, which is defined as

$$\bar{h} \quad = \quad \sum_{<pq>} d_{pq} h'_{pq} / \sum_{<pq>} d_{pq},$$

where h'_{pq} ($\leq h_{pq}$) is the actual number of hops for commodity <p,q> in the final heuristic solution; (iv) the gap between the final upper and lower bounds expressed as a percentage of the best lower bound (*% **gap***); and (v) the total CPU time (in seconds on an IBM 3083) for generating heuristic solutions and subgradient optimization (*CPU*). The first three statistics describe the structure of the final heuristic solution, while the last two measure the algorithm's effectiveness.

As expected, the results in Table I show that the % FC and % gap increase while the number of edges in the final design decreases as the fixed-to-variable cost ratio increases from 10 to 50. For ratios of 10 and 20, all gaps are less than or equal to 5%. For one problem instance in this category, the algorithm generated the optimal solution and proved its optimality (by generating a lower bound equal to the upper bound); for 8 other instances, the gap was less than 1%. With a fixed-to-variable cost ratio of 50, the gaps are larger (ranging from 4 to 23%). These latter problems represent extreme values of fixed costs, and might have inherently higher duality gaps. For consistency, all the % gaps reported in Table I correspond to the difference between the best upper and lower bounds at the end of 250 subgradient iterations. (The gaps might possibly be lower if we increased the limit on the number of subgradient iterations. For example, for the test problem with $h_0 = 3$ and $r = 50$, we were able to reduce the 20% gap to 17% by applying the subgradient procedure for 100 more iterations; however, the CPU time increased by 75%.)

As the hop restrictions become tighter (i.e., when h_0 decreases from 5 to 3), the final design becomes more dense as expected, but the problems also become more difficult to solve (the % gap increases). Interestingly, the % gap does not exhibit any consistent pattern of variation as the cost randomness factor changes from 0.0 (Euclidean costs) to 1.0 (completely random costs). The Build heuristic generates good initial upper bounds with very little computational effort (less than 2 seconds of CPU time in almost all instances). Applying the drop heuristic at intermediate stages of the subgradient procedure reduces the initial cost by 4.5% on average. The subgradient procedure requires around 70% of the total computational time, and improves the initial lower bound by an average of 30.6%.

4.3.2 Effect of Problem Size and Hop Restrictions

In Section 4.3.1, we fixed the problem size and considered the effect of cost structure and hop restrictions on algorithmic performance. We now study the effect of problem size on the quality of bounds and computation time. For this set of tests, we fixed the cost randomness factor at 0.5, and the fixed-to-variable cost ratio at 20. We considered four basic network sizes containing 10, 20, 25 and 30 nodes, respectively. For each network size, we generated sparse as well as dense networks (with correspondingly sparse and dense demand patterns). Table II reports the performance of the Lagrangian relaxation algorithm for 8 test networks with different sizes, for various (uniform) hop restrictions.

Again, the % gaps decrease as the hop parameter increases. In all but three instances, the gaps reduce to below 10% within 250 iterations. The % gaps do not show any consistent variation as the network size and density increase; for the 10-node problem the % gap is higher for the denser network, while the converse is true for the 20-node problem. As expected, the computation times increase with problem size.

In summary, our computational results suggest that the Lagrangian-based algorithm is quite effective. For almost all problems with fixed-to-variable cost ratios of up to 20, the method generates tight lower bounds and good heuristic solutions that are generally guaranteed to be within 10% or less from the optimal solutions. We emphasize that many of our test problems are very difficult to solve optimally. For instance, the integer programming formulation corresponding to the 30-node, 420-edge test problem contains 353,220 integer variables, and 542,640 constraints; this problem size is well beyond the realm of capabilities of current state-of-the-art, general integer programming methods.

4.4 The Network Cost-Performance Tradeoff

The results of the previous section show that the Lagrangian-based algorithm is effective in generating good cost-minimizing solutions for the hop-constrained design model for a wide range of problem structures. This section attempts to demonstrate that, by systematically varying the hop parameters, the hop-constrained model achieves our original objective of generating a variety of base network designs with different cost-performance characteristics. In general, evaluating the performance (i.e., reliability, blocking, delay, etc.) of each design would require detailed context-specific analyses and simulations which we did not implement. Instead, to illustrate the effect of hop constraints on service level, we use two sample performance metrics - average number of hops, and average connectivity.

The (weighted) **average number of hops** $\bar{h}$, defined in Section 4.3, might serve as a measure of operational (routing) complexity, with lower values corresponding to easier routing. Furthermore, under certain assumptions, the average delay per message $\bar{T}$ in a packet switched network is directly proportional to the average number of hops $\bar{h}$. (For example, if the traffic arrivals and service rates satisfy Kleinrock's assumptions, say, Possion arrivals and independent, exponentially distributed message lengths with mean $1/\mu$ on every link, and if each link has a fixed buffer capacity B, then $\bar{T} = \bar{h}/\mu B$.) Thus, $\bar{h}$ might serve as a rough indicator to assess changes in delay performance.

