Upgrading Arcs to Minimize the Maximum Travel Time in a Network

Ann Melissa Campbell
Department of Management Sciences
University of Iowa
Iowa City, Iowa 52242-1994
ann-campbell@uiowa.edu,
Timothy J. Lowe
Department of Supply Chain and Information Systems
Pennsylvania State University
University Park, PA 16202
tjl12@psu.edu,
and
Li Zhang
Department of Mathematics and Computer Science
The Citadel
Charleston, South Carolina 29409
li.zhang@citadel.edu

October 13, 2005

Abstract

In transportation and telecommunication systems, the performance of the underlying network can often be improved by upgrading some of the arcs in the network. If an arc is upgraded, the travel time along this arc is reduced by a discount factor α, $0 \leq \alpha < 1$. With respect to different minimax network objectives, we define a series of problems that involve finding the best q arcs in a network to upgrade. We show that these problems are NP-hard on general graphs, but polynomially solvable on trees. Finally, we compare three heuristic solution approaches for complete graphs based on these polynomial results and find one to far outperform the others.

Keywords: dynamic programming, minimax objective, graph diameter, graph radius, arc upgrading, tree graphs, heuristics

1 Introduction

In transportation and telecommunication systems, the performance of the underlying network can often be improved by upgrading some of the arcs in the network. In such real world applications, upgrading an arc between a pair of nodes corresponds to using a rapid mode of transit, such as using a plane instead of a truck between a pair of cities in a transportation network or using a higher speed cable between a pair of servers in a telecommunication network. These upgrades decrease transportation times through the network and can be critical in achieving delivery deadlines. Because upgrading arcs is expensive, it is important to carefully evaluate the decisions of how many arcs and which arcs to upgrade.

If an arc is upgraded, the travel time along this arc is reduced by a discount factor α, $0 \leq \alpha < 1$. With respect to different network objectives, we define the following problems.

The q-upgrading arc diameter problem (Q-D) selects q upgrading arcs such that the *diameter* of the network is minimized. The diameter of a network equals the travel time on the maximum shortest path between any origin-destination (o-d) pair.

The q-upgrading arc radius problem (Q-R) selects q upgrading arcs and locates the *vertex center* such that the *radius* of the network is minimized. The vertex center of a network is a node whose maximum shortest path to the other nodes is as small as possible. This maximum shortest path is the radius of the network. The vertex center may correspond to the best vertex location for an emergency services facility where the nodes correspond to locations of customers.

These q-upgrading arc problems are important for time-sensitive or guaranteed time distribution systems, such as emergency services and express mail services. In these systems, the diameter or the radius of a network represents the best time guarantee that can be offered to all customers. To be competitive, it is important that this value be as low as possible.

To the best of our knowledge, these q-upgrading arc problems have not been studied in the literature. Related work contributed by Paik et al. [22, 23, 24, 25] addresses network upgrading problems which upgrade the vertices in a network. In their models, if a vertex v is upgraded, then the travel time on each edge incident to v reduces by a factor x, $0 \leq x < 1$. Signal flow in electronic circuits is one of the applications in which upgrading vertices are used to improve the performance of a circuit [7, 17]. Demgensky et al. [8] consider a minimum cost flow problem on a network where the per unit cost of flow on an arc can be reduced by upgrading the arc. Upgrading an arc incurs a fixed cost, so the problem is to find the optimal arcs to upgrade subject to a budget constraint and such that the total flow cost is minimum. They show that the problem is NP-hard on series-parallel networks but provide a polynomial time approximation algorithm for such networks.

Campbell et al. [3, 4] address hub arc location problems which locate upgrading arcs in a network such that the sum of the transportation cost between all o-d pairs is minimized. The endpoints of the upgrading arcs dictate the location of hubs, and the flow between all origin-destination pairs is assumed to include at least one of these hub nodes. This restriction on the structure of flow between nodes as well as the objective function makes the problems quite different than the problems addressed here.

Arc upgrading problems are also closely related to general hub location and hub covering problems. In hub location problems, the travel time on the

arc between every pair of hubs reduces by a factor α, $0 \leq \alpha \leq 1$. See [1, 5, 13, 14, 19, 20, 21] for more details and examples. In hub covering problems, the typical objective is to select the minimum number of hubs necessary so that all origin-destination pairs can be served within a given time limit. Examples of papers discussing hub covering include [31, 6].

