OPERATIONS RESEARCH CENTER

Working Paper

Network Flow Models for Designing Diameter-Constrained Minimum Spanning and Steiner Trees

by

Luis Gouveia
Thomas L. Magnanti

OR 359-01 **August 2001**

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Network Flow Models for Designing Diameter-Constrained Minimum Spanning and Steiner Trees

Luis Gouveia [1] and Thomas L. Magnanti [2]

[1] DEIO - CIO
Faculdade de Ciências da Universidade de Lisboa
Bloco C/2 - Campo Grande
CIDADE UNIVERSITARIA
1700 Lisboa
Portugal
email address: lgouveia@fc.ul.pt

[2] Department of Electrical Engineering and Computer Science and
Sloan School of Management
MIT
Cambridge, MA USA, 02139
email address: magnanti@mit.edu

August 2001

Abstract

The Diameter-Constrained Minimum Spanning Tree Problem seeks a least cost spanning tree subject to a (diameter) bound imposed on the number of edges in the tree between any node pair. A traditional multicommodity flow model with a commodity for every pair of nodes was unable to solve a 20-node and 100-edge problem after one week of computation. We formulate the problem as a directed tree from a selected central node or a selected central edge. Our model simultaneously finds a central node or a central edge and uses it as the source for the commodities in a directed multicommodity flow model with hop constraints. The new model has been able to solve the 20-node, 100-edge instance to optimality after less than four seconds. We also present model enhancements when the diameter bound is odd (these situations are more difficult). We show that the linear programming relaxation of the best formulations discussed in this paper always give an optimal integer solution for two special, polynomially-solvable cases of the problem. We also examine the Diameter Constrained Minimum Steiner Tree problem. We present computational experience in solving problem instances with up to 100 nodes and 1000 edges. The largest model contains more than 250,000 integer variables and more than 125,000 constraints.

Keywords: Spanning Trees, Steiner Trees, Diameter Constraints, Multicommodity Flow Models, Hop-Indexed Models

1. Introduction

The minimal spanning tree problem (MST) and Steiner tree problem (ST), along with the traveling salesman (TSP), are the most celebrated problems in the field of combinatorial optimization. In the classical minimal spanning tree problem, we are given a prescribed graph G = (V,E) with node set V and edge set E as well as a cost c_e associated with each edge e of E. We wish to find a spanning tree T of the graph with minimum total cost, as measured by the sum of the costs of the edges in the spanning tree. In the Steiner tree problem, the tree needs to span only a subset of the nodes in the underlying graph G. The other nodes, called Steiner nodes, are optional. The MST and ST problems arise directly in many applications (for example, the design of telecommunication systems) and as a subproblem in many other applications, including the TSP.

In this paper, we consider a computationally challenging class of the MST and ST problems, those with bounds imposed on the number of edges in the tree between node pairs. The most general version of this problem imposes a bound B_{pq} on the number of edges in the tree between every pair of nodes p and q. Note that we can assume that $B_{pq} = B_{qp}$ since in any feasible constrained tree, the path between nodes p and q can contain no more than min $\{B_{pq}, B_{qp}\}$ edges and so we can replace B_{pq} and B_{qp} by min $\{B_{pq}, B_{qp}\}$.

One version of this problem, known as the Fixed Root Diameter Minimal Spanning Tree Problem or Rooted Diameter Minimal Spanning Tree Problem (RDMST), imposes a constraint that the tree path from a specified root node 1 to every other node j contains no more than H edges (H is a given positive integer). That is, $B_{1q} = H$ and $B_{pq} = +\infty$ if $p \neq 1$ and $q \neq 1$ (or, $B_{pq} \geq \min\{|V|-1,2H\}$). This problem models the design of centralized telecommunication networks with quality of service constraints. The root node represents the site of a central processor (computer) and the remaining nodes represent terminals that are required to be linked to the central processor. The path constraints limit the number of edges between the root node and any other node and guarantee a specified level of service with respect to certain performance measures, for example, guarantee a prescribed level of reliability to potential link or node failures (see, for example, Woolston and Albin (1988)). This special case has received considerable attention in the literature. Gouveia (1998) has discussed applications, linear integer programming formulations, lower bounding methods, and relevant references for the RDMST.

In this paper, we focus on a version of the problem with uniform constraints between every node pair, which we refer to as the Diameter-constrained Minimal Spanning Tree Problem (DMST). In this case, we set $B_{pq} = D$ for all nodes p and q. This case imposes a bound on the diameter of the tree, which is the maximum number of edges in any of its paths. The DMST models situations when all of the nodes can communicate with each other, and we wish to guarantee a certain level of service between any node pair. When $D \geq 4$, the DMST problem is NP-Hard (Garey and Johnson (1979)). When D = 2 or 3, the problem is easy to solve (see Appendix 2). In a previous study of this problem, Achuthan, Caccetta, Caccetta and

Geelen (1992,1994) have proposed and tested several exact approaches. The first of these papers describes several branch-and-bound algorithms that use the unconstrained minimum spanning tree solution as the bounding component and differing branching rules. The authors solve complete random cost instances containing up to 30 nodes. The second paper proposes a branching rule based upon the interesting fact that the nodes of a feasible solution can be partitioned into layers around a central node (D even) or a central edge (D odd). The branching assigns nodes to different layers and relaxes the problem into a kind of minimum arborescence problem defined on a layered graph. With this approach, the authors have improved upon their previously results and been able to solve instances with up to 50 nodes and D = 4. Achuthan, Caccetta, Caccetta and Geelen (1992,1994) have also proposed two other alternative models, which we briefly summarize in Section 3.1. More recently, Abdalla, Deo and Fraceschini (1999) have presented some heuristics for the DMST and examined parallel implementations of these heuristics.

In this paper we examine network flow-based formulations for the DMST and for the Steiner tree version of the problem. For ease of exposition, we cast most of our development for the minimal spanning tree problem, and then later in Section 5 describe modest alterations needed to accommodate the more general Steiner tree problem. After formulating a multicommodity flow-based integer programming model for the general case, we introduce several alternative models for the DMST that lead to more effective solution procedures. We then report on computational experience indicating that the modelling improvements can induce very substantial reductions in solution time. For example, after over one week of computations, the original formulation was unable to solve one problem, whereas computations with the alternative formulations were able to solve the problem in less than one second.

Our modelling improvements use four essential ideas.

First, following Magnanti and Wong (1984), we use a multicommodity flow formulation that has proven to be valuable in modelling many network design problems.

Second, we model the problem as a multicommodity flow problem with a single source, instead of the traditional model with a commodity for every pair of nodes. We simultaneously find a "central node" or a "central edge" that serves as the source for the commodities. This modelling approach permits us to reduce the number of commodities by a factor of n, the number of nodes of the graph.

Third, we direct the problem treating the solution as directed tree from a selected central node or a selected central edge. The idea of directing network design problems has proven to be a powerful modelling construct in the past to improve formulations. In several contexts, Wong (1984), Balakrishan, Magnanti and Michandani (1994), Goemans (1994), Geomans and Myung (1993), Chopra and Rao (1994), and Magnanti and Raghavan (1999) have shown how to use this technique to improve models of the hierarchical network design problem and various network connectivity problems including the Steiner tree problem. In particular, the directing technique is useful for modelling the minimal spanning tree problem,

since the directed multicommodity flow model or an equivalent enhanced undirected model, but not a traditional undirected model, gives an extended description of the convex hull of incidence vectors of spanning trees (see Martin (1986) and Magnanti and Wolsey (1996)).

Fourth, following Gouveia (1998), we use a hop-indexed formulation to improve the multicommodity flow formulation. His computational results suggest that in the context of network design problems involving hop constraints, the linear programming relaxation of an appropriate hop-indexed reformulation can improve substantially on the linear programming relaxation of a multicommodity flow formulation.

Our results will demonstrate the power of using the single-sourcing approach combined with directing the model and using a hop-index formulation for obtaining significant improvements in solution times.

The remainder of this paper is organized as follows. In Section 2, we introduce a basic multicommodity flow formulation of the diameter-constrained minimal spanning tree problem. In Section 3, we show how to model the DMST as a single source directed problem, distinguishing situations in which the diameter restriction D is even and odd. We discuss both traditional network flow and hop-indexed flow models. Our computational results show that these models are not as successful for solving the DMST instances when D is odd as when D is even. Thus, we also present some valid inequalities for the situation when D is odd. In Section 4, we present a different formulation when D is odd based upon solving the problem in an expanded graph containing pseudo-nodes corresponding to potential central edges. Appendix 1 shows that the linear programming relaxation of these enhanced models dominates the linear programming relaxation of the original models presented in Section 3. Section 5 briefly discusses minor modifications to the spanning tree formulations for the Steiner case. In Section 6, we report on computation experience on graphs with up to 80 nodes and 800 edges. These problems instances contain up to 250,000 integer variables. In Appendix 2, we present tight formulations for the polynomially-solvable cases of the DMST when D = 2 and D = 3.

2. Basic Formulations

To formulate a multicommodity flow model MCF of the diameter-constrained problem, we use two sets of variables. Variables x_e ($e \in E$) indicate whether the minimum spanning tree contains edge e, and directed flow variables y_{ij}^{pq} ($\{i,j\} \in E$; $p,q \in V$; $p \neq j$ and $q \neq i$) specify whether the unique path from node p to node q traverses edge $\{i,j\}$ in the direction i to j.

Table 0. Modeling a Diameter Constrained Spanning Tree

Model MCF
$minimize \quad \sum_{e \in E} c_e x_e$
$subject\ to \sum_{e \in E} x_e = n$
$\sum_{j \in V} y_{ij}^{pq} - \sum_{j \in V} y_{ji}^{pq} = \begin{cases} 1 & i = p \\ 0 & i \neq p,q \\ -1 & i = q \end{cases} \quad for\ all\ i,p,q \in V$
$y_{ij}^{pq} + y_{ji}^{pq} \leq x_e \quad for\ all\ e = \{i,j\} \in E, p,q \in V$
$\sum_{\{i,j\} \in E} (y_{ij}^{pq} + y_{ji}^{pq}) \leq D \quad for\ all\ p,q \in V$
$y_{ij}^{pq} \in \{0,1\} \quad for\ all\ e = \{i,j\} \in E, p,q \in V$
$x_e \in \{0,1\} \quad for\ all\ e \in E.$

This formulation is a multi-source multi-destination formulation for the minimum spanning tree problem, as given in Magnanti and Wong (1974), augmented with the set of cardinality constraints $\sum_{\{i,j\} \in E} (y_{ij}^{pq} + y_{ji}^{pq}) \leq D$ for all p,q ∈ V. These inequalities state that the path between any two nodes p to q contains no more than D edges. The formulation MCF is quite similar to a formulation presented in Balakrishnan and Altinkemer (1992) for a more general network design problem with hop constraints.

As stated, the MCF formulation has a commodity, and so flow variables y_{ij}^{pq}, between every pair p and q of nodes. Since $y_{ij}^{pq} = y_{ji}^{qp}$ (for all i,j,p,q ∈ V) in any integer solution to the problem, we can eliminate half the variables and commodities, creating a reduced problem, using only the variables with p < q. If the value of the variables (x_e, y_{ij}^{pq} with p < q) are feasible for the linear programming relaxation of the reduced problem, then the value of the variables (x_e, y_{ij}^{pq} with p < q, $y_{ij}^{pq} = y_{ji}^{qp}$ with p > q) are feasible for the linear programming relaxation of the full commodity problem. This observation permits us to conclude that the value of the optimal solution of the linear programming relaxation of the reduced model is equal to the value of the optimal solution of the linear programming relaxation of the full commodity model.

Unfortunately, the reduced MCF model still contains a large number of variables and constraints and can be difficult to solve. On a Pentium Pro 450Mhz computer, the CPLEX solver (version 7.0) was unable to solve a modest-sized 20 node and 100 edge instance of the MCF formulation after one week of computation. This model contains about 40,000 integer variables and 20,000 constraints.

Following Balakrishnan, Magnanti and Wong (1989), we could tighten the linear programming relaxation of the MCF formulation using generalizations of the constraints $y_{ij}^{pq} + y_{ji}^{pq} \leq x_e$ that include flow variables for pairs of commodities on the lefthand side of the inequalities. However, the model with

the new constraints would contain a much larger number of constraints and it seems unlikely that using it would permit us to solve this problem instance. A different idea is to tighten the linear programming relaxation of the constrained path problem that is embedded in the MCF formulation. We describe one such idea (a hop-indexed model) in the next section in the context of a different model for the DMST problem. The possible improvements obtained in this way for the MCF model will not, however, be readily realizable because of the large number of commodities in the formulation. Thus, we will not explore the hop-indexed approach in the context of the MCF model.

Throughout the remainder of our discussion, for any optimization model P, we let P_L denote its linear programming relaxation, F(P) denote its set of feasible solutions, and v(P) denotes its optimal objective value. To obtain the linear programming relaxation MCF_L of any model like MCF that contains binary variables, we replace constraints of the form $x_e \in \{0,1\}$ and $y_{ij}^{pq} \in \{0,1\}$ by the lower and upper bound constraints $0 \le x_e \le 1$ and $0 \le y_{ij}^{pq} \le 1$.

As noted, the MCF model is based on a multi-source multi-destination network flow model for the minimum spanning tree presented in Magnanti and Wong (1974). As is well known, we can obtain a single-source multi-destination model for the minimum spanning tree whose linear programming relaxation is as tight as the linear programming relaxation of the multi-source multi-destination model (see, for instance, Magnanti and Wolsey (1996) and the next section of this paper). This single-source model involves only n–1 commodities, defined from a specific node to all other nodes in the graph. The "distance" constraints between every pair of nodes involved in the DMST problem suggest that we might not be able to use a similar idea for modeling the DMST problem. However, as shown in the next section, special properties of trees will permit us to also model the DMST problem as a single-source network flow problem.

3. Single Sourcing and Directing the Problem

In this section, we use the following elementary properties of trees to formulate single source and directed versions of the DMST problem. This formulation permits us to reduce the number of commodities by a factor of n and to obtain much better algorithmic performance.

i) A tree T has diameter no more than an even integer D if and only if some node p of T satisfies the property that the path from node p to any other node of the tree contains at most D/2 edges.

ii) A tree T has diameter no more than an odd integer D if and only if some edge {p,q} of T satisfies the property that the path to any other node of the tree from either node p or node q contains at most (D–1)/2 edges.