The (weighted) **average connectivity** statistic is a surrogate measure for network vulnerability. For a given commodity <p,q>, we define **connectivity** as the number of alternate arc-disjoint paths from p to q in the final design; it gives a lower bound on the number of arcs that must fail before all p-to-q communication is blocked. Using the relative demands for all commodities as the weights for their respective connectivity values gives average connectivity. Thus, providing multiple paths for high demand commodities is considered more important. Our implementation uses a maximum flow algorithm to compute the connectivity of each commodity in a given design. When each edge of the final design has a capacity of 1 unit, the maximum flow from p to q gives the number of alternate arc-disjoint paths for commodity <p,q>.

To study the effect of various hop restrictions on cost, average number of hops, and average connectivity, we consider a single problem instance with 25 nodes, 300 edges, and 300 commodities. Observe that this network has a complete topology (i.e., it contains one edge connecting every pair of nodes) and complete demand (i.e., the set of commodities K contains one commodity for each node pair). The cost randomness factor is fixed at 0.5, the average demand at 5 units, and the fixed-to-variable cost ratio at 20.

Table III shows the computational results for 22 different hop restrictions. The first 7 solutions correspond to uniform hop restrictions, with parameter h_0 ($=h_1= h_2$) ranging from 1 to 7; the other 15 solutions correspond to commodity-dependent hop restrictions with varying means and widths. For example, the hop restriction [2,2] corresponds to a uniform restriction of 2 hops for all commodities. On the other hand, the restriction [2,5] generates commodity-dependent hop parameters with a range width of 3 and mean of 3.5; in this case, high demand commodities have hop parameters close to 2 and low demand commodities may use up to 5 edges.

Since the data is randomly generated, the actual cost values for different final solutions are not very meaningful. Instead, we consider a *normalized cost index* that expresses the actual cost as a percentage of the lowest total cost over all 22 solutions (obtained with the least stringent hop restrictions). Thus, a design with a cost index of 140 is 40% more expensive than the lowest cost design. Using this normalized cost criterion facilitates our assessment of the cost-benefit tradeoff as the hop restrictions vary.

As expected, the results of Table III show that total cost and average connectivity increase, while the average number of hops decreases as the hop constraints become more restrictive. In particular, with a uniform hop parameter h_0 of 1 (solution # 1), the design must contain all edges of the original network, every commodity must have a single-hop route, and each commodity has 24 alternate arc-disjoint paths (including the single-hop route). For our problem instance, this design is also the most expensive because of the significant fixed costs. As we relax the hop constraints, the designs become less expensive while average connectivity declines. The lowest cost design corresponds to the least restrictive (uniform) hop restriction with $h_0 = 7$.

Observe that, as we increase the hop parameters beyond a certain threshold, the same final design remains optimal. For example, in the uniform hop case, the solution is unchanged when h_0 increases from 6 to 7. Similarly, with commodity-dependent hop parameters, the same final design satisfies the hop restrictions [3,6] and [3,7]. Finally, we note that for all but one instance the % gap is less than 6.5%, implying that the costs shown in Table III closely approximate the optimal costs for different hop restrictions.

To better assess the cost versus performance tradeoff, Figure 2 shows a graphical display of the average number of hops and normalized cost for different solutions. (Solution # 1 is not shown in this figure since it has a normalized cost of over 400, while the remaining solutions have normalized costs ranging from 100 to 150.) As this figure illustrates, the method generates several 'dominated' solutions. For instance, solution # 2 (with hop restriction [2,2]) dominates solution # 10 (with hop restriction [1,4]) since the latter design has higher cost and higher average number of hops (and lower connectivity). Similarly, solution # 3 dominates solution # 18. The set of undominated solutions define an "efficient" frontier that characterizes the cost-benefit tradeoff when the average number of hops changes. The piecewise linear curve in Figure 2 shows, for instance, that reducing the average number of hops from 1.74 (solution # 2) to 1.71 (solution # 9) increases the total cost by over 16% (relative to the lowest cost design).

To summarize, our hop-constrained network design model helps to identify a selected subset of good solutions with varying performance characteristics. A pictorial representation such as Figure 2 can greatly assist the network designer in assessing the tradeoffs between conflicting criteria before selecting an appropriate backbone network topology. We emphasize that we have used average connectivity and average number of hops as measures of service performance only for illustrative purposes. In general, similar tradeoff analyses can be performed using any desired performance measure.

5. Model Variants and Extensions

In Section 2 we presented a basic version of the hop-constrained design model that selects primary routes for each commodity and provides adequate capacity on these routes. This section briefly describes how to adapt and extend the model to plan buffer capacities and select alternate routes.

5.1 Planning buffer capacities

Buffer capacities improve service levels by decreasing delays, and accomodating contingency traffic. We briefly outline two methods to judiciously allocate these capacities - a two-step process and an integrated model.