As observed by to Handler (1973), the "central" node p and "central" edge {p,q} in these properties are easy to determine. If a tree has diameter D, then starting from any node i, we find any node j that is farthest away from it, in the sense of number of edges in the unique path connecting these nodes. Then from node j, we find the node k farthest from it. The midpoint of the tree path joining nodes j and k is either a node p or the midpoint of edge {p,q}. This result shows an intimate relationship between the diameter constrained spanning tree problem and a center location problem. Other researchers (Camerini, Galbiati and Maffioli (1980)) have used this observation when studying a related minimax diameter spanning tree problem.

These observations show that when D is even we could solve the diameter constrained minimum spanning tree problem by solving |V| fixed root problems and that when D is odd we could solve the problem by solving |E| fixed root problems after shrinking each edge to create a pseudo node. We will instead create single models that simultaneously select the central node and central edge and the diameter-constrained tree.

To easy our notation in describing these models, we will augment the original network by adding an additional node 0 and zero cost edges {0,j} for all node $j \in V$ to create an augmented graph. We will refer to this network as the *augmented network* and the network without the node 0 and its incident edges as the *original network*. We let $V_0 = V \cup \{0\}$ and $E_0 = E \cup \{\{0,j\}$ for all $j \in V\}$ denote the node and edge sets in the n+1 node augmented network. Then in this graph:

i) when D is even, we wish to select a minimum cost spanning tree containing a single edge {0,j} and ensure that the unique path from node 0 to every node contains at most (D/2) + 1 edges, and

ii) when D is odd we wish to select a minimum cost set of n+1 edges so that: (i) n edges form a spanning tree containing exactly two edges {0,i} and {0,j}, (ii) the unique path from node 0 to every node contains at most [(D-1)/2] + 1 edges, and (iii) the $(n+1)^{th}$ edge e = {i,j} from the original network connects the endpoints i and j of the two edges {0,i} and {0,j} .

Figures 1 and 2 give examples of the diameter constrained spanning tree problems on the augmented networks when D = 4 and D = 5.

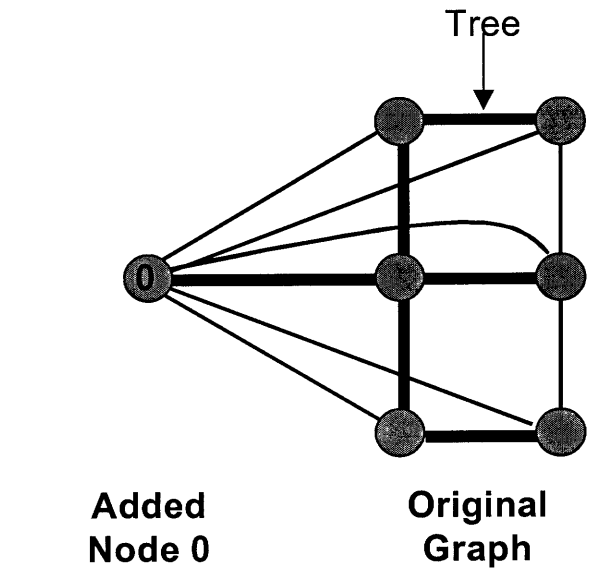

Figure 1 - Augmented Network for Situations with D Even (D = 4)

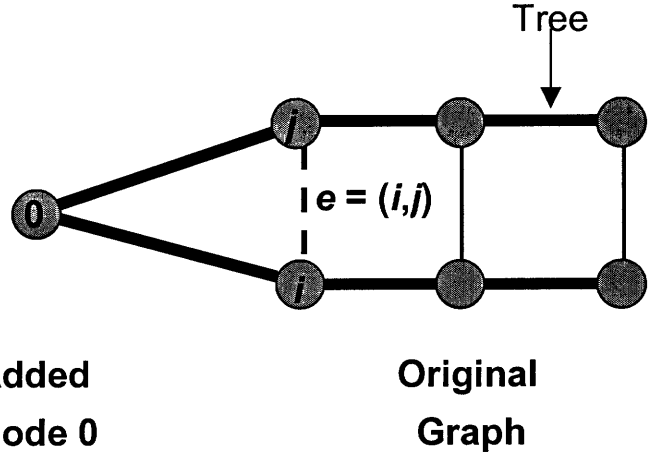

Figure 2- Augmented Network for Situations with D Odd (D = 5).

In choosing the underlying tree in Figure 1 or 2, we could choose n undirected edges or we could replace each edge {i,j} in the network with two directed arcs (i,j) and (j,i), each with the same cost c_{ij} as the edge {i,j}. For the directed version of the models, we let A_0 denote the (directed) arc set in the augmented graph and A denote the (directed) arc set in the original network. In the following sections, we will investigate both the undirected and directed models.

Achuthan and Caccetta (1992,1993) have proposed formulations for the DMST based on a similar augmented network. Their models are based on a set of modified Miller-Tucker-Zemlin subtour elimination constraints. Although their models are quite compact, results given in Gouveia (1995) using a similar model for the fixed-root version of the problem have shown that the modified Miller-Tucker-

Zemlin subtour elimination constraints represent a rather weak modeling approach, in terms of the associated linear programming relaxations, for imposing a limit on the maximum number of arcs in a path. Some results given in Section 6 using a simplified version of their model, as suggested in subsequent work by Achuthan, Caccetta, Caccetta and Geelen (1994), exhibit the same computational behavior for the DMST.

3.1 Underlying Core Models

The models presented in this section for the diameter constrained spanning tree problem contain two core subproblems: a rooted minimum spanning tree problem and hop-constrained path problem. We can model these core models in different ways and by mixing and matching the possible core components, we can create several alternate models for the diameter constrained spanning tree problem.

Table I shows core models for a rooted minimum spanning tree problem defined on an undirected graph (V_0,E_0) with n + 1 nodes and a rooted directed minimum spanning tree problem (the minimum arborescence problem) defined on a directed graph (V_0,A_0) with n + 1 nodes. Although the directed graph (V_0,A_0) could have any general structure, for our purposes, we obtain it by directing a given undirected graph (V,A) as indicated in the last section. The undirected model uses two sets of variables. Variables x_e (e $\in$ E) indicate whether the minimum spanning tree contains edge e, and directed flow variables y_{ij}^k ({i,j} $\in$ E; k $\in$ V; i $\neq$ k) specify whether the unique path from the root node to node k traverses edge {i,j} in the direction i to j. The directed model uses two sets of variables as well: the directed flow variables as in the undirected model and variables x_{ij} ((i,j) $\in$ A) indicating whether the spanning tree contains the directed arc (i,j).

Table I. Modeling a Rooted Spanning Tree

Undirected Minimum Spanning Tree Model	Minimum Arborescence Model
$\text{minimize} \quad \sum_{e\in E_0} c_e x_e$	$\text{minimize} \quad \sum_{(i,j)\in A_0} c_{ij} x_{ij}$
$\text{subject to} \quad \sum_{e\in E_0} x_e = n$	$\text{subject to} \quad \sum_{(i,j)\in A_0} x_{ij} = n$
$\sum_{j\in V} y_{ij}^k - \sum_{j\in V_0} y_{ji}^k = \begin{cases} 1 & i = 0 \\ 0 & i \neq 0, k \\ -1 & i = k \end{cases} \quad \text{for all } k \in V, i \in V_0$	$\sum_{j\in V} y_{ij}^k - \sum_{j\in V_0} y_{ji}^k = \begin{cases} 1 & i = 0 \\ 0 & i \neq 0, k \\ -1 & i = k \end{cases} \quad \text{for all } k \in V, i \in V_0$
$y_{ij}^k + y_{ji}^k \leq x_e \quad \text{for all } e = \{i,j\} \in E_0, k \in V$	$y_{ij}^k \leq x_{ij} \quad \text{for all } (i,j) \in A_0, k \in V$
$y_{ij}^k \geq 0 \quad \text{for all } e = \{i,j\} \in E_0, k \in V$	$y_{ij}^k \geq 0 \quad \text{for all } (i,j) \in A_0, k \in V$
$x_e \in \{0,1\} \text{ for all } e \in E_0.$	$x_{ij} \in \{0,1\} \text{ for all } (i,j) \in A_0.$

The cardinality constraints in these models guarantee that the tree solution contains n edges (arcs). The inequalities $y_{ij}^k + y_{ji}^k \leq x_e$ and $y_{ij}^k \leq x_{ij}$ prohibit flowing on any edge e or arc (i,j) that is not included in the

minimum spanning tree. Together with the flow conservation constraints, they guarantee that the solution is connected. Note that the constraints in the directed model imply that $1 \le \sum_i y_{ik}^k \le \sum_i x_{ik}$ for all values of $k \in$ V, which together with the constraint $\sum_{(i,j)\in A} x_{ij} = n$ imply that $\sum_i y_{ik}^k = \sum_i x_{ik} = 1$ for all $k \in$ V.

Let *ST* denote the feasible set of the undirected minimum spanning tree model and *DST* the feasible set of the directed minimum spanning tree (directed arborescence) model.

As is well known (see, for example, Magnanti and Wolsey (1996)), the linear programming relaxation of the directed model always has an optimal solution with integer values for the variables *x*. The undirected model might not have an integer optimal solution. As noted by Magnanti and Wolsey (1996), it is not difficult to formulate an undirected model whose linear programming bound is equal to the linear programming bound given by the directed model. However, the enhanced undirected model contains far more constraints than the directed model and, for our purposes, is impractical from a computational perspective.

To model the hop constraints we can use either of the two models shown in Table II. In both cases, we assume we are modeling a problem of finding a directed path containing at most *H* arcs from a given root node (node 0) to a specific node *k*. The constrained path model uses only flow variables defined on the arcs of the underlying network, sending one unit of flow from the root node to node k, but using at most *H* arcs. The hop-constrained model is an extended formulation. Besides the flow variables, it also uses binary variables z_{ij}^{hk} indicating whether arc (i,j) is the h[th] arc in the path joining the root node and node *k*.

Table II. Modeling a Hop-Constrained Path (from node 0 to node k)

Constrained Path Model	Hop-Constrained Path Model
$\sum_{j\in V_0} y_{ij}^k - \sum_{j\in V} y_{ji}^k = \begin{cases} 1 & i = 0 \\ 0 & i \ne 0,k \\ -1 & i = k \end{cases}$ $\sum_{(i,j)\in A_0} y_{ij}^k \le H$ $y_{ij}^k \in \{0,1\}$ *for all* $(i,j) \in A_0$.	$\sum_{j\in V} z_{0j}^{1k} = 1$ $\sum_{j\in V} z_{ij}^{2k} = z_{0i}^{1k}$ *for all* $i \in V$ $\sum_{j\in V} z_{ij}^{h+1,k} - \sum_{j\in V} z_{ji}^{hk} = 0$ *for all* $i \in V, h = 2,\ldots,H-1$ $\sum_{j\in V} z_{jk}^{Hk} = 1$ $y_{ij}^k = \sum_{h=1,\mathrm{K}\,H} z_{ij}^{hk}$ *for all* $(i,j) \in A_0$ $z_{ij}^{kh} \in \{0,1\}$ *for all* $(i,j) \in A_0, h = 1,\ldots,H$ $z_{kk}^{kh} \in \{0,1\}$ *for all* $k \in V$, $h = 2,\ldots,H$.

Since the 0-1 variables y_{ij}^k specify a path between the root and node k, the inequality $\sum_{(i,j)\in A_0} y_{ij}^k \le H$ in the constrained path model restricts the chosen path to contain at most H arcs. The Hop-Constrained Path model contains constraints stating that an arc enters node i in position h if and only

if another arc emanates from this node in position h + 1 and that one arc enters node k in position H. Note that this model contains "loop" variables z_{kk}^{hk} (h = 2,..., H) with zero cost to model situations when the path from the root node to node k contains fewer than H arcs (that is, $z_{ik}^{hk} = 1$ for some node i and h $\leq$ H). The equalities $y_{ij}^{k} = \sum_{h=1,K\,H} z_{ij}^{hk}$ relate the original variables with the extended variables. Although the original flow variables y_{ij}^{k} are unnecessary for obtaining a valid formulation for the core subproblem, we will need them for relating the hop-indexed variables with the flow variables included in the tree models.

Let DP_H^k denote set of feasible solutions of the constrained path model and DHP_H^k the set of feasible solutions of the hop-constrained model.

The hop-constrained problem contains far more variables than the constrained path problem. However, it is a network flow (path) problem on an expanded network and so has the advantage that the extreme points of its linear programming relaxation are integer-valued (see Gouveia (1998)) whereas, in general, the linear programming relaxation of the constrained path model has fractional extreme points. This result implies that for the linear programming relaxations, in the space of y variables, the feasible set of the hop-constrained path model is contained in the feasible set of the constrained path model.

3.2 Network Flow Models for the Diameter Constrained MST Problem

3.2.1 The Situation When D is Even

As shown in Table III, by choosing either direct or undirected versions of the spanning tree problem and the constrained path or hop variable versions of the hop-constrained path problem, when D is even, we can create four different models of the diameter constrained minimum spanning tree problem.

Table III. Models of the DMST when D is Even

	Undirected	Directed
Path	**Model UMCF** $minimize \sum_{e \in E_0} c_e x_e$ $subject\ to\ (x,y) \in ST$ $\sum_{j \in V} x_{\{0,j\}} = 1$ $y \in DP^k_{(D/2)+1}\ for\ all\ k \in V.$	**Model DMCF** $minimize \sum_{(i,j) \in A_0} c_{ij} x_{ij}$ $subject\ to\ (x,y) \in DST$ $\sum_{j \in V} x_{0j} = 1$ $y \in DP^k_{(D/2)+1}\ for\ all\ k \in V.$
Hop	**Model HopUMCF** $minimize \sum_{e \in E_0} c_e x_e$ $subject\ to\ (x,y) \in ST$ $\sum_{j \in V} x_{\{0,j\}} = 1$ $(z,y) \in DHP^k_{(D/2)+1}\ for\ all\ k \in V.$	**Model HopDMCF** $minimize \sum_{(i,j) \in A_0} c_{ij} x_{ij}$ $subject\ to\ (x,y) \in DST$ $\sum_{j \in V} x_{0j} = 1$ $(z,y) \in DHP^k_{(D/2)+1}\ for\ all\ k \in V.$

The degree constraint imposed on the root, the flow conservation for the root node, and the constraints linking the x and y variables guarantee that the linear programming relaxation of these models satisfy the equalities $y^k_{0j} = x_{0j}$ ($y^k_{0j} = x_{\{0,j\}}$) for all j,k $\neq$ 0 and thus, when we impose integrality, the node k $\in$ V with $x_{0k} = 1$ serves as a transshipment node for all the commodities. In accordance with the interpretation given at the beginning of this section, this transshipment node will be the central node of the tree in the original graph.