5.1.1 Two-step process

Planning buffer capacities might be viewed as the second step of the hierarchical process described in Section 1. In this scheme, the hop-constrained model first generates several network topologies and corresponding base capacities. For each topology, a subsequent step augments capacities on selected links to improve performance. This second step might, for instance, use a model similar to the approach proposed by Kleinrock (1976). Kleinrock's model selectively allocates buffer capacities to links in the base design in order to minimize average delay per message subject to a budget restriction. In particular, under certain assumptions about traffic arrivals, message sizes, and link expansion costs, Kleinrock proposes a "square-root" formula for optimally allocating a budgeted amount D_e to reduce average delay. Bitran and Tirupati (1988) propose (in the manufacturing context) enhancements and variants of this model to improve the performance of queueing networks. For instance, they describe models to minimize the total additional investment required to achieve a prespecified delay target, and to reallocate capacities from one part of the network to another for reducing delays.

5.1.2 Integrated models

Instead of applying a separate buffer allocation model as the second step, the hop-constrained model can itself be easily modified to account for certain types of buffer capacities. The modifications involve changing the fixed and variable cost parameters (F_{ij} and c_{ij}) and the demand values d_{pq}. We describe the model enhancements for three different types of buffers - fixed buffers, proportional buffers, and commodity-dependent buffers. Designers might use any combination of these three buffer types.

Consider first a *fixed buffer* policy which specifies that each selected link {i,j} of the network must have a (fixed) spare capacity of b_{ij} units. The hop-constrained model can incorporate this requirement by using a higher fixed cost; in particular, we use an inflated fixed cost $\hat{F}_{ij}$ which is equal to the original fixed cost F_{ij} plus $b_{ij}*c_{ij}$ (the second terms is the incremental cost of the required buffer capacity).

A *proportional buffer* policy specifies the required buffer level on each link as a proportion of the traffic volume on that link. Suppose we specify a proportion β_{ij} for each link {i,j}, i.e., {i,j} must have spare capacity of at least $\beta_{ij}*X$ units if it carries a flow of X units. To represent this requirement in the hop-constrained design model, we use inflated variable costs $\hat{c}_{ij} = (1+\beta_{ij})*c_{ij}$, where c_{ij} is the original cost per unit of capacity on link (i,j). In fact, since the hop-constrained model can also handle piecewise-linear concave costs, we can represent *capacity-dependent proportional buffers*, where the incremental buffer proportion, say, $\gamma_{ij}(X)$ declines as the volume of flow X increases. This type of buffering strategy is consistent with Kleinrock's (1976) observation that, as throughput increases, a smaller buffer proportion suffices to maintain the same delay performance. In general, we can approximate any concave buffer capacity requirement (as reflected, for instance, in Kleinrock's square root law for buffer allocation) with a piecewise linear function in the hop-constrained model.

Finally, suppose the planner specifies *commodity-dependent buffer* requirements as follows: the selected route for each commodity <p,q> must have excess capacity equal to a fraction δ_{pq} of the projected demand d_{pq}. The hop-constrained model can accomodate this requirement by using a modified demand value of $d_{pq}*(1+\delta_{pq})$ for each commodity <p,q> (or equivalently a modified variable cost for each commodity).

The hop-constrained formulation's ability to model fixed buffers, volume-dependent proportional buffers, and commodity-dependent buffers provides a powerful set of capabilities to the network planner. Just as we parameterize the hop constraints to generate various network configurations, we can also parametrically change the buffer specifications (i.e., change the parameters b_{ij}, β_{ij}, γ_{ij}, and δ_{pq}). For each specification, the hop-constrained model identifies a cost-effective design that serves as the basis for secondary enhancements and detailed performance evaluation.

5.2 *Modeling alternate routes*

The hop constraints indirectly influence the density of the topological design, and hence the availability of alternate routes for various commodities. To plan (contingency) capacities on these alternate routes, we can employ the buffering strategies described earlier and/or explicitly model the contingency flows. As we illustrate next, the latter approach requires additional decision variables and constraints in the hop-constrained formulation. Fortunately, the decomposition method proposed in Section 3 also solves the relaxation of this enhanced model.

Suppose we wish to select a design that has adequate capacity to accomodate at least a prespecified fraction of various demands along alternate routes. In this scheme, we select (as before) a *primary* route for each commodity; this primary route is the preferred communication path whenever all its links are operational. We must also select and plan capacities for a *secondary* route that will be used when one or more links on the primary path fail (or are highly congested). For purposes of this discussion, we will consider primary and secondary routes that are arc-disjoint, i.e., they do not have any links in common. Also, for convenience, we assume that the secondary route has no hop restrictions. Let ϕ_{pq} $(0 \le \phi_{pq} \le 1)$ be the fraction of the demand d_{pq} that the secondary route must accomodate.

To incorporate alternate routing, we change the original hop-constrained formulation as follows:

- add a new set of binary routing variables, say, w_{ij}^{pq} for each commodity <p,q>. Let $w_{ij}^{pq} = 1$ if commodity <p,q> uses (directed) arc (i,j) for its secondary route, and 0 otherwise;
- include the w-variables in the forcing constraints (2.4) of formulation [HCDP] as follows:

$$z_{ij}^{pq} + z_{ji}^{pq} + w_{ij}^{pq} + w_{ji}^{pq} \le y_{ij} \quad \text{for all } <p,q> \in K, \text{ all } (i,j) \in E. \qquad (2.4')$$

  The new forcing constraints specify that edge {i,j} must be included in the design if it belongs to either the primary or secondary route for commodity <p,q>. Since y_{ij} is at most 1, edge {i,j} cannot simultaneously belong to both the primary and secondary routes; thus, constraints (2.4') ensure that these two routes are arc-disjoint; and

- add the cost term $\sum_{(i,j)} \sum_{<p,q>} c_{ij} \phi_{pq} d_{pq} w_{ij}^{pq}$ to the objective function to account for the spare capacity on the secondary route.

To solve this enhanced formulation, we again dualize the forcing constraints (2.4'). The Lagrangian problem decomposes into an edge selection subproblem and a routing subproblem for each commodity. The edge selection subproblem is the same as before, while the routing subproblems now involve determining both the z and w variables. To find the optimal values of the z-variables we use the hop-constrained shortest path algorithm as before; a regular shortest path solution provides the optimal w-values (since we assumed that secondary routes do not have hop constraints). Observe that the Lagrangian subproblem solution may not necessarily select exclusive arcs for the primary and secondary paths (since we have dualized constraints (2.4') which ensure arc-disjoint paths). Thus, to obtain a feasible starting solution for local improvement, we might: (i) use the z-solution to select the primary route for each commodity, and (ii) generate alternate routes by applying a shortest path algorithm (perhaps, using some information from the w-solution) to the residual network when links of the primary route are removed.

We might also consider a stronger relaxation that involves simultaneously generating the hop-constrained primary routes and arc-disjoint secondary routes to minimize total variable costs. Effectively, the problem formulation contains an additional set of strengthening constraints

$$z_{ij}^{pq} + z_{ji}^{pq} + w_{ij}^{pq} + w_{ji}^{pq} \leq 1 \qquad \text{for all } <p,q> \in K, \text{ all } (i,j) \in E,$$

which are not dualized (and hence appear in the routing subproblem). We suspect that this arc-disjoint routing subproblem is not polynomially solvable, necessitating an enumerative technique to find the optimal routing. One interesting possibility is to develop a variant of the K-shortest path algorithm (see, for example, Dreyfus (1969)), modified to account for hop constraints on the primary routes.

6. Conclusions

In this paper, we have proposed a hop-constrained network design model to support long-term network planning decisions. For this model, we developed an effective solution method that can solve relatively large problems. By systematically varying the hop restrictions, we demonstrated the method's ability to generate several alternative cost-minimizing designs with different performance characteristics.

Using hop constraints as a surrogate for service restrictions greatly reduces the complexity of the model, and enables us to generate provably near-optimal solutions relatively quickly using an optimization-based approach. In contrast, representing delay restrictions explicitly requires numerous assumptions about traffic arrivals, service rates, failure rates, etc., and might possibly introduce non-linearities in the model. Modeling connectivity restrictions (for instance, the requirement that the network must be 3-connected) is equally complex. Consequently, models that explicitly incorporate both delay and connectivity restrictions can often be solved only by heuristic methods that do not provide any bounds on the quality of the solutions. Because we can solve the hop-constrained model effectively, our approach provides a useful design tool for performing sensitivity analyses. Similar models might also apply to other large-scale planning problems in logistics and manufacturing systems design.

Several extensions and variants of the basic model merit further investigation. First, it would be interesting to develop and test alternative multiplier adjustment schemes (such as dual ascent) and heuristic initialization and improvement methods to further improve the quality of the upper and lower bounds. Second, our model applies to the design of new networks that have no current transmission and switching capacity. Adapting the model and solution method for network expansion planning is an important next step. To account for an existing network, we require additional capacity constraints in the problem formulation; these constraints make the problem more difficult to solve and might adversely affect algorithmic performance. Finally, developing multi-period versions of the hop-constrained model is a fruitful area for further development. In the multi-period setting, demands for various commodities are specified for each period of the planning horizon, and network expansion costs change over time. The model must account for two types of tradeoffs: fixed design costs versus routing costs in each period, and expanding the network beyond current requirements in anticipation of future demand in order to exploit economies of scale. The basic hop-constrained network design model that we have developed in this paper can serve as the building block for these and other model extensions.

Acknowledgements: We would like to thank the referees and editors for their insightful comments and helpful suggestions to correct, clarify and improve the presentation.