Our computational results show that the linear programming bound given by the undirected multi-source/multi-destination MCF model is generally much better than the linear programming bound given by the undirected UMCF model. However, in some cases the UMCF model can provide a tighter linear programming relaxation.

Example. In a complete graph on five nodes numbered 1 to 5, suppose D = 2 and edges {i,j} with j = i+1 (mod 5) have cost 1. Each remaining edge has a cost 10. The MCF model has an optimal linear programming value of 7 while the UMCF model has an optimal linear programming value of 8.5. The optimal value to the problem is 22. ♦

Using the directed DMCF formulation, the CPLEX solver was able to solve the previously mentioned DMST instance with 20 nodes and 100 edges within 200 seconds. For all instances that we tested, the linear programming bound given by the directed DMCF model is as good (actually better) than the linear

programming model given by the traditional model MCF. We have been unable to verify whether this relationship holds for all cases. For the previous five-node example, the DMCF model has a linear programming value of 13.0.

As in other related models (see Magnanti and Wolsey (1996)), the linear programming relaxation of the directed model DMCF is at least as strong as (and generally is much stronger than) the linear programming relaxation of the undirected model UMCF. Similarly, the directed hop-indexed model has a stronger linear programming relaxation than the undirected model.

We note that when creating these UMCF and DMCF models, we can remove one set of flow balancing constraints for each node $k \neq 0$ (either from the tree model or from the constrained path model) since they become redundant. A similar observation applies to the hop-indexed models HopUMCF and HopDMCF. Using the equalities $y_{ij}^k = \sum_{h=1,K,(D/2)+1} z_{ij}^{hk}$, it is easy to show that the flow conservation constraints of the tree model are aggregations of the hop-indexed flow conservation constraints of the Hop-Constrained Path model. Therefore, we can remove the flow conservation constraints from the tree model. Furthermore, the constraints $y_{ij}^k = \sum_{h=1,K,(D/2)+1} z_{ij}^{hk}$ relating the two sets of flow variables permit us to rewrite the linking constraints ($y_{ij}^k \leq x_{ij}$ in the directed model and $y_{ij}^k + y_{ji}^k \leq x_e$ in the undirected model) using the z and x variables and so we can eliminate the y variables from the hop-indexed models. With these observations, it is easy to show that the directed models DMCF and HopDMCF are the same as the models described by Gouveia (1996,1998) for the fixed-root version of the problem with an additional degree constraint imposed on the extra node.

Finally, we note that the previously stated property that the extreme points of the linear programming relaxation of the Hop-Constrained Path model are integer-valued whereas, in general, the linear programming relaxation of the constrained path model has fractional extreme points permits us to show that the linear programming relaxation of the directed hop-indexed model HopDMCF is at least as strong as the linear programming relaxation of the directed model DMCF. Similarly, the linear programming relaxation of the undirected hop-indexed model HopUMCF is at least as strong as the linear programming relaxation of the undirected model UMCF. Formally, we have:

Proposition 3.1: $v(\text{HopDMCF}_L) \geq v(\text{DMCF}_L)$. $v(\text{HopUMCF}_L) \geq v(\text{UMCF}_L)$.

Proof. As we have previously noted for the linear programming relaxation, in the space of y variables, the feasible set of the hop-constrained path model is contained in the feasible set of the constrained path model. Adding the (directed) spanning tree constraints retains this property, which implies the stated result. ♦

The computational results given in Section 6 indicate that this dominance is strict for most cases. Using the directed hop-indexed formulation, the CPLEX solver was able to solve the previously mentioned 20 node and 100 edge DMST instance in less than four seconds.

3.2.2 The Situation When D is Odd

We have observed that our models for D even are simple modifications of models for the fixed-root version of the problem. As Figure 2 shows, for deriving a model for situations when D is odd, we need to be more elaborate. To model these situations, we must choose a single (central) edge {i,j} from the original network as well as two incident edges {0,i} and {0,j} (or two incident arcs (0,i) and (0,j) for the directed models). To do so, we let E_e be a zero-one variable indicating whether or not we choose the central edge e = {i,j} and impose the following edge selection constraints:

$$\sum_{e \in E} E_e = 1$$
$$x_{ok} = \sum_{e \in E(k)} E_e \quad \textit{for all } k \in V$$
$$E_e \in \{0,1\} \qquad \textit{for all } e \in E.$$

In these expressions, E(k) denotes the set of original edges incident to node k. In the space containing all of the x variables and the edge variables E_e from the original network, we let *ES* denote the set of feasible solutions to these Edge Selection constraints. Since the chosen edge e has two incident nodes, exactly two of the variable x_{ok} in this model will have value one. Consequently, we need not include the extra constraint stating that exactly two edges emanate from the root node, node 0. We also note a similar observation in the context of the corresponding linear programming relaxation. In fact, by addingthe constraints $x_{0k} = \sum_{e \in E(k)} E_e$ for all k ∈ V, we obtain $\sum_k x_{0k} = 2 \sum_{e \in E} E_e$. The equality $\sum_{e \in E} E_e = 1$ implies $\sum_k x_{0k} = 2$.

As in the situation when D is even, we obtain four different models. We might make one other observation about these models: if the network contains the central edge {j,k}, then it will contain both edges {0,j} and {0,k} and so we can send commodity k directly from node 0 to node k and not send any of this commodity on edge {0,j}. Consequently, we can replace the constraint $y_{oj}^k \le x_{oj}$ (j ≠ k) with the stronger constraint $y_{oj}^k \le x_{oj} - E_{jk}$.

Table IV. Models of the DMST when D is Odd

	Undirected	Directed
Path	**Model UEMCF** $minimize \sum_{e\in E_0} c_e x_e + \sum_{e\in E} c_e E_e$ $subject\ to\ (x,E) \in ES$ $(x,y) \in ST$ $y \in DP^k_{((D-1)/2)+1}\ for\ all\ k \in V$ $y^k_{0j} \le x_{\{0,j\}} - E_{jk}\ for\ all\ k,j \in V; k \ne j.$	**Model DEMCF** $minimize \sum_{(i,j)\in A_0} c_{ij} x_{ij} + \sum_{e\in E} c_e E_e$ $subject\ to\ (x,E) \in ES$ $(x,y) \in DST$ $y \in DP^k_{((D-1)/2)+1}\ for\ all\ k \in V$ $y^k_{0j} \le x_{0j} - E_{jk}\ for\ all\ k,j \in V; k \ne j.$
Hop	**Model HopUEMCF** $\text{min imize} \sum_{e\in E_0} c_e x_e + \sum_{e\in E} c_e E_e$ $\text{subject to } (x,E) \in ES$ $(x,y) \in ST$ $(z,y) \in DHP^k_{((D-1)/2)+1}\ for\ all\ k \in V$ $y^k_{oj} \le x_{oj} - E_{jk}\ for\ all\ k,j \in V; k \ne j.$	**Model HopDEMCF** $minimize \sum_{(i,j)\in A_0} c_{ij} x_{ij} + \sum_{e\in E} c_e E_e$ $subject\ to\ (x,E) \in ES$ $(x,y) \in DST$ $(z,y) \in DHP^k_{((D-1)/2)+1}\ for\ all\ k \in V$ $y^k_{oj} \le x_{oj} - E_{jk}\ for\ all\ k,j \in V; k \ne j.$

As we observed before, the two directed models satisfy the constraints $\sum_i x_{ik} = 1$ for all k ∈ V. The equalities $x_{0k} = \sum_{e\in E(k)} E_e$ in the set ES imply that $\sum_{i\ne 0} x_{ik} + \sum_{e\in E(k)} E_e = 1$ for all k ∈ V. That is, every node k ∈ V either receives an incoming arc from the original network or is an endpoint of the central edge. One way to view these models (e.g., by eliminating the variables x_{0k} but retaining the variables y^k_{0j}) is that we are establishing a node (the index 0 plays this role) in the middle of the chosen edge (with $E_e = 1$) with the property that the unique path from this node to any other node contains at most [(D-1)/2+1] edges or arcs. This formulation avoids the need to add nodes in the middle of each edge of the original network and then treat the added and original nodes differently since we do not require a path to added nodes.

As in the D even models, some constraints for the D odd models become redundant when we combine the formulations for the two core subproblems. As for the D even case, the hop-indexed models produce a better linear programming bound than the constrained path models. Formally, we have

Proposition 3.2: v (HopDEMCF$_L$) ≥ v (DEMCF$_L$). v (HopUEMCF$_L$) ≥ v (UEMCF$_L$).

Our computational results show that our models are not as successful for solving DMST instances when D is odd as when D is even. One possible explanation for this result might be given by the constraints linking the flow variables with arc (edge) design variables for the arcs (edges) emanating from the

additional node 0. As we have noted before, the four models for situations when is D even satisfy the equalities $y^k_{0j} = x_{0j}$ ($y^k_{0j} = x_{\{0,j\}}$) for all j,k ≠ 0 and thus, node k ≠ 0 with $x_{0k} = 1$ serves as a transshipment node for all the commodities. In the context of the D odd models, we cannot guarantee that one of the two nodes p and q with $x_{0p} = x_{0q} = 1$, say node p, will be a transshipment node for a given commodity k. The best we can say is that node p could be a transshipment node for that commodity (which is easily seen by combining the linking constraints for arc (0,p) (or edge {0,p}) with the corresponding flow conservation constraint). This behavior motivates, in a certain sense, the formulation described in Section 4.

3.2.3 Equivalent and more compact formulations when D is odd

By using an auxiliary augmented network, we have developed formulations for situations when D is even and D is odd. It is fairly easy to obtain equivalent formulations (in terms of corresponding linear programming relaxations) involving only variables associated with the original network.

As one such example, let us develop a reduced formulation with an equivalent linear programming relaxation to the model HopDEMCF. The equalities $x_{0k} = \sum_{e \in E(k)} E_e$ for all k ∈ V in the model HopDEMCF permit us to eliminate the variables x_{0k} (k ∈ V) from the model. Then, the cardinality constraint $\sum_{(i,j) \in A_0} x_{ij} = n$ becomes $\sum_{(i,j) \in A} x_{ij} + 2\sum_{e \in E} E_e = n$, which simplifies even further to $\sum_{(i,j) \in A} x_{ij} = n - 2$ since $\sum_{e \in E} E_e = 1$. The inequalities $z^{1k}_{0i} + E_{ik} \le x_{0i}$ for all i,k ∈ V (we ignore the term "$+ E_{\{i,k\}}$" when i = k) become $z^{1k}_{0i} \le \sum_{e \in E(i) \setminus \{i,k\}} E_e$.

Next, we use the equalities $\sum_{j \in V} z^{2k}_{ij} = z^{1k}_{0i}$ for all i,k ∈ V to remove the hop indexed flow variables associated with the arcs leaving the auxiliary root node (that is, the hop indexed variables with h = 1). The equalities $\sum_{i \in V} z^{1k}_{0i} = 1$ for all k ∈ V become $\sum_{i \in V} \sum_{j \in V} z^{2k}_{ij} = 1$. Finally, the inequalities $z^{1k}_{0i} \le \sum_{e \in E(i) \setminus \{i,k\}} E_e$ for all i,k ∈ V become $\sum_{j \in V} z^{2k}_{ij} \le \sum_{e \in E(i) \setminus \{i,k\}} E_e$. Note that when i = k, these inequalities reduce to $z^{2i}_{ii} \le \sum_{e \in E(i)} E_e$ because the model does not contain the variables z^{2i}_{ij} i,j ∈ V (i ≠ j). When i ≠ k the previous inequalities have the following intuitive interpretation: if any arc in position 2 emanates from a given node i and belongs in the path to node k, then the central edge should be adjacent to node i and should differ from the edge {i,k}. Table V depicts these modifications to the model:

Table V. Reducing the number of variables and constraints in the model HopDEMCF

Original Model	Reduced Model
$\sum_{(i,j)\in A_0} x_{ij} = n$	$\sum_{(i,j)\in A} x_{ij} = n-2$
$z_{0i}^{1k} + E_{ik} \le x_{0i}$ *for all* $i,k \in V$	$\sum_{j\in V} z_{ij}^{2k} \le \sum_{e\in E(i)\setminus\{i,k\}} E_e$ *for all* $i,k \in V$
$\sum_{i\in V} z_{0i}^{1k} = 1$ *for all* $k \in V$.	$\sum_{i\in V}\sum_{j\in V} z_{ij}^{2k} = 1$ *for all* $k \in V$.

To close this subsection, we note that the linear programming relaxation of the HopDMCF formulation always has an optimal integer solution for the DMST when D = 2 (as shown in Appendix 2). However, when D is odd even the strongest of our models, the model HopDEMCF, can have fractional extreme points. We describe exact formulations for the case D = 3 in Appendix 2. As a direct consequence of the study of the D = 3 case, in the next subsection we present a set of valid inequalities that permit us to strengthen the linear programming relaxation of HopDEMCF.

3.2.4 Valid inequalities when D is odd

The inequalities presented in this section are based on an observation concerning any optimal solution when D is odd, namely, any node linked to the central edge is always linked to the closest endpoint of the central edge. For every non-central arc (i,j) $\in$ A, let L(i,j) denote a set of potential central edges {i,p} $\in$ E whose endpoint i is not farther to node j than is node p. That is, L(i,j) = {{i,p} $\in$ E: p $\neq$ j and $c_{ij} \le c_{pj}$}. The previous observation concerning optimal solutions for situations when D is odd indicates that if an optimal solution contains an arc (i,j) connected to the central edge, then L(i,j) contains the central edge.

We show next how to use information from the sets L(i,j) to derive valid inequalities for situations when D is odd. These inequalities are based on disaggregated versions of the inequalities $\sum_{j\in V} z_{ij}^{2k} \le \sum_{e\in E(i)\setminus\{i,k\}} E_e$ (for all i,k $\in$ V and i $\neq$ k) described in the previous subsection. We start by weakening the these inequalities into $z_{ij}^{2k} \le \sum_{e\in E(i)\setminus\{i,k\}} E_e$ for all (i,j) $\in$ A and k $\in$ V. We can interpret these weaker inequalities as indicating that if arc (i,j) is in position 2 on the path to node k, then the central edge should be adjacent to node i and should differ from the edge {i,k}. However, by our previous observation concerning optimal solutions for situations when D is odd, we know that the central edge is restricted to the set L(i,j)\{i,k}. That is, we can strengthen these inequalities to $z_{ij}^{2k} \le \sum_{e\in L(i,j)\setminus\{i,k\}} E_e$ for all (i,j) $\in$ A and k $\in$ V.

The new set of constraints permit us to obtain two new models for the situations when D is odd. We obtain one of the new models, denoted HopDEMCF+, by adding the new constraints to the model HopDEMCF. We obtain the other, denoted HopDEMCF* by replacing the $O(n^2)$ constraints

$\sum_{j\in V} z_{ij}^{2k} \le \sum_{e\in E(i)\setminus\{i,k\}} E_e$ (for all i,k ∈ V and i ≠ k) in HopDEMCF by the new set of O(n*|A|) constraints $z_{ij}^{2k} \le \sum_{e\in L(i,j)\setminus\{i,k\}} E_e$ for all (i,j) ∈ A and k ∈ V. Consequently, the original model HopDEMCF differs from the model HopDEMCF* only concerning the set of constraints relating the hop-indexed variables z_{ij}^{2k} with the central edge variables E_e. Table VI depicts the differences between the three models:

Table VI. The three directed hop-indexed models for D odd

Model HopDEMCF	Model HopDEMCF+	Model HopDEMCF*
$z_{ii}^{2i} \le \sum_{e\in E(i)} E_e$ *for all* $i \in V$	$z_{ii}^{2i} \le \sum_{e\in E(i)} E_e$ *for all* $i \in V$	$z_{ii}^{2i} \le \sum_{e\in E(i)} E_e$ *for all* $i \in V$
$\sum_{j\in V} z_{ij}^{2k} \le \sum_{e\in E(i)\setminus\{i,k\}} E_e$ *for all* $i,k \in V$ *and* $i \ne k$.	$\sum_{j\in V} z_{ij}^{2k} \le \sum_{e\in E(i)\setminus\{i,k\}} E_e$ *for all* $i,k \in V$ *and* $i \ne k$	$z_{ij}^{2k} \le \sum_{e\in L(i,j)\setminus\{i,k\}} E_e$ *for all* $(i,j) \in A$ *and* $k \in V$.
	$z_{ij}^{2k} \le \sum_{e\in L(i,j)\setminus\{i,k\}} E_e$ *for all* $(i,j) \in A$ *and* $k \in V$.	

Clearly, the linear programming lower bound given by HopDEMCF+ cannot be worse than the linear programming lower bound given either by HopDEMCF or HopDEMCF*. Thus, we have the following result.

Proposition 3.3. $v(\text{HopDEMCF+}_L) \ge v(\text{HopDEMCF}_L)$. $v(\text{HopDEMCF+}_L) \ge v(\text{HopDEMCF*}_L)$.

Our computational results show that these inequalities can be strict. In general, neither of the linear programming models HopDEMCF and HopDEMCF* dominates the other. Our computational results show that for random cost graphs the new model HopDEMCF* produces a better bound than the original model HopDEMCF. The situation is reversed for the instances with Euclidean costs.

Using the new model HopDEMCF+, we have been able to solve problems that we could not solve using the HopDEMCF model because its linear programming lower bound was not tight enough. We note, however, that in some cases, the extra constraints in HopDEMCF+ have contributed to huge solution times to obtain the integer solution. In some of these cases, the third model, the model HopDEMCF* which produces a weaker linear programming bound but is more compact than the strongest model, proved to be a viable alternative for obtaining the optimal integer solution.

We also note that due to the inclusion of the new set of constraints, the linear programming relaxation of the models HopDEMCF+ and HopDEMCF* always have an optimal integer solution for the DMST when D = 3 (the Appendix 1 establishes this result). This observation also implies that $v(\text{HopDEMCF*}_L) \ge v(\text{HopDEMCF}_L)$ when D = 3.

4. Enhanced Formulations for the DMST when D is odd

The formulations described in this section are also based on our prior observation that in an optimal solution to situations when D is odd, any node linked directly to the central edge is always linked to the closest endpoint of the central edge. In this section we use this property to construct a different model. We will contract every possible central edge {p,q} into a supernode and replace the two arcs connecting the nodes p and q to any other node j by the shorter of the arcs (p,j) and (q,j).

This observation permits us to model situations when D is odd as a special version of the D even problem in an appropriately expanded graph. Let G = (V,E) be the original undirected graph. We define the expanded graph $G_E = (V_E,E_E)$ as follows:

$$V_E = V \cup \{\{i,j\}: \{i,j\} \in E\}$$

$$E_E = E \cup \{\{\{i,j\},k\}: \{i,j\} \in V_E \setminus V,\ k \in V\setminus\{i,j\} \text{ and } (\{k,i\} \in E \text{ or } \{k,j\} \in E)\}.$$

The set V_E includes all the nodes in V plus a new set of nodes, the "supernodes," corresponding to edges in E. The set E_E includes all the edges in E plus a new set of edges linking each supernode to the neighbors of either of the endnodes of the edge corresponding to the supernode. The number of edges in the path from the supernode (say {p,q}) to any other node in T_E is at most (D–1)/2 if and only if the number of edges in the path from the central edge {p,q} is at most (D–1)/2 in T.

Based on our experience with the models presented in the previous section, we shall cast our discussion in terms of only directed models. Thus, we replace each edge $e = \{i,j\} \in E_E$ of the undirected expanded graph by two directed arcs (i,j) each with the same cost as the edge e and obtain a directed expanded graph $G_E = (V_E,A_E)$. We replace an edge of the form {{i,j},k} in the undirected graph by only the single arc ({i,j},k) in the directed graph. We define the arc cost structure in G_E as follows. If an arc in G_E is of the form (i,j) with $i,j \in V$, then it has the same cost as it has in the original graph G. Otherwise, the arc has the form ({i,j},k) and we set its cost equal to the minimum of c_{ik} and c_{jk}. The definition of the new arcs and the way their costs are defined is typical of graph theory algorithms that shrink several nodes into supernodes (Edmonds' minimum cost arborescence algorithm is an example). Using the definition of the arc costs in A_E, it is easy to transform any feasible tree T_E in G_E into a feasible tree T in G with the central edge {p,q}. Furthermore, the definition of the costs in G_E guarantees that T_E is optimal in G_E if and only if T is optimal in G.

Following our discussion of Section 3, we will augment the expanded graph by adding an additional node 0 and zero cost arcs connecting the root node to each supernode. Figure 3 illustrates this transformation. In this augmented graph, we wish to select a tree T_E containing a single arc emanating from

node 0, spanning a single supernode {p,q}, and all the original nodes in V except nodes p and q. The tree must satisfy the property that the unique path from node 0 to every node contains at most (D-1)/2 + 1 arcs.

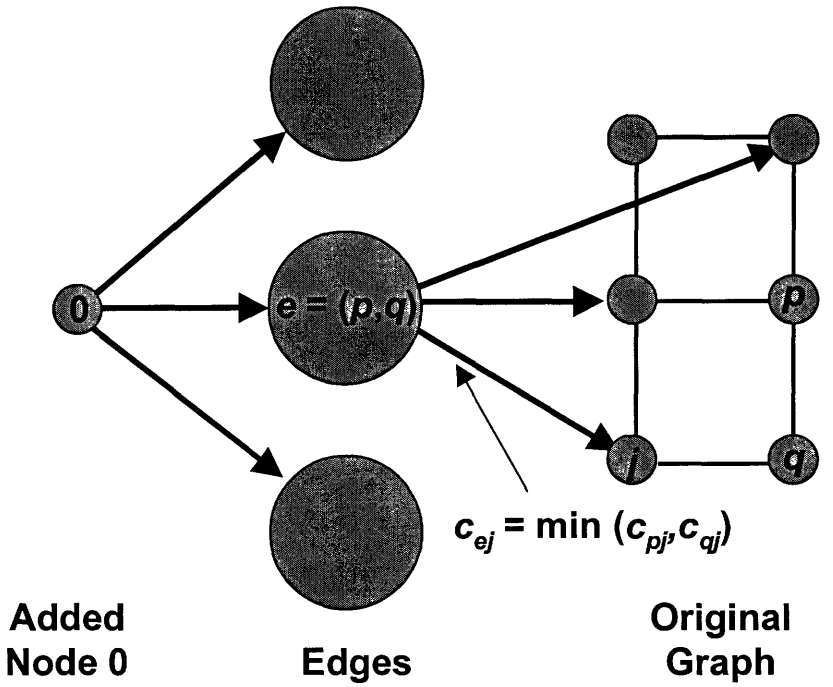

Figure 3 - Augmented Enhanced Network for Situations with D Odd

The resulting model is similar to the problem discussed in Section 3 for situations when D is even. Indeed, we can write the DMCF and HopDMCF models for situations when D is odd in the context of this augmented graph with minor modifications of the models when D is even. The variables x and y will now have indices corresponding to three types of arcs:

(i) the arcs (0,e) for e = {p,q} $\in$ E,

(ii) the arcs (e,j) for e = {p,q} $\in$ E, j $\in$ V\{p,q} and ({p,j} $\in$ E or {q,j} $\in$ E)}.

(iii) the arcs (i,j) for (i,j) $\in$ A.

Table VII. Modeling a Directed Rooted Restricted Tree in the Augmented Expanded Graph

Directed Restricted Tree Model

$$\text{minimize} \sum_{(i,j)\in A_E} c_{ij}x_{ij}$$

$$\text{subject to} \sum_{(i,j)\in A_E} x_{ij} = n-1$$

$$\sum_{e\in E} x_{0e} = 1$$

$$\sum_{j\in V} y_{ij}^k - \left(\sum_{j\in V} y_{ji}^k + \sum_{e\in E\setminus(E(i)\cup E(k))} y_{ei}^k\right) = \begin{cases} 0 & \text{for all } i,k\in V;\ i\neq k \\ \sum_{e\in E(k)} x_{0e} - 1 & \text{for all } i,k\in V;\ i=k \end{cases}$$

$$\sum_{j\in V\setminus\{p,q\}} y_{ej}^k - y_{0e}^k = \begin{cases} 0 & \text{for all } k\in V, e=\{p,q\}\in E\setminus E(k) \\ -x_{0e} & \text{for all } k\in V, e=\{p,q\}\in E(k) \end{cases}$$

$$\sum_{e\in E} y_{0e}^k = 1 \quad \text{for all } k\in V$$

$$y_{ij}^k \le x_{ij} \quad \text{for all } (i,j)\in A_E, k\in V$$

$$y_{ij}^k \ge 0 \quad \text{for all } (i,j)\in A_E, k\in V$$

$$x_{ij}\in\{0,1\} \quad \text{for all } (i,j)\in A_E.$$

Based on our observation motivating this formulation, when e = {p,q} the cost c_{ej} (e ∈ E; j ∈ V\{p,q}) is given by c_{ej} = min {c_{pj},c_{qj}}. The first two constraints state that the tree contains n − 1 arcs and that exactly one arc leaves the auxiliary root node. The conservation flow constraints for the nodes in V state that when i = k, either (i) $\sum_{e\in E(k)} x_{oe} = 0$ and node k receives one unit of flow from the root node (note that the model does not contain the variables y_{kj}^k for all k,j ∈ V) and node k is included in the tree, or (ii) $\sum_{e\in E(k)} x_{oe} = 1$ and node k does not receive flow from the root node (we use again the fact that the model does not contain the variables y_{kj}^k for all k,j ∈ V) and thus, it is not included in the tree. When i ≠ k, the corresponding constraint simply states that node i is a transshipment node for commodity k. The flow conservation constraints for the nodes e ∈ E state that if e = {k,p} for some p ∈ V, then node e receives one unit of flow of commodity k if and only if arc (0,e) is in the solution (the model does not contain the variables y_{ej}^k for all k,j ∈ V, e ∈ E and e = {k,p} for some p in V). If e = {p,q} for p,q ≠ k, then node e is a transshipment node for commodity k. The remaining constraints are self-explanatory.

To obtain valid formulations when D is odd, we need to combine this model with simple adaptations of the path models described in the previous section. In this way, we shall obtain an enhanced directed multicommodity flow model (Enh-DMCF) and an enhanced directed hop indexed flow model (Enh-HopDMCF). For simplicity we do not formally state the complete models (we refer the reader to the Appendix 1 which contains a complete description of the model Enh-HopDMCF).

It is possible to show that the linear programming relaxation of the enhanced models Enh-DMCF and Enh-HopDMCF dominate the linear programming relaxation of DEMCF and HopDEMCF+ respectively. That is,

Proposition 4.1 $v(\text{Enh-DMCF}_L) \geq v(\text{DEMCF}_L)$. $v(\text{Enh-HopDMCF}_L) \geq v(\text{HopDEMCF+}_L)$.

As demonstrated by our computational results, the inequalities can be strict. The proof of this full result is quite elaborate and so we omitted from this discussion. Appendix 1 provides a proof of half of the result, namely that the linear programming relaxation of the model Enh-HopDMCF dominates the linear programming relaxation of the model HopDEMCF+.

Our computational results show that the lower bound given by the linear programming relaxation of the new model Enh-HopDMCF is in general significantly better than the lower bound given by the linear programming relaxation of the models described in the previous section for situations when D is odd. In fact, the lower bounds given by this model for situations with D odd are close to the optimal integer value and nearly of the same quality as the lower bounds given by DMCF when D is even. However, there is an obvious disadvantage to using the Enh-HopDMCF formulation. Notice that while the original graph G has n nodes and m edges, the expanded graph G_E contains n + m nodes and O(nm) edges. The number of variables and constraints in the new model is substantially larger than the number of variables and constraints in the other models. In fact, our computational results show that either the linear programming relaxation of the new model is too large to be solved or the CPU times needed to solve its linear programming relaxation are quite large when compared with the CPU times of the models described in the previous section.

Besides being a good choice for solving small-sized instances of problems with D odd, we also note that the study of this new model together with the study of the D = 3 case suggested the inequalities described in Section 3.2.4. These inequalities have permitted us to solve instances that are not solved by either the original HopDEMCF model (its linear programming lower bound was not tight enough) or the enhanced model Enh-HopDMCF (its linear programming model was too large to be solved or requires too much time to be solved).