References

AHUJA, V. 1979. Routing and Flow Control in Systems Network Architecture. *IBM Syst. J.* **18**, 298--314.

ASH, G. R., R. H. CARDWELL, and R. P. MURRAY. 1981. Design and Optimization of Networks with Dynamic Routing. *The Bell System Technical Journal,* **60**, 1787-1820.

BALAKRISHNAN, A., T. L. MAGNANTI and R. T. WONG. 1989. A Dual Ascent Procedure for Large Scale Uncapacitated Network Design. *Operations Research* **37**, 716-740.

BERTSEKAS, D. P. 1980. A Class of Optimal Routing Algorithms for Communication Networks. *Proc. 1980 Int. Conf. on Circuits and Computers,* Atlanta, Georgia.

BERTSEKAS, D. P. 1984. Second Derivative Algorithms for Minimum Delay Distributed Routing in Networks. *IEEE Trans. Communications* **COM--32**, 911--919.

BILLHEIMER, J., and P. GRAY. 1973. Network Design with Fixed and Variable Cost Elements. *Trans. Sci.* **7**, 49--74.

BONUCCELLI, M. A. 1981. Allocating Additional Link Capacities in Computer Communication Networks. IBM Res. Report RC 8967, Yorktown Heights, New York.

CANTOR, D. G., and M. GERLA. 1974. Optimal Routing in a Packet Switched Computer Network. *IEEE Trans. Computers* **C--23**, 1062--1069.

CHOU, W., F. FERRANTE and M. BALAGANGADHAR. 1978. Integrated Optimization of Distributed Processing Networks. *Nat. Comp. Conf.,* 795--811.

COURTOIS, P. J., and P. SEMAL. 1981. An Algorithm for the Optimization of Nonbifurcated Flows in Computer Networks. *Perform. Eval.* **1**, 139--152.

DREYFUS, S. E.. 1969. An Appraisal of Some Shortest Path Algorithms. *Oper. Res.* **17**, 395-412.

ESWARAN, K. P., and R. E. TARJAN. 1976. Augmentation Problems. *SIAM J. Comput.* **5**, 653--665.

FISHER, M. L. 1981. The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Man. Sci.* **27**, 1--18.

FRANK, H., and W. CHOU. 1971. Routing in Computer Networks. *Networks* **1**, 99--122.

FRATTA, L., M. GERLA and L. KLEINROCK. 1973. The Flow Deviation Algorithm: An Approach to Store--and--Forward Computer Communication Network Design. *Networks* **3**, 97--133.

GAVISH, B., and K. ALTINKEMER. 1987. Backbone Network Design Tools with Economic Tradeoffs. Working Paper. Krannert Graduate School of Management, Purdue University

GAVISH, B., and I. NEUMAN. 1986. Capacity and Flow Assignment in Large computer Networks. *Proc IEEE--INFOCOM 86*, 275--284.

GAVISH, B., and I. NEUMAN. 1987. Routing in a Network with Unreliable Components. Working Paper, Graduate School of Business, New York University, New York.

GEOFFRION, A. M. 1974. Lagrangean Relaxation and its Uses in Integer Programming. *Math. Prog. Study* **2**, 82--114.

GERLA, M. 1973. Deterministic and Adaptive Routing Policies in Packet Switched Computer Networks. Presented at the ACM--IEEE 3rd Data Communications Symposium, Tampa, Florida.

GERLA, M., H. FRANK, W. CHOU, and J. ECKL. 1974. A Cut Saturation Algorithm for Topological Design of Packet Switched Communication Networks. *Proc. Nat. Telecomm. Conf.* **NTC--74**, 1074--1085.

GERLA, M., and L. KLEINROCK. 1977. On the Topological Design of Distributed Computer Networks. *IEEE Trans. Communications* **COM--25**, 48--60.

HELD, M., P. WOLFE and H. P. CROWDER. 1974. Validation of Subgradient Optimization. *Math. Prog.* **6**, 62--88.

JOHNSON, D. S., J. K. LENSTRA and A. H. G. RINNOOY KAN. 1978. The Complexity of the Network Design Problem. *Networks* **8**, 279--285.

KLEINROCK L. 1976. *Queueing Systems, Volume 2: Computer Applications* John Wiley, New York

LAWLER, E. L. 1976. *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, New York.

LEBLANC, L. J., and R. REDDOCH. 1990. Reliable Link Topology/Capacity Design and Routing in Backbone Telecommunication Networks. Working Paper No. 90-08, Owen Graduate School of Management, Vanderbilt University, Nashville, Tennessee, October.

MAGNANTI, T. L., and R. T. WONG. 1981. Accelerating Benders Decomposition: Algorithmic Enhancements and Model Selection Criteria. *Oper. Res.* **29**, 464--484.