5. Minimum Steiner Trees With Diameter Constraints

In many applications the tree need not connect all the nodes of the network. The set of nodes is partitioned into two sets, R and S. In telecommunications settings, the elements of the set R usually are terminals that must be connected to each other and the set S represents switching nodes. The tree is required to span all nodes in R and might or might not include some of the nodes in S, the so-called Steiner nodes. The problem defined in this way is the classical Minimum Steiner Tree Problem (see Hwang, Richards and Winter (1992) and Magnanti and Wolsey (1996)). As mentioned in the introduction, quality

of service requirements might suggest a diameter constraint stating that the number of edges in the tree path between every pair of nodes in R does not exceed a given value D. When R = V, we obtain the DMST problem discussed in the previous sections. When |R| = 2, the problem becomes a shortest path problem between two specified nodes with an additional constraint stating that the path cannot contain more than D edges. This problem is modeled simply as an unconstrained shortest path problem in an appropriate graph and the solution for this problem provides the underlying idea for creating the hop-indexed models discussed in Section 3.

It is easy to modify all the models we have presented for the DMST problem for the Steiner version of the problem: (i) the commodity indices in the formulations range only over the set R instead of the entire node set V; and (ii) we replace the constraint stating that the number of edges (arcs) in the spanning tree is equal to |V| – 1 by a constraint stating that the number of edges (arcs) in the tree should be greater or equal to |R| – 1.

Because of modification (i), when |R| is small our models would be small and so more easily solvable. Computational results presented in Section 8 for the diameter Steiner tree problem with |R| = |V|/2 and R = |V|/4 confirm that these Steiner instances are easier to solve than the corresponding spanning tree versions.

6. Computational Experience

6.1 Problem Instances

To understand what might be achievable with the models discussed in this paper, we have generated several problem instances with up to 100 nodes and 1000 edges. We have considered two groups of instances: random cost instances and Euclidean instances. For each value of n (number of nodes) and m (number of edges) we have generated different random and Euclidean instances. We have used a uniform distribution in the interval [1,100] to obtain the cost of each edge included in each random cost instance. To obtain the Euclidean instances, we have i) randomly generated the coordinates of n nodes in a 100 by 100 square grid, ii) selected the cost of each candidate edge (i,j) as the integer part of the Euclidean distance between the two nodes i and j, and iii) defined the edge set E associated with each instance by choosing the m least cost edges of the corresponding complete graph. The directed models will contain twice as many arcs as the undirected models. For each instance, we have tried several values of the diameter parameter ranging from 4 to 8.

Several of the generated instances are infeasible for small values of the parameter D. Thus, for some instances, we have omitted the results for D = 4 and 5. The problems are infeasible because of the way we have selected the edges for each instance. To address this concern, we have conducted a few preliminary results with Euclidean instances generated in a different manner. First, we select the edges of the minimum star solution. Then, we select the m – (n-1) least cost edges of the corresponding complete graph minus the

minimum star edges. Our results have shown that these instances are much easier to solve than the ones tested in this paper. We believe these problems are easier to solve because the minimum star solution might give some relevant information concerning the optimal integer solution for D greater than 2. We noticed, for instance, that when D is even, the root of the minimum star is the root of the optimal integer solution of many of the instances tested.

We performed all tests in a Pentium II, 450 Mhz computer. We used the CPLEX 7.0 package to solve the linear programming models and to obtain the optimal integer values.

6.2 Situation When D is Even

Table VIII and IX summarize our results for the random and Euclidean instances with n = 20 and 30 and D even, i.e., D = 4, 6 and 8. The first column specifies the number of nodes, edges and required nodes. We choose the set R arbitrarily. The second column specifies the value of D. The next four columns depict the gaps given by the optimal linear programming bound of the models MCF, UMCF, DMCF and HopDMCF. These columns indicate the value [(OPT – LB)/OPT]*100 (OPT is the value of the optimal solution and LB is the value of the lower bound given by the optimal linear programming solution of the model indicated at the top of the column). The last column indicates the optimal value. We have restricted the results corresponding to MCF and UMCF to the n = 20 instances and have truncated the lower bounds to the first decimal digit. We specify two values beneath the reported lower bound. The first value is the CPU time needed to solve the linear programming relaxation. To assess the quality of the lower bounds, using branch-and-bound we have tried to obtain the optimal solution or an upper bound value on the optimum. The second value, when given, is the additional CPU time needed to obtain the optimal value. CPU times are given in seconds.

Our results show that the multi-source/multi-destination model MCF is not able to solve instances with 20 nodes. Single sourcing allows us to solve many such problems. The reported CPU times indicate that we can solve these instances, though with some difficulty, using the undirected model UMCF. On the other hand, the directed model DMCF solves these instances rather easily, but requires huge CPU times to solve the n = 30 instances. The hop-indexed model is able to quickly solve the n = 20 and 30 instances.

|V|,|E|,|R|	D	MCF	UMCF	DMCF	HopDMCF	Optimum
20,100,10	4	30.0 (25 + 41782)	38.2 (2 + 953)	9.7 (2 + 108)	0.0 (1 + 0)	138
20,100,10	6	29.1 (16 + 24100)	38.5 (2 + 2098)	2.5 (4 + 60)	0.0 (2 + 0)	119
20,100,10	8	29.0 (8 + 9397)	39.3 (2 + 2663)	2.1 (2 + 195)	0.0 (8 + 1)	115
20,100,20	4	22.4 (375 + ?)	25.7 (14 + 2530)	13.8 (17 + 1134)	0.0 (4 + 0)	233
20,100,20	6	15.3 (68 + ?)	16.8 (6 + 38594)	12.5 (11 + 4400)	1.0 (11 + 35)	178
20,100,20	8	4.8 (33 + ?)	5.3 (4 + 12628)	4.0 (6 + 499)	0.0 (10 + 0)	154
30,200,15	4			12.4 (29 + 1409)	0.0 (2 + 0)	159
30,200,15	6			7.7 (12 + 1371)	0.0 (5 + 0)	108
30,200,15	8			0.0 (4 + 0)	0.0 (5 + 0)	96
30,200,30	4			11.4 (182 + 7373)	0.0 (6 + 0)	234
30,200,30	6			9.5 (69 + 18135)	0.0 (22 + 1)	157
30,200,30	8			1.1 (28 + 537)	0.0 (17 + 1)	135

Table VIII – Results for the RANDOM instances with n = 20, 30 and even values of D. The specification ? indicates that we did not attempt to solve the integer program.

We also tested the formulation using the modified Miller-Tucker-Zemlin (MTZ) constraints given in Achuthan, Caccetta, Caccetta and Geelen (1994) for the spanning instances with 20 and 30 nodes. This formulation uses a small number of variables and constraints when compared to the other formulations. Thus, we were able to obtain the corresponding linear programming bounds very quickly. On the other hand, these lower bounds are rather weak and for most of the cases tested, we have obtained huge search trees when attempting to obtain the optimal integer solution. In fact, we could not solve any of the Euclidean 30 node instances due to memory storage limitations. Moreover, the best lower bound obtained at the moment we ran out of memory were still far from the optimal solution (as an example, for the Euclidean instance with D = 8 the best lower bound was equal to 410 after nearly 40,000 seconds of computation). It is interesting to note that for all the cases tested the MTZ lower bounds are worse than the trivial lower bounds given by the cost of the unconstrained spanning tree. Consider, for instance, the previous Euclidean example with n=30 and D = 8. The MTZ linear programming bound is equal to 340.4 while the minimum spanning tree bound is equal to 396.

|V|,|E|,|R|	D	MCF	UMCF	DMCF	HopDMCF	Optimum
20,100,10	4	12.2 (44 + 38150)	22.4 (3 + 532)	0.8 (2 + 21)	0.0 (0 + 0)	235
20,100,10	6	16.2 (26 + 176423)	26.1 (2 + 3778)	0.0 (1 + 0)	0.0 (1 + 0)	217
20,100,10	8	22.1 (8 + 86209)	31.4 (2 + 2224)	2.1 (1 + 0)	0.0 (4 + 0)	217
20,100,20	4	2.8 (684 + ?)	7.6 (20 + 2790)	1.3 (16 + 186)	0.0 (1 + 0)	369
20,100,20	6	2.2 (216 + ?)	6.1 (9 + 2864)	1.0 (15 + 241)	0.2 (6 + 19)	322
20,100,20	8	2.5 (97 + ?)	5.4 (6 + 11907)	0.7 (10 + 887)	0.0 (9 + 0)	308
30,200,15	4			3.5 (25 + 771)	0.4 (3 + 4)	338
30,200,15	6			2.7 (12 + 882)	0.1 (10 + 3)	289
30,200,15	8			0.5 (13 + 80)	0.0 (32 + 0)	274
30,200,30	4			5.5 (258 + 22179)	1.7 (20 + 1026)	599
30,200,30	6			3.2 (298 + 23639)	0.8 (156 + 10899)	482
30,200,30	8			3.4 (94 + 21717)	0.8 (319 + 8569)	437

Table IX – Results for the EUCLIDEAN instances with n = 20, 30 and even values of D. The specification ? indicates that we did not attempt to solve the integer program.

Table X specifies the gaps given by the linear programming bound of the hop-indexed model HopDMCF for the instances with n = 40, 60, 80 and 100. As before, the first column of this new table indicates the number of nodes, number of edges and number of required nodes of the corresponding instance. The next three columns refer to the random instances and the remaining three columns refer to the Euclidean instances. Each group of three columns specifies the gaps for D = 4, 6 and 8 respectively. Each entry is similar to the entries in the previous tables. That is, the value on top center of any cell is the gap multiplied by 100 and the value on the right-hand corner is the optimal integer value. The values on the bottom of a cell indicate, respectively, the CPU times for obtaining the linear programming solution and the optimal integer solution. As noted before, and as indicated by the designation INF, the corresponding problem may be infeasible in some cases.

		Random			Euclidean	
|V|,|E|,|R|	4	6	8	4	6	8
40,400,20	1.6 165 (16 + 10)	2.7 106 (74 + 398)	0.0 90 (49 + 1)	0.0 392 (17 + 1)	0.7 326 (84 + 243)	0.0 303 (167 + 1)
40,400,40	0.0 309 (86 + 1)	0.0 189 (445 + 2)	0.0 161 (277 + 3)	0.04 672 (135 + 34)	0.6 555 (1801 + 35054)	0.5 507 (4865 + 46166)
60,600,15	0.0 115 (9 + 1)	2.8 70 (65 + 448)	0.0 52 (41 + 2)	0.0 326 (8 + 1)	1.6 292 (164 + 1012)	0.0 268 (181 + 1)
60,600,30	0.9 255 (64 + 161)	1.6 120 (434 + 4967	0.0 95 (262 + 4)	0.0 707 (33 + 2)	0.3 505 (819 + 19131	0.3 464 (3055 + 13023)
60,600,60	0.7 326 (340 + 11943)	1.3 175 (12037 + t))	0.0 127 (1567 + 129)	0.1 1180 (260 + 1282)	0.5 837 (12397 + t)	m)
80,800,10	0.0 62 (4 + 1)	0.0 36 (27 + 0)	0.0 34 (13 + 1)	0.0 277 (5 + 1)	0.0 220 (50 + 1)	0.0 214 (207 + 1)
80,800,20	0.0 140 (299 + 2)	0.0 82 (27 + 0)	0.0 71 (473 + 3)	0.0 567 (6 + 1)	0.8 417 (639 + 3749)	0.7 377 (2284 + 12139)
80,800,40	3.8 249 (145 + 4486)	0.0 125 (925 + 4)	0.0 104 (2068 + 5)	0.0 939 (28 + 3)	1.3 627 (3652 + 115866)	0.0 552 (9504 + 6)
80,800,80	5.7 424 (1082 + 105273)	m)	m)	m)	m)	m)
100,1000,12	2.7 55 (5 + 10)	0.0 40 (43 + 2)	0.0 40 (168 + 2)	0.0 55 (4 + 1)	0.5 466 (181 + 219)	0.0 427 (489 + 2)
100,1000,25	2.7 123 (33 + 60)	0.0 75 (401 + 3)	0.0 67 (920 + 5)	0.0 894 (10 + 2)	0.4 679 (612 + 6542)	0.5 630 (1647 + 17531)
100,1000,50	2.1 274 (341 + 15853)	1.3 153 (10304 + 99319)	0.0 127 (9987 + 10)	INF	m)	m)

Table X – Results for the RANDOM and EUCLIDEAN instances for the model HopDMCF with $n \geq 40$ and even values of D. The specification t) indicates that we did not obtain the optimal solution after 2 days of CPU time (the value shown on the right-hand top is an upper bound). The designation m) indicates that we could not solve the problem due to memory requirements.

The results show that the HopDMCF model maintains the behavior already shown for the smaller instances, namely that its linear programming bound is quite good. The CPU times required to obtain the optimal linear programming solution increase with the diameter parameter value for the Euclidean instances and, in certain cases, are huge. Notice, however, that the largest model being solved (the HopDMCF model with n = 80, m = 800, |R| = 80 and D = 4) contains about 250,000 integer variables (but far fewer constraints, 130,000). The results also show that the CPU times needed for obtaining the optimal integer solutions of the Euclidean instances are, in general, much higher than the CPU times needed for solving the random instances.

6.3 Situation When D is Odd

For the instances with D odd we have restricted our computational experiments to the models DEMCF, HopDEMCF, HopDEMCF+, HopDEMCF* and Enh-HopDMCF (the results for D even suggest that the traditional MCF model and the undirected models are not worth comparing with the previous four models). Tables XI and XII have the same format as Tables VIII and IX and contain the results for the random and Euclidean instances with n = 20 and 30 and D odd, that is, D = 5 and 7. Tables XI and XII also empirically compare the linear programming relaxations of the HopDEMCF, HopDEMCF+, HopDEMCF* and Enh-

HopDMCF models. We have indicated the "best" model by designating in bold the CPU times corresponding to the model that most quickly produces the optimal integer solution.

|V|,|E|,|R|	D	DEMCF	HopDEMCF	HopDEMCF+	HopDEMCF*	Enh-HopMCF	Optimum
20,100,10	5	6.9 (1 + 145)	0.0 **(0 + 0)**	0.0 **(0 + 0)**	0.0 **(0 + 0)**	0.0 (11 + 1)	120
20,100,10	7	9.7 (1 + 33)	5.1 (1 + 41)	0.0 **(1 + 0)**	2.7 (1 + 33)	0.0 (70 + 1)	116
20,100,20	5	22.3 (8 + 4190)	8.5 (2 + 132)	2.4 (4 + 84)	4.1 (2 + 57)	0.0 **(49 + 1)**	205
20,100,20	7	12.2 (5 + 1241)	6.7 (4.4 + 272)	1.5 **(8 + 47)**	4.3 (3 + 215)	1.5 (442 + 310)	165
30,200,15	5	19.4 (8 + ?)	9.2 **(2 + 95)**	3.7 (5 + 115)	4.7 (3 + 104)	0.0 (157 + 4)	132
30,200,15	7	3.7 (6 + ?)	2.5 (3 + 3)	0.0 (6 + 0)	0.0 **(3 + 1)**	0.0 (880 + 4)	98
30,200,30	5	20.4 (43 + ?)	6.6 (7 + 1211)	0.0 **(9 + 1)**	2.8 (5 + 245)	0.0 (462 + 7)	195
30,200,30	7	7.8 (25 + ?)	4.4 (16 + 957)	0.0 **(29 + 107)**	0.5 (19 + 207)	0.0 (4228 + 7)	144

Table XI – Results for RANDOM instances when n = 20, 30 and D is odd. The specification ? indicates that we did not attempt to solve the integer program..

|V|,|E|,|R|	D	DEMCF	HopDEMCF	HopDEMCF+	HopDEMCF*	Enh-HopMCF	Optimum
20,100,10	5	8.8 (1 + 71)	6.8 (0 + 40)	0.0 **(2 + 0)**	4.9 (11 + 1)	0.0 (10 + 0)	225
20,100,10	7	6.3 (1 + 92)	5.5 **(1 + 60)**	1.7 (9 + 314)	7.3 (2 + 417)	0.0 (142 + 1)	217
20,100,20	5	8.7 (12 + 3855)	4.6 (3 + 179)	2.2 (7 + 238)	4.9 (2 + 287)	0.0 **(41 + 1)**	347
20,100,20	7	6.1 (6 + 1219)	3.7 (6 + 958)	1.1 (14 + 414)	3.2 (4 + 541)	0.0 **(395 + 1)**	316
30,200,15	5	12.2 (19 + ?)	6.7 (4 + 1004)	2.4 (24 + 806)	11.8 (4 + 1156)	0.0 **(95 + 3)**	309
30,200,15	7	7.6 (10 + ?)	5.1 **(25 + 964)**	2.1 (78 + 1238)	9.4 (9 + 1914)	0.0 (6976 + 3)	278
30,200,30	5	11.2 (104 + ?)	5.8 (22 + 8257)	4.2 (61 + 17947)	10.2 (7 + 31496)	0.1 **(195 + 6)**	534
30,200,30	7	9.8 (44 + ?)	5.7 (47 + 102013)	3.8 **(140 + 62216)**	6.0 (32 + 77941)	1.2 (31168 + 24129)	463

Table XII – Results for EUCLIDEAN instances when n = 20, 30 and D is odd. The specification ? indicates that we did not attempt to solve the integer program.

The results show that the model HopDEMCF is not as successful in solving DMST instances when D is odd as is the HopDMCF model for solving DMST instances when D is even. As a consequence, we have used valid inequalities for the D odd case (see Section 3.2.4) and have derived the Enh-HopDMCF formulation with all the edges shrunk into nodes. Our results show that from a theoretical perspective, the linear programming relaxation of this enhanced model is quite good. In fact, for most of the Euclidean instances tested, this model proved to be the best choice for obtaining the optimal integer solution. Unfortunately, because the enhanced model contains so many variables and constraints, it may become

impractical computationally for larger problem instances. For instance, we encountered computer storage limitations when attempting to solve several instances with this formulation with n = 40 and m = 400.

The HopDEMCF+ model with the addition of n*|A| inequalities implied by the enhanced formulation provides a compromise. Our results for the n = 20 and 30 instances show the importance of the HopDEMCF+ model when solving some of the instances with odd values of D. In most of the cases, the formulation reduced the gap given by the original HopDEMCF model by more than half. For the random cases, the HopDEMCF+ produces zero gaps in several cases and proved to be a sound alternative to the enhanced model. Unfortunately, the results are not as good for the Euclidean cases. These instances proved to be more difficult to solve. It is interesting to note that the "alternate" hop-indexed model HopDEMCF* is also worth using for the random cases. For the Euclidean cases, the linear programming bounds are quite bad. In fact, in a few situations, it produces a lower bound that is worse than the one obtained by the model DEMCF.

The next two tables, Tables XIII and XIV, present results for situations when D is odd for larger instances. We do not report results from the weak DEMCF model and the huge enhanced model Enh-HopDMCF (as we have mentioned before, we have tried unsuccessfully, to solve some of these instances). For the Euclidean instances, we do also not show the results from the model HopDEMCF* as the results of Table X indicate that this model behaves rather poorly for this class of instances.

The results demonstrate the relevance of the models developed in Section 3.2.4. The HopDEMCF+ and HopDEMCF*models (this one in the context of the Random instances), have helped to solve instances that we have not been able to solve with the original hop-indexed model HopDEMCF. Unfortunately, with these improvements, our best models for the situations when D is odd are not of the same quality level as our models for the situations when D is even.

As before, we have indicated the "best" model by designating in bold the CPU times corresponding to the model that most quickly produces the optimal integer solution. We note that the CPU times needed for obtaining the optimal integer solution depend strongly on the behavior of the integer programming package, namely in how it obtains a good upper bound. In several cases, the linear programming bound given by the model HopDEMCF+ is at least as good as the best lower bound obtained by the original model HopDEMCF after 1 day of computations of the branch-and-bound code. However, in a few of these cases, the strengthened HopDEMCF+ requires a reasonable amount of time to obtain the optimal integer solution because it finds a good upper bound rather late in the optimization process. One example of this is the instance with n = |R| = 60, D = 7 and 300 edges. After 10,000 seconds of computation of the branch-and-bound method using the HopDEMCF+ model, the best lower bound was already within one unit of the optimal integer solution. Unfortunately, the computations yielded an upper bound corresponding to the optimal value rather late in the optimization method yielding about 90,000 seconds of CPU time.

|V|,|E|,|R|	D	HopDEMCF	HopDEMCF+	HopDEMCF*	Optimum
40,400,20	5	7.5 (14 + 737)	2.5 (22 + 890)	6.5 **(10 + 372)**	128
40,400,20	7	9.4 (39 + 1381)	2.9 (99 + 749)	6.0 **(29 + 233)**	101
40,400,40	5	9.1 (56 + 13529)	5.6 (104 + 23645)	7.5 **(40 + 8642)**	253
40,200,40	7	1.0 (114 + 209)	0.0 (174 + 4)	0.0 **(69 + 4)**	171
60,600,15	5	16.1 (7 + 667)	6.0 (13 + 640)	9.4 **(9 + 477)**	88
60,600,15	7	9.0 (34 + 769)	1.6 (139 + 183)	7.7 **(25 + 715)**	61
60,600,30	5	8.5 (33 + 12240)	3.5 **(62 + 3318)**	7.0 (30 + 6425)	160
60,600,30	7	7.7 **(150 + 5313)**	2.1 (395 + 7093)	6.4 (90 + 5897)	109
60,600,60	5	14.8 (142 + t))	10.0 (280 + t))	13.5 (96 + t))	257 u)
60,600,60	7	7.2 **(686 + 61113)**	2.9 (1795 + t))	6.0 (551 + 85466)	150
80,800,10	5	16.9 (5 + 188)	9.7 (6 + 234)	14.5 **(5 + 184)**	46
80,800,10	7	14.7 (15 + 294)	0.0 **(89 + 2)**	5.8 (18 + 1011)	34
80,800,20	5	9.4 **(17+ 2252)**	4.8 (33 + 2405)	8.6 (19 + 2455)	111
80,800,20	7	8.0 (164 + 4408)	3.8 **(253 + 3049)**	8.5 (114 + 8836)	78
80,800,40	5	12.3 (74+ 50355)	7.8 (124 + 103318)	12.2 **(58 + 19673)**	186
80,800,40	7	3.6 (418 + 25887)	0.0 **(1306 + 10)**	2.8 (362 + 120191)	114
100,1000,12	5	5.1 (7 + 483)	0.0 **(7 + 3)**	0.0 (8 + 9)	45
100,1000,12	7	4.2 (30 + 1038)	0.0 **(59 + 4)**	2.5 (38 + 773)	40
100,1000,25	5	7.9 (40 + 7179)	1.2 **(67 + 152)**	7.4 (23 + 1675)	97
100,1000,25	7	6.3 (145 + 7074)	1.4 (577 + 2217)	3.3 **(110 + 2530)**	71

Table XIII – Results for Random instances when $n \geq 40$ and D is odd. The designation t) indicates that we did not obtain the optimal solution after 2 days of CPU time. The designation u) indicates that the value shown on the right-hand column is an upper bound since none of the three models was able to solve the corresponding problem.

|V|,|E|,|R|	D	HopDEMCF	HopDEMCF+	Optimum
40,400,20	5	6.3 **(32 + 3921)**	3.6 (71 + 7815)	362
40,400,20	7	4.4 **(136 + 7962)**	2.1 (1052 + 27565)	317
40,400,40	5	4.1 **(185 + 52493)**	2.4 (641 + 81911)	612
40,400,40	7	4.8 (1021 + t))	3.3 (3770 + t))	527 u)
60,600,15	5	6.4 **(36 + 3386)**	3.1 (76 + 5720)	307
60,600,15	7	6.6 (193 + 50002)	4.3 **(1004 + 46240)**	280
60,600,30	5	7.7 **(142 + 98422)**	5.0 (697 + t))	585
60,600,30	7	5.9 (992 + t))	4.3 (5443 + t)	481 u)
80,800,10	5	16.9 **(5 + 188)**	9.7 (6 + 234)	46
80,800,10	7	14.7 (15 + 294)	0.0 **(89 + 2)**	34
80,800,20	5	9.4 **(17+ 2252)**	4.8 (33 + 2405)	111
80,800,20	7	8.0 (164 + 4408)	3.8 **(253 + 3049)**	78

Table XIV – Results for Euclidean instances when $n \geq 40$ and D is odd. The designation a) indicates that we did not obtain the optimal solution after 2 days of CPU time. The designation u) indicates that the value shown on the right-hand column is an upper bound since none of the three models was able to solve the corresponding problem.

6.4 Summary and Conclusions

Our computational results lead to the following conclusions:

(1) Using the models presented in the paper, we have been able to solve Steiner instances containing up to 100 nodes and 1000 edges (with $|R| \leq (1/2)|V|$ and $D \leq 8$) and spanning tree instances with up to 60 nodes and 600 edges and with $D \leq 8$.

(2) Euclidean problems appear to be more difficult than random instances.

(3) For situations when D is even, the linear programming relaxation of the model HopDMCF produces gaps of less than one percent for almost all problem instances.

(4) Situations when D is odd appear to be more difficult than those when D is even.

(4a) The linear programming gaps for the models range from a few percent to as much as 17 percent, with gaps of 5 to 8 percent being typical.

(4b) For these instances, the models HopDEMCF, HopDEMCF+ and HopDEMCF* compete for generating solutions most quickly.

7. Conclusions

We have introduced single source models for the diameter constrained minimum spanning tree and Steiner tree problems. A traditional model introduces a commodity and imposes a hop constraint for each pair of required nodes. By simultaneously selecting a central node or a central edge and using it as source for the commodities in a multicommodity flow model, our models contain only n commodities.

Our best model, which introduces a directed version of the model with hop constraints, has been able to solve a small sized instance in less then one second that the traditional model has not been able to solve after one week of computation. We have presented computational results for spanning tree and Steiner tree instances with up to 100 nodes and 1000 edges and have been able to solve a model with slightly more than 250,000 integer variables and 250,000 constraints. To the best of our knowledge, our results provide the first computational experience in solving the diameter-constrained Steiner tree problem.

We have also shown that the linear programming relaxation of the best models discussed in this paper always produce an integer optimal solution for two easily solved cases of the DMST.

Our progress in improving the algorithmic performance of solution methods for this class of problems rests upon several modeling ideas: exploiting underlying graph structure, using multicommodity flow models for network design problems, directing network design models, introducing extended (in this case hop-indexed) formulations, and using polyhedral methods (valid inequalities). As in other application contexts reported in the literature, our experience illustrates the value of embracing a multi-faceted modeling approach for solving (mixed) of integer programming models.

Acknowledgments

We are grateful to the referees for their suggestions leading to a substantially improved presentation of the results of this paper. We are particularly grateful to the referees whose comments have led us to cast our models using the auxiliary network.

References

Abdalla, A., Deo, N., and Fraceschini, R., "Parallel Heuristics for the Diameter-Constrained Minimum Spanning Tree Problem," Congressus Numeratium, (1999)

Achuthan, N. R., Caccetta, L., Caccetta, P., and Geelen, J. F. (1992), "Algorithms for the Minimum Weight Spanning Tree with Bounded Diameter Problem," in Optimization Techniques and Applications, Vol. 1, (Edited by P. H. Phua, et al.), World Scientific, pp 297-304..

Achuthan, N. R., Caccetta, L., Caccetta, P., and Geelen, J. F. (1994), "Computational Methods for the Diameter Restricted Minimum Weight Spanning Tree Problem," Australasian Journal of Combinatorics, Vol. 10, pp 51-71.

Balakrishnan, A., and Altinkemer, K. (1992), "Using a Hop-Constrained Model to Generate Alternative Communication Network Design," ORSA Journal on Computing, Vol. 4, pp. 192 - 205.

Balakrishnan, A., Magnanti, T., and Wong, R. (1989), "A Dual Ascent Procedure for Large-Scale Uncapacitated Network Design," Operations Research, Vol. 37, pp. 714-740.

Balakrishnan, A., Magnanti, T., and Mirchandani, P. (1994), "Modelling and Heuristic Worst-Case Performance Analysis of the Two-Level Network Design Problem," Management Science, Vol. 40, pp. 846-867.

Camerini, P., Galbiati, G., and Maffioli, F. (1980), "Complexity of Spanning Tree Problems: Part I," European Journal of Operational Research, Vol. 5, pp. 346-352.

Chopra, S., and Rao, M. (1994), "The Steiner Tree Problem I: Formulations, Compositions and Extensions of Facets," Mathematical Programming, Vol. 64, pp. 209-229.

Garey, M., and Johnson D. (1979), "Computers and Intractability: a Guide to the Theory of Np-Completeness," Freeman, San Francisco.

Goemans, M. (1994), "The Steiner Tree Polytope and Related Polyhedra," Mathematical Programming, Vol. 63, pp. 157-182.

Goemans, M., and Myung, Y. (1993), "A Catalog of Steiner Tree Formulations," Networks, Vol. 23, pp. 19-28.

Gouveia, L. (1995), "Using the Miller-Tucker-Zemlin Constraints to Formulate Minimal Spanning Trees with Hop Constraints," Computers and Operations Research, 22, pp. 959-970.