MARUYAMA, K., K. FRATTA and D.T. TANG. 1977. Heuristic Design Algorithm for Computer Communication Networks with Different Classes of Packets. *IBM J. Res. Develop.*, **21**, 360--369.

MARUYAMA, K., and D.T. TANG. 1976. Discrete Link Capacity Assignment in Communication Networks. *Proc. Third Int. Comput. Comm. Conf.*, 92--97.

MONMA, C. L., and D. D. SHENG. 1986. Backbone Network Design and Performance Analysis: A Methodology for Packet Switching Networks. *IEEE Journal on Selected Areas in Communications*, **SAC-4**, 946-965.

PIRKUL, H., and S. NARASIMHAN. 1987. A New Algorithm for the Design of Backbone Networks. Working Paper, College of Business, The Ohio State University, Columbus, Ohio.

PIRKUL, H., and S. NARASIMHAN. 1988. Primary and Secondary Route Selection in Backbone Computer Networks. Working Paper, College of Business, The Ohio State University, Columbus, Ohio.

TYMES, L. R. W. 1981. Routing and Flow Control in TYMNET. *IEEE Trans. Communications* **COM--29**, 392--398.

YUM, T. 1981. The Design and Analysis of a Semidynamic Deterministic Routing Rule. *IEEE Trans. Communications* **COM--29**, 498--504.

ZANGWILL, W. I. 1968. Minimum Concave Cost Flows in Certain Networks *Man. Sci.* **14**, 429--450

Table I. The effect of Cost Randomness Factor and FC/VC Ratio on Algorithmic Performance

FC/VC ratio r	Hop par. h_0	Cost Randomness Factor w														
		0.0					0.5					1.0				
		% FC	# of edges	ave. Hop	% Gap	CPU time†	% FC	# of edges	ave. Hop	% Gap	CPU time†	% FC	# of edges	ave. Hop	% Gap	CPU time†
10	3	16	34	2.26	3.16	116	17	39	2.10	0.69	129	19	37	2.28	1.25	97
	4	12	28	2.54	1.41	129	15	35	2.30	0.55	255	13	28	2.67	0.03	139
	5	10	24	2.74	0.05	180	15	35	2.64	0.57	298	11	25	2.89	0.04	178
20	3	24	31	2.29	3.50	140	24	33	2.22	3.55	147	33	37	2.27	4.28	94
	4	19	25	2.69	4.80	154	24	32	2.40	2.89	170	21	25	2.78	0.65	127
	5	16	22	2.93	0.00	126	20	28	2.59	2.84	320	19	23	2.97	0.95	161
50	3	50	33	2.28	20.18	129	45	32	2.28	19.22	166	56	36	2.29	23.20	100
	4	38	24	2.72	9.19	144	37	26	2.65	11.79	224	43	25	2.80	9.96	135
	5	34	21	3.12	4.13	168	33	23	3.00	8.93	250	36	21	3.23	13.39	162

* Network size: 20 nodes, 180 edges, 180 commodities
† CPU time in seconds on an IBM 3083

Table II. Effect of Problem Size on Algorithmic Performance

Network size		Uniform Hop Parameter h_0														
		3					4					5				
Nodes	Edges*	% FC	# of edges	ave. Hop	% Gap	CPU time†	% FC	# of edges	ave. Hop	% Gap	CPU time†	% FC	# of edges	ave. Hop	% Gap	CPU time†
10	20	47	11	2.15	5.89	3	41	9	2.49	1.94	4	41	9	2.49	4.88	5
10	40	37	12	2.03	9.90	6	35	12	2.14	8.00	9	29	10	2.47	10.70	13
20	90	37	29	2.23	11.47	53	31	25	2.52	5.83	84	31	25	2.50	5.69	101
20	180	24	33	2.2	3.55	147	24	32	2.40	2.89	170	20	28	2.59	2.84	320
25	150	36	44	2.28	11.39	120	29	36	2.55	5.07	365	27	34	2.64	5.28	337
25	300	25	51	2.26	4.23	380	21	43	2.55	1.89	784	19	41	2.68	1.92	1040
30	210	33	58	2.37	8.31	214	26	45	2.78	4.19	386	23	42	2.91	2.70	622
30	420	23	65	2.23	4.53	1916	19	57	2.48	1.65	1399	18	55	2.54	1.43	2195