Gouveia, L. (1996), "Multicommodity Flow Models for Spanning Trees with Hop Constraints," European Journal of Operational Research, Vol. 95, pp. 178-190.

Gouveia, L. (1998), "Using Variable Redefinition for Computing Lower Bounds for Minimum Spanning and Steiner Trees with Hop Constraints," INFORMS Journal on Computing, Vol. 10, pp. 180-188.

Handler, G. Y. (1973), "Minimax Location of a Facility in an Undirected Tree Graph", Transportation Science, Vol. 7, pp. 287-293.

Hwang, F., Richards, D., and Winter, P. (1992), "The Steiner Tree Problem," North Holland, Amsterdam.

Magnanti, T. and Raghavan, S. (1999) Strong Formulations for Network Design Problems with Connectivity Requirements, Working paper OR 332-99, Operations Research Center, MIT, April 1999.

Magnanti, T. and Wolsey, L. (1996), "Optimal Trees" in "Network Models," Handbooks in Operations Research and Management Science, Vol. 7, pp. 503-615.

Magnanti, T. and Wong, R. (1984), "Network Design and Transportation Planning: Models and Algorithms.," Transportation Science, Vol. 18, pp. 1 - 55.

Martin, R. (1986), "A Sharp Polynomial Size Linear Programming Formulation of the Minimum Spanning Tree Problem," Working Paper, University of Chicago

Wong, R.T. (1984), "A Dual Ascent Approach for Steiner Tree Problems on a Directed Graph," Mathematical Programming, Vol. 28, pp. 271-287.

Woolston, K. and Albin, S. (1988), "The Design of Centralized Networks with Reliability and Availability Constraints," Computers and Operations Research, Vol. 15, pp. 207-217.

Appendix 1 – Comparing the Enhanced and Original Models

In this Appendix we show that the linear programming relaxation of the enhanced model Enh-HopDMCF dominates the linear programming relaxation of the model HopDEMCF+ described in Section 3.2.4. To make the proof easier to understand, in Tables XV and XVI we rewrite the models HopDEMCF+ and Enh-HopDMCF (we consider the reduced model HopDEMCF described in Section 3.2.3 augmented with the valid inequalities described in Section 3.2.4).

Table XV. The Model HopDEMCF+

$$\textit{minimize} \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{e\in E} c_e E_e$$

$$\textit{subject to} \sum_{(i,j)\in A} x_{ij} = n-2$$

$$\sum_{e\in E} E_e = 1$$

$$\sum_{i\in V}\sum_{j\in V} z_{ij}^{2k} = 1 \qquad \textit{for all } k \in V$$

$$\sum_{j\in V} z_{ij}^{h+1,k} - \sum_{j\in V} z_{ji}^{hk} = 0 \qquad \textit{for all } i,k \in V, h = 2,...,[(D-1)/2]$$

$$\sum_{j\in V} z_{jk}^{[(D-1)/2]+1,k} = 1 \qquad \textit{for all } k \in V$$

$$\sum_{h=2,...,H} z_{ij}^{hk} \le x_{ij} \qquad \textit{for all } (i,j) \in A, k \in V$$

$$\sum_{j\in V} z_{ij}^{2k} \le \sum_{e\in E(i)\setminus\{i,k\}} E_e \qquad \textit{for all } i,k \in V \textit{ and } i \ne k$$

$$z_{ii}^{2i} \le \sum_{e\in E(i)} E_e \qquad \textit{for all } i \in V$$

$$z_{ij}^{2k} \le \sum_{e\in L(i,j)\setminus\{i,k\}} E_e \qquad \textit{for all } (i,j) \in A \textit{ and } k \in V$$

$$z_{ij}^{hk} \in \{0,1\} \qquad \textit{for all } (i,j) \in A, h = 2,...,[(D-1)/2]+1, k \in V$$

$$z_{kk}^{hk} \in \{0,1\} \qquad \textit{for all } k \in V, h = 2,...,[(D-1)/2]+1$$

$$x_{ij} \in \{0,1\} \qquad \textit{for all } (i,j) \in A$$

$$E_e \in \{0,1\} \qquad \textit{for all } e \in E.$$

As in our discussion of the reduced model in Section 3.2.3, this model does not contain variables associated with arcs leaving the auxiliary root node.

To simplify the relationship between the Enh-HopDMCF formulation and the formulation HopDEMCF+, we shall define the variables of the enhanced model in the context of the original graph G = (V,A). As mentioned in Section 4, the enhanced formulation uses the following sets of arc design variables. Variables x_{0e} (e = {p,q} ∈ E) indicate whether the tree contains the arc (0,e). Variables x_{ej} (e = {p,q} ∈ E; j ∈ V\{p,q}) indicate whether the tree contains the arc (e,j). Variables x_{ij} ((i,j) ∈ A) indicate whether the tree contains the arc (i,j) and node i is not an endnode of the root edge.

Table XVI. The Model Enh-HopDMCF

$$\text{minimize} \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{e\in E} c_{0e}x_{0e} + \sum_{j\in V}\sum_{e\in E\setminus E(j)} c_{ej}x_{ej}$$

$$\text{subject to} \sum_{(i,j)\in A} x_{ij} + \sum_{e\in E} x_{0e} + \sum_{j\in V}\sum_{e\in E\setminus E(j)} x_{ej} = n-1$$

$$\sum_{e\in E} x_{0e} = 1$$

$$\sum_{e\in E} z_{0e}^{1k} = 1 \qquad \text{for all } k \in V$$

$$\sum_{j\in V\setminus\{p,q\}} z_{ej}^{2k} + z_{ee}^{2k} = z_{0e}^{1k} \qquad \text{for all } e = \{p,q\} \in E, k \in V$$

$$\sum_{j\in V} z_{ij}^{3k} = \sum_{e\in E\setminus(E(i)\cup E(k))} z_{ei}^{2k} \qquad \text{for all } i,k \in V$$

$$\sum_{j\in V} z_{ij}^{h+1,k} - \sum_{j\in V} z_{ji}^{hk} = 0 \qquad \text{for all } i,k \in V, h = 3,\ldots,[(D-1)/2]$$

$$\sum_{j\in V} z_{jk}^{[(D-1)/2]+1,k} + \sum_{e\in E(k)} z_{ee}^{[(D-1)/2]+1,k} = 1 \qquad \text{for all } k \in V$$

$$z_{ee}^{hk} = z_{ee}^{h+1,k} \qquad \text{for all } k \in V, e \in E(k),\ h = 2,\ldots,[(D-1)/2]$$

$$\sum_{h=3,\ldots,H} z_{ij}^{hk} \le x_{ij} \qquad \text{for all } (i,j) \in A, k \in V$$

$$z_{ej}^{2k} \le x_{ej} \qquad \text{for all } e = \{p,q\} \in E, k, j \in V\setminus\{p,q\}$$

$$z_{ij}^{hk} \in \{0,1\} \qquad \text{for all } (i,j) \in A, h = 3,\ldots,[(D-1)/2]+1, k \in V$$

$$z_{0e}^{1k} \in \{0,1\} \qquad \text{for all } e \in E, k \in V$$

$$z_{ej}^{2k} \in \{0,1\} \qquad \text{for all } e = \{p,q\} \in E,\ j,k \in V\setminus\{p,q\}$$

$$z_{ee}^{hk} \in \{0,1\} \qquad \text{for all } e \in E(k), k \in V, h = 2,\ldots,[(D-1)/2]+1$$

$$z_{kk}^{hk} \in \{0,1\} \qquad \text{for all } k \in V, h = 3,\ldots,[(D-1)/2]+1$$

$$x_{0e} \in \{0,1\} \qquad \text{for all } e = \{p,q\} \in E$$

$$x_{ej} \in \{0,1\} \qquad \text{for all } e = \{p,q\} \in E, j \in V\setminus\{p,q\}$$

$$x_{ij} \in \{0,1\} \qquad \text{for all } (i,j) \in A.$$

As before, we have eliminated the flow variables y from this model. Given the structure of the augmented graph, the model contains three types of hop-indexed variables ((i) variables z_{ij}^{hk} with h ≥ 3 have the same interpretation as before: they specify whether arc (i,j) is the h[th] arc in the path from the root node to node k, (ii) variables z_{ej}^{2k} (e = {p,q} ∈ E; j,k ∈ V\{p,q}) indicating whether arc (e,j) is the second arc in the path to node k and (iii) variables z_{0e}^{1k} (e = {p,q} ∈ E; k ∈ V\{p,q}) indicating whether arc (0,e) is the first arc in the path to node k or to a edge e ∈ E(k). The model also contains two types of "loop" variables: (i) variables z_{ee}^{hk} (e ∈ E(k) and h = 2,…,H) with zero cost to model situations when the path from the root node to node e contains fewer than *H* arcs (that is, $z_{ee}^{hk} = 1$ for some e ∈ E(k) and h ≤ H) and (ii) variables z_{kk}^{hk} (k ∈ V and h = 3,…,H) with zero cost to model situations when the path from the root node to node k contains fewer than *H* arcs (that is, $z_{kk}^{hk} = 1$ for some h ≤ H).

As noted before, the enhanced model for situations with D odd is a version of the D even model in a special graph. Thus, it is natural that it shares some of the properties of the D even models presented in Section 3.2.1. In particular, any feasible solution of the corresponding linear programming relaxation satisfies the linking constraints for arcs emanating from the auxiliary root node as equalities. That is, $z_{0e}^{1k} = x_{0e}$ for all e ∈ E and k ∈ V and so we can substitute for the variables z_{0e}^{1k} in terms of the variables x_{0e} in the formulation. After making these substitutions, we can remove the constraints in the third set because they become equal to the second set. The constraints $\sum_{j \in V \setminus \{p,q\}} z_{ej}^{2k} + z_{ee}^{2k} = x_{0e}$ for all e ∈ E and k ∈ V (we have substituted for the z variable on the righthand side of the original constraint using the corresponding x variable) are easier to understand if we consider the two mutual exclusive cases, e ∈ E(k) or e ∈ E\E(k). When e ∈ E(k), the variables z_{ej}^{2k} are not defined and so $z_{ee}^{2k} = x_{0e}$. When e ∈ E\E(k), the variables z_{ee}^{2k} are not defined and so $\sum_{j \in V \setminus \{p,q\}} z_{ej}^{2k} = z_{0e}$.

We can relate the edge shrinking effect used for defining the supernodes in the enhanced network with the sets L(i,j). However, we need to consider carefully the situations when $c_{ij} = c_{pj}$ for three different nodes i, j and p. When shrinking the edge {i,p}, we need to specify the arc of the original network that corresponds to arc ({i,p},j) of the expanded network. For simplicity, we may break ties by choosing arc (i,j) as the original arc if and only if i < p. This choice reflects a slightly altered definition of the sets L(i,j) as the set of potential root edges {i,p} ∈ E whose endpoint i is closer to node j than is node p. That is, L(i,j) = {{i,p} ∈ E: p ≠ j and ($c_{ij} < c_{pj}$ or ($c_{ij} = c_{pj}$ and i < p))}. The last condition in the definition of L(i,j) guarantees that an edge does not belong to two different sets L(i,j) and L(p,j) with i ≠ p. That is, if $c_{ij} = c_{pj}$ (i < p), then {i,p} belongs to L(i,j) but not to L(p,j).

For simplicity, we let (x1,z1,E1) denote a solution in the model HopDEMCF and by (x2,z2) a solution in the enhanced model Enh-HopDMCF. Table XVII depicts the relationships between variables of these two solutions.

Table XVII. Relating the Variables in the Enhanced Model and Original Model

Original		Enhanced	
$E1_e$	=	$x2_{0e}$	$e \in E$
$x1_{ij}$	=	$x2_{ij} + \sum_{e \in L(i,j)} x2_{ej}$	$(i,j) \in A$
$z1_{ij}^{2k}$	=	$\sum_{e \in L(i,j) \setminus E(k)} z2_{ej}^{2k}$	$(i,j) \in A, k \in V$
$z1_{ij}^{hk}$	=	$z2_{ij}^{hk}$	$(i,j) \in A, k \in V, h = 3,...,H$
$z1_{kk}^{2k}$	=	$\sum_{e \in E(k)} z2_{ee}^{2k}$	$k \in V$
$z1_{kk}^{hk}$	=	$\sum_{e \in E(k)} z2_{ee}^{hk} + z2_{kk}^{hk}$	$k \in V, h = 3,...,H.$

The first equality follows from the fact that a feasible solution for any of the models defined in the original graph contains the edge e as a central edge if and only if a feasible solution for the enhanced models contains the edge (0,e). The second equality follows from the fact that a feasible solution for the original model contains an arc (0,j) if and only if a feasible solution for the enhanced model contains one of the arcs (0,e) for e ∈ E(j). The third equality follows from the fact that a feasible solution for the original models contains an arc (i,j) if and only if a feasible solution for the enhanced models contains the arc (i,j) or one of the arcs (e,j) with e ∈ L(i,j). The relations for hop-indexed variables are based on similar arguments. We note, however, that the layout of the enhanced graph permits us to distinguish between hop-index variables for h = 2 and h ≥ 3 in the model defined in the original graph. When establishing the relationships for these variables with h = 2, we need to consider the commodity index k and remove edges incident to k from the range of variation of the index e (when j ≠ k). The last two relations follow from the fact that a solution in the original network contains a loop for node k in position h if and only if the corresponding solution in the enhanced network contains a loop for a node e ∈ E(k) in the same position or a loop for node k in the same position (the last alternative only holds for h ≥ 3).

Proposition 8.1 v(Enh-HopDMCF$_L$) ≥ v(HopDEMCF$_L$).

Proof: Starting with a feasible solution (x2,z2) for Enh-HopDMCF$_L$, let (x1,z1,E1) be the solution obtained by using the relations in Table XVII. We will show that the solution (x1,z1,E1) is feasible for HopEMCF$_L$.

Constraints $\sum_{e \in E} E_e = 1$ in ES: The relation $E1_e = x2_{0e}$ for all e ∈ E (from Table XVII) shows that (x2,z2) satisfies the constraint $\sum_{e \in E} x2_{0e} = 1$ in Enh-HopDMCF$_L$ if and only if (x1,z1,E1) satisfies the constraint $\sum_{e \in E} E1_e = 1$ of HopDEMCF+$_L$.