Parameter Settings: Cost Randomness Factor **w**=0.50, FC/VC Ratio **r**= 20

† CPU time in seconds on an IBM 3083

* Number of Commodities = Number of Edges

Table III. The effect of hop restriction on algorithmic performance

Prob No.	Hop Restriction $[h_1,h_2]$	Design Characteristics			Average No. of Hops	Connectivity			% GAP
		Cost Index †	% FC	No. of edges		Average	Max	Min	
1	[1,1]	412.1	79	300	1.00	24.00	24	24	0.00
2	[2,2]	128.7	39	78	1.74	5.30	9	4	15.51
3	[3,3]	104.9	25	51	2.26	3.50	6	2	4.23
4	[4,4]	100.6	21	43	2.55	2.68	5	1	1.89
5	[5,5]	100.2	19	41	2.68	2.55	5	1	1.92
6	[6,6]	100.0	19	41	2.68	2.55	5	1	1.53
7	[7,7]	100.0	19	41	2.68	2.55	5	1	1.53
8	[1,2]	150.1	45	89	1.62	6.24	10	4	6.38
9	[1,3]	144.9	44	83	1.71	5.78	9	4	4.53
10	[1,4]	141.0	41	75	1.83	5.22	8	4	3.23
11	[1,5]	140.9	40	74	1.86	5.03	8	4	3.27
12	[1,6]	138.0	40	73	1.94	5.02	9	3	2.67
13	[1,7]	137.9	40	71	1.96	4.93	7	3	1.70
14	[2,3]	110.0	30	54	2.15	3.51	6	2	5.41
15	[2,4]	109.0	25	51	2.20	3.37	6	2	5.52
16	[2,5]	109.6	28	55	2.14	3.67	7	2	6.02
17	[2,6]	109.5	27	54	2.18	3.57	6	2	4.92
18	[2,7]	108.8	26	51	2.28	3.40	6	2	6.31
19	[3,4]	101.5	21	44	2.52	2.80	5	1	2.20
20	[3,4]	101.5	22	44	2.51	2.75	5	1	2.08
21	[3,6]	101.3	21	43	2.56	2.68	5	1	3.17
22	[3,7]	101.3	21	43	2.56	2.68	5	1	3.58

† Normalized Cost = Total Cost for design as a percentage of lowest design cost

Network Size: 25 nodes, 300 edges, 300 commodities
Parameter Settings: Cost Randomness Factor w=0.50; FC/VC Ratio r=20