Cardinality constraints $\sum_{(i,j)\in A} x_{ij} = n-2$: By adding $x1_{ij} = x2_{ij} + \sum_{e\in L(i,j)} x2_{ej}$ (from Table XVII) for all (i,j) ∈ A, we obtain $\sum_{(i,j)\in A} x1_{ij} = \sum_{(i,j)\in A} x2_{ij} + \sum_{j\in V}\sum_{e\in E\setminus E(j)} x2_{ej}$ (the index in the last term follows from the fact that we obtain the edge set E\E(j) when we consider the union of the sets L(i,j) for all i and a fixed j). By adding $E1_e = x2_{0e}$ for all e ∈ E to the result to the previous equality we obtain $\sum_{e\in E} E1_e + \sum_{(i,j)\in A} x1_{ij} = \sum_{e\in E} x2_{0e} + \sum_{(i,j)\in A} x2_{ij} + \sum_{j\in V}\sum_{e\in E\setminus E(j)} x2_{ej}$. Using $\sum_{e\in E} E1_e = 1$ in HopDEMCF+$_L$, we can conclude that (x2,z2) satisfies the cardinality constraint $\sum_{e\in E} x2_{0e} + \sum_{(i,j)\in A} x2_{ij} + \sum_{j\in V}\sum_{e\in E\setminus E(j)} x2_{ej} = n-1$ in Enh-HopDMCF$_L$ if and only if (x1,z1,E1) satisfies the cardinality constraint $\sum_{(i,j)\in A} x1_{ij} = n-2$ of HopDEMCF+$_L$.

Constraints $\sum_{i\in V}\sum_{j\in V} z_{ij}^{2k} = 1$ for all k ∈ V: By adding $z1_{ij}^{2k} = \sum_{e\in L(i,j)\setminus E(k)} z2_{ej}^{2k}$ (from Table XVII) for all i and j and the same k, we obtain $\sum_{j\in V}\sum_{i\in V} z1_{ij}^{2k} = \sum_{j\in V}\sum_{e\in E\setminus(E(j)\cup E(k))} z2_{ej}^{2k}$. Since the model Enh-HopDMCF$_L$ does not contain the variables $z2_{ej}^{2k}$ with e ∈ E(k), we know that $\sum_{e\in E}\sum_{j\in V} z2_{ej}^{2k} = \sum_{j\in V}\sum_{e\in E\setminus(E(j)\cup E(k))} z2_{ej}^{2k}$ and the previous equality becomes $\sum_{j\in V}\sum_{i\in V} z1_{ij}^{2k} = \sum_{e\in E}\sum_{j\in V} z2_{ej}^{2k}$. Consider, now, the flow conservation constraint $\sum_{e\in E}\sum_{j\in V} z2_{ej}^{2k} + \sum_{e\in E(k)} z2_{ee}^{2k} = 1$ from Enh-HopDMCF$_L$ for the same k. Combining the two previous equalities, we obtain $\sum_{j\in V}\sum_{i\in V} z1_{ij}^{2k} + \sum_{e\in E(k)} z2_{ee}^{2k} = 1$ for the same k. Since $\sum_{e\in E(k)} z2_{ee}^{2k} = z1_{kk}^{2k}$ (from Table XVII), $\sum_{i\in V}\sum_{j\in V} z1_{ij}^{2k} = 1$ for the same k. Thus, the solution (x1,z1,E1) satisfies the constraints $\sum_{i\in V}\sum_{j\in V} z1_{ij}^{2k} = 1$ for all k ∈ V of HopDEMCF+$_L$.

Hop-indexed flow conservation constraints (h ≥ 3): Consider the equality obtained by adding the constraints $z2_{ee}^{h+1,k} = z2_{ee}^{hk}$ for all e ∈ E(k) (from Enh-HopDMCF$_L$) and fixed values of k and h ≥ 3 together with $\sum_{j\in V} z2_{ij}^{h+1,k} - \sum_{j\in V} z2_{ji}^{hk} = 0$ for a given i and the same values of k and h ≥ 3 (also from Enh-HopDMCF$_L$). By combining this equality with the sets of equalities from Table XVII, $z1_{ij}^{hk} = z2_{ij}^{hk}$ for all (i,j), k and h ≥ 3 and $z1_{kk}^{hk} = \sum_{e\in E(k)} z2_{ee}^{hk} + z2_{kk}^{hk}$ for all k ∈ V and h ≥ 3, we obtain the constraint $\sum_{j\in V} z1_{ij}^{h+1,k} - \sum_{j\in V} z1_{ji}^{hk} = 0$ for the same i, k and h ≥ 3 in HopDEMCF+$_L$. Thus, the solution (x1,z1,E1) satisfies the hop-indexed flow conservation constraints for h ≥ 3.

Hop-indexed flow conservation constraints (h = 2): By adding the equalities $z1_{ij}^{2k} = \sum_{e \in L(i,j) \setminus E(k)} z2_{ej}^{2k}$ (from Table XVII) for all i, a fixed j and a fixed k to $z1_{kk}^{hk} = \sum_{e \in E(k)} z2_{ee}^{hk}$ (also from Table XVII) for the same k and h = 2, we obtain $\sum_{i \in V} z1_{ij}^{2k} + z1_{kk}^{2k} = \sum_{e \in E \setminus (E(j) \cup E(k))} z2_{ej}^{2k} + \sum_{e \in E(k)} z2_{ee}^{2k}$ (the index in the first term of the right-hand side follows from the fact that we obtain the edge set E\(E(j)∪E(k)) when we consider the union of the sets L(i,j)\E(k) for all i and a fixed j). By combining this equality with the flow conservation constraints $\sum_{i \in V} z2_{ji}^{3k} = \sum_{e \in E \setminus (E(j) \cup E(k))} z2_{ej}^{2k}$ for the same j and k (from the model Enh-HopDMCF$_L$) and with $z2_{ee}^{3k} = z2_{ee}^{2k}$ for all e ∈ E(k) (also from the model Enh-HopDMCF$_L$) we obtain $\sum_{i \in V} z2_{ji}^{3k} + \sum_{e \in E(k)} z2_{ee}^{3k} = \sum_{i \in V} z1_{ij}^{2k} + z1_{kk}^{2k}$ for the same j and k. Finally, by using $z1_{ij}^{hk} = z2_{ij}^{hk}$ (from Table XVII) for adequate pairs (i,j), the same k and h = 3 together with to $z1_{kk}^{hk} = \sum_{e \in E(k)} z2_{ee}^{hk} + z2_{kk}^{hk}$ for the same k and h = 3, we obtain $\sum_{i \in V} z1_{ji}^{3k} = \sum_{i \in V} z1_{ij}^{2k}$ for the same j and k of the original model HopDEMCF+$_L$. Thus, the solution (x1,z1,E1) satisfies the hop-indexed flow conservation constraints for h = 2 of HopDEMCF+$_L$.

Hop-indexed flow conservation constraints (h = ((D-1)/2) + 1): By using $z1_{ij}^{hk} = z2_{ij}^{hk}$ for all (i,j), k and h = ((D-1)/2) + 1 together with $z1_{kk}^{hk} = \sum_{e \in E(k)} z2_{ee}^{hk} + z2_{kk}^{hk}$ for the same k and h we can show that (x2,z2) satisfies the constraint $\sum_{j \in V} z2_{jk}^{hk} + \sum_{e \in E(k)} z2_{ee}^{hk} = 1$ for all k ∈V and h = ((D-1)/2) + 1 in Enh-HopDMCF$_L$ if and only if (x1,z1,E1) satisfies the constraint $\sum_{j \in V} z1_{jk}^{hk} = 1$ for all k ∈ V and h = ((D-1)/2) + 1 in HopDEMCF+$_L$.

Constraints linking the z with the x variables: Fix a triple (i,j,k). By adding constraints $z2_{ej}^{2k} \leq x2_{ej}$ (from Enh-HopDMCF$_L$) for all e ∈ L(i,j)\E(k) to the constraint $\sum_{h=3,\ldots,[(D-1)/2]+1} z2_{ij}^{hk} \leq x2_{ij}$ (also from Enh-HopDMCF$_L$) for the same triple (i,j,k), we obtain the inequality $\sum_{e \in L(i,j) \setminus E(k)} z2_{ej}^{2k} + \sum_{h=3,\ldots,[(D-1)/2]+1} z2_{ij}^{hk} \leq \sum_{e \in L(i,j) \setminus E(k)} x2_{ej} + x2_{ij}$. By using the following relations from Table XVII, $x1_{ij} = x2_{ij} + \sum_{e \in L(i,j)} x2_{ej}$ for the same (i,j), $z1_{ij}^{2k} = \sum_{e \in L(i,j) \setminus E(k)} z2_{ej}^{2k}$ for the same (i,j) and the same k and $z1_{ij}^{hk} = z2_{ij}^{hk}$ for the same (i,j), k and h ≥ 3, we see that the previous inequality implies the forcing constraint $\sum_{h=2,\ldots,H} z1_{ij}^{hk} \leq x1_{ij}$ for the same triple (i,j,k) in the model HopDEMCF+$_L$. Thus, the solution (x1,z1,E1) satisfies the constraints $\sum_{h=2,\ldots,H} z1_{ij}^{hk} \leq x1_{ij}$ for all (i,j) ∈ A and k ∈ V of HopDEMCF+$_L$.

Constraints $\sum_{j\in V} z_{ij}^{2k} \leq \sum_{e\in E(i)\setminus\{i,k\}} E_e$ for all i,k ∈ V linking the z with the E variables: Consider the modified linking constraints $\sum_{j\in V} z2_{ej}^{2k} + z2_{ee}^{2k} = x2_{0e}$ for all e and k from Enh-HopDMCF$_L$. Let us replace the "=" sign by a "≤" sign (note that after this modification, we are still obtaining a valid inequality for Enh-HopDMCF$_L$). By adding these constraints for all e ∈ E(i)\{i,k} for a fixed i ≠ k, we obtain the inequality $\sum_{e\in E(i)\setminus\{i,k\}} \sum_{j\in V} z2_{ej}^{2k} \leq \sum_{e\in E(i)\setminus\{i,k\}} x2_{0e}$ for the same i and k (note that variables $z2_{ee}^{2k}$ are not defined when e ∈ E\E(k)). By using $E1_e = x2_{0e}$ for all e ∈ E(i)\{i,k}, we obtain $\sum_{e\in E(i)\setminus\{i,k\}} \sum_{j\in V} z2_{ej}^{2k} \leq \sum_{e\in E(i)\setminus\{i,k\}} E1_e$. Since $\sum_{e\in L(i,j)\setminus E(k)} \sum_{j\in V} z2_{ej}^{2k} \leq \sum_{e\in E(i)\setminus\{i,k\}} \sum_{j\in V} z2_{ej}^{2k}$ and $\sum_{j\in V} z1_{ij}^{2k} \leq \sum_{e\in E(i)\setminus\{i,k\}} \sum_{j\in V} z2_{ej}^{2k}$ (the first inequality results from the fact that L(i,j)\E(k) ⊆ E(i)\{i,k) for all i and j and the second results by adding $z1_{ij}^{2k} = \sum_{e\in L(i,j)\setminus E(k)} z2_{ej}^{2k}$ (from Table XVII) for all j and a fixed i and again noting that L(i,j)\E(k) ⊆ E(i)\{i,k)), we obtain $\sum_{j\in V} z1_{ij}^{2k} \leq \sum_{e\in E(i)\setminus\{i,k\}} E1_e$ for the same i and k with i ≠ k. Thus, the solution (x1,z1,E1) satisfies the constraints $\sum_{j\in V} z1_{ij}^{2k} \leq \sum_{e\in E(i)\setminus\{i,k\}} E1_e$ for all i,k ∈ V (with i ≠ k) of HopDEMCF+$_L$.

When i = k, we add the constraints $\sum_{j\in V} z2_{ej}^{2k} + z2_{ee}^{2k} \leq x2_{0e}$ for all e ∈ E(k) and, by noting that variables $z2_{ej}^{2k}$ are not defined when e ∈ E(k), we obtain $\sum_{e\in E(k)} z2_{ee}^{2k} \leq \sum_{e\in E(k)} x2_{0e}$ for the same k. Considering $E1_e = x2_{0e}$ (from Table XVII) for all e ∈ E(k) and $z1_{kk}^{2k} = \sum_{e\in E(k)} z2_{ee}^{2k}$ (also from Table XVII) for the same k, we obtain $z1_{kk}^{2k} \leq \sum_{e\in E(k)} E1_e$ and thus, the solution (x1,z1,E1) satisfies the constraints $z1_{kk}^{2k} \leq \sum_{e\in E(k)} E1_e$ for all k ∈ V of HopDEMCF+$_L$. [Note: Once more, we can derive these inequalities as equalities – see Section 3.2.3 – if we start with $\sum_{j\in V} z2_{ej}^{2k} + z2_{ee}^{2k} = x2_{0e}$].

Constraints $z_{ij}^{2k} \leq \sum_{e\in L(i,j)\setminus E(k)} E_e$ for all (i,j) ∈ A and k ∈ V linking the z with the E variables: We consider a weaker version of the inequalities $\sum_{j\in V} z2_{ej}^{2k} + z2_{ee}^{2k} \leq x2_{0e}$, namely the inequalities $z2_{ej}^{2k} \leq x2_{0e}$ for all e = {p,q} and j ≠ p,q. Using $E1_e = x2_{0e}$ (from Table XVII) for all e, we obtain $z2_{ej}^{2k} \leq E1_e$. By adding these inequalities for all e ∈ L(i,j)\E(k) and a fixed i ∈ V, using $z1_{ij}^{2k} = \sum_{e\in L(i,j)\setminus E(k)} z2_{ej}^{2k}$ (from Table XVII) for the same triple (i,j,k), we obtain $z1_{ij}^{2k} \leq \sum_{e\in L(i,j)\setminus E(k)} E1_e$ for the same triple (i,j,k). Thus, the solution (x1,z1,E1) satisfies the constraints $z1_{ij}^{2k} \leq \sum_{e\in L(i,j)\setminus E(k)} E1_e$ for all (i,j) ∈ A and k ∈ V of HopDEMCF+$_L$.

Variable bound inequalities: It is easy to show that the solution (x1,z1,E1) satisfies the variable bound inequalities in HopDEMCF$_L$ because the solution (x2,z2) satisfies the variable bound inequalities in Enh-HopDMCF$_L$.

We have shown that the solution (x1,z1,E1) obtained from (x2,z2) by using expressions in Table XVII is feasible for HopEMCF$_L$. We note that the definition of the costs guarantees that the two solutions have the same cost and, thus, the optimal linear programming value of a model defined in the original graph cannot exceed the optimal linear programming value of the corresponding model defined in the expanded graph, thus establishing the validity of Proposition 8.1. ♦