FIGURE 1: Piecewise Linear, Concave Cost Function

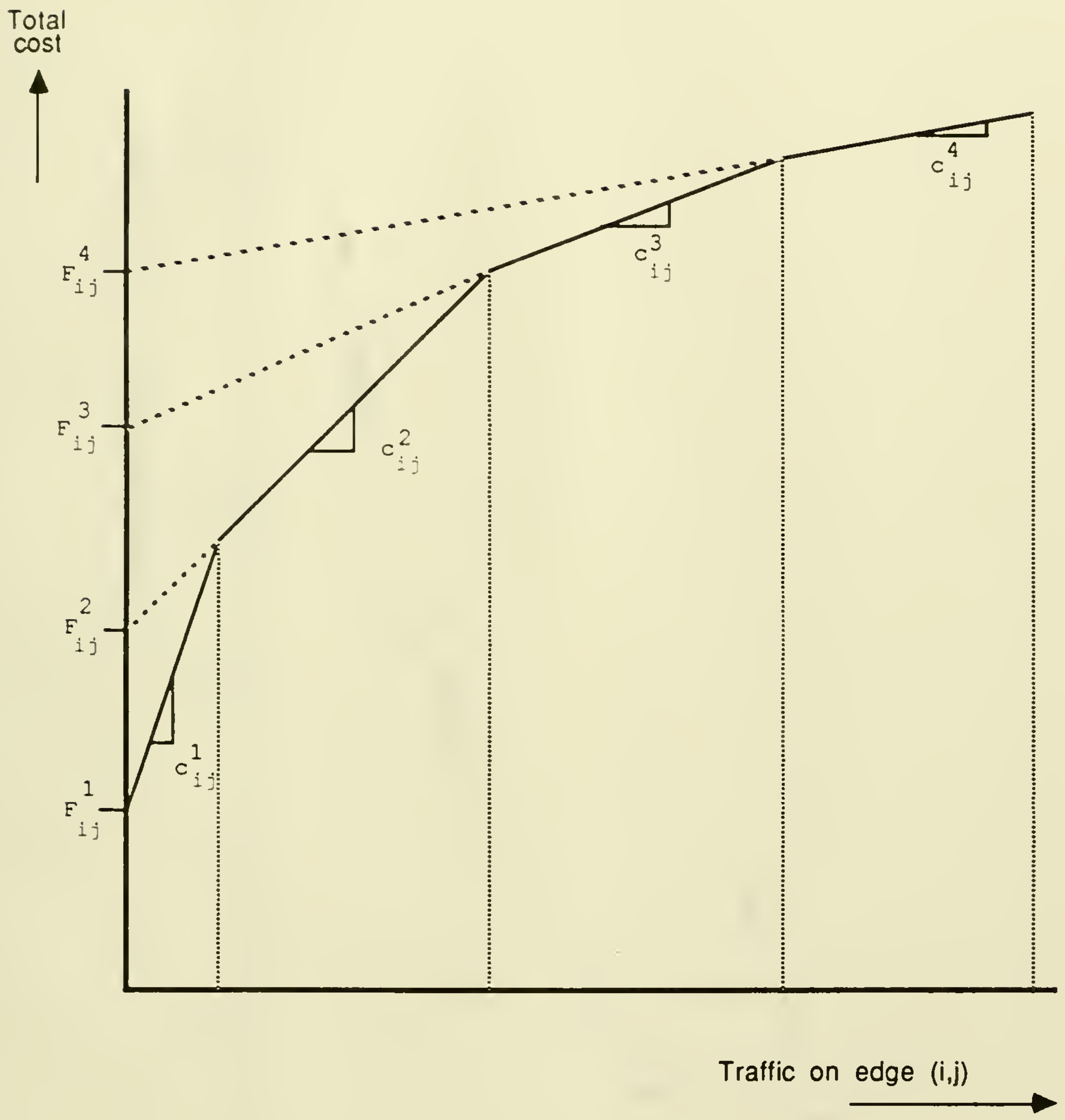
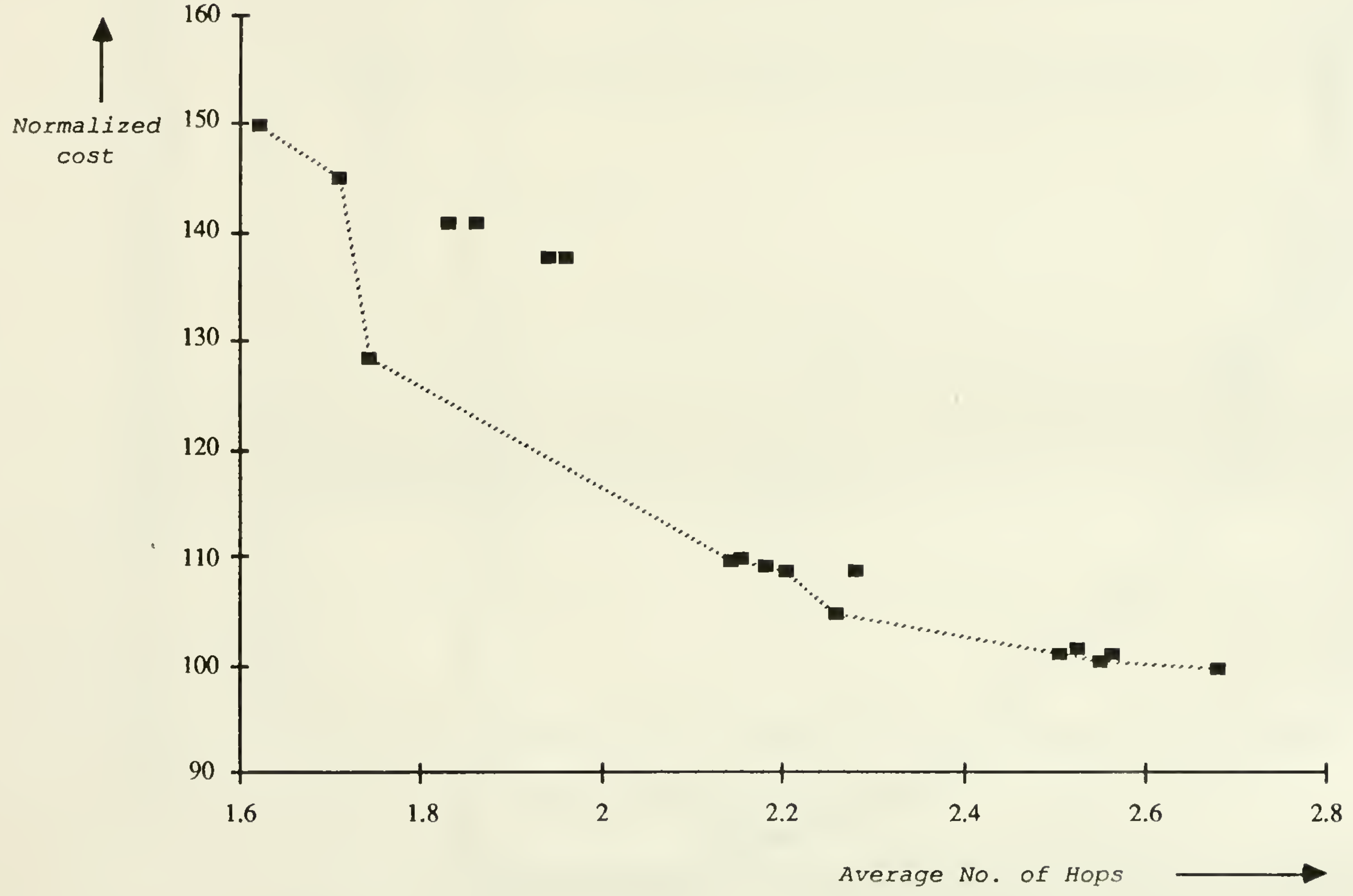


FIGURE 2: COST versus AVERAGE HOPS


Normalized cost
Average No. of Hops