Another problem related to Q-R is the vertex-restricted p-center problem. The p-center problem selects p vertices in a network and allocates each demand node to one of the p centers such that the maximum shortest path from any demand node to its center is minimized. This problem and a version where center locations can be on arcs as well as nodes are well studied in the literature, for example, see [11, 12, 18, 29, 30].

Section 2 establishes that the q-upgrading arc problems are NP-hard on a general graph. Section 3 demonstrates that several cases involving special graphs are polynomially solvable. Section 4 shows that variants that include a budget constraint are NP-hard even on these special graphs. Three heuristic approaches for Q-D on a complete graph are described and evaluated in Section 5, with one clearly outperforming the others. Section 6 concludes the paper and discusses future research opportunities.

2 Problems on a general graph

Consider an undirected connected graph $G = (V, E)$ with node set $V = \{v_1, ..., v_n\}$ and arc set E. The travel time on an arc between v_x and v_y (or $e \in E$) is $t_{x,y}$ (or t_e), where these values obey the triangular inequality. We assume these values are symmetric, that is, $t_{x,y} = t_{y,x}$ and that $t_{u,u} = 0$. A discount factor α ($0 \leq \alpha < 1$) is a multiplier that reduces the travel time on an arc if the arc is chosen to be one of the q upgrading arcs. The travel time $\theta_{i,j}$ between an o-d pair v_i, v_j is equal to the travel time along the shortest path between v_i and v_j in the graph. Assume $0 < q \leq |E|$ (if $q > |E|$, the solution is to select every arc in E). The *resulting graph* for a solution is the graph after the q upgrading arcs in the solution are upgraded. Note that the travel times on arcs in a resulting graph may no longer obey the triangular inequality.

To provide some perspective on the challenge of solving these problems, we first provide an integer programming formulation of Q-D. In the formulation, we denote the set of arcs adjacent to node v_j by $E(j)$. (In cases when no ambiguity results, we will often refer to node v_j as simply "j".) We make use of the following variables:

$$x_e^{s,t} = \begin{cases} 1 & \text{if arc } e \text{ is used on the path from } s \text{ to } t \\ 0 & \text{otherwise} \end{cases}$$

$$y_e = \begin{cases} 1 & \text{if arc } e \text{ is upgraded} \\ 0 & \text{otherwise} \end{cases}$$

$$b_j^{s,t} = \begin{cases} 1 & \text{if node } v_j \text{ is visited on the path from } s \text{ to } t \\ 0 & \text{otherwise} \end{cases}$$

d_e = final travel time for arc e

z = value of maximum shortest path (the objective)

We now have:

$$\min \quad z \tag{1}$$

subject to:

$$z \geq \sum_{e \in E} d_e x_e^{s,t} \quad 1 \leq s < t \leq n \tag{2}$$

$$d_e = t_e + (\alpha t_e - t_e) y_e \quad \forall e \in E \tag{3}$$

$$\sum_{e \in E(s)} x_e^{s,t} = 1 \quad 1 \leq s < t \leq n \tag{4}$$

$$\sum_{e \in E(t)} x_e^{s,t} = 1 \quad 1 \leq s < t \leq n \tag{5}$$

$$\sum_{e \in E(j)} x_e^{s,t} = 2 b_j^{s,t} \quad \forall j \in N \backslash \{s,t\}, 1 \leq s < t \leq n \tag{6}$$

$$\sum_{e \in E} y_e = q \tag{7}$$

$$y_e \in \{0,1\} \quad \forall e \in E \tag{8}$$

$$x_e^{s,t} \in \{0,1\} \quad 1 \leq s < t \leq n, \forall e \in E \tag{9}$$

$$b_j^{s,t} \in \{0,1\} \quad \forall j \in N \backslash \{s,t\}, 1 \leq s < t \leq n \tag{10}$$

The objective function (1) and constraint (2) minimize the maximum shortest path between any pair of nodes s, t in G. Constraint (3) determines the "final" travel time for arc e given the upgrade decision for the arc. Constraint (4) enforces that there is exactly one arc leaving s on the path from s to t, and constraint (5) enforces that there is exactly one arc entering t on the path

from s to t. Constraint (6) ensures that the path from s to t is connected. Constraint (7) restricts the number of upgrading arcs to q.

Both Q-D and Q-R are in class NP. For Q-D, if we are given the locations of the q upgrading arcs, then we can find the shortest path between every o-d pair in the resulting graph using Floyd's all pairs shortest paths algorithm [9] in $O(n^3)$. The objective value (i.e., the diameter of the resulting graph) of this solution is found in $O(n^2)$ time by finding the maximum of these shortest path values. For Q-R, if we are given the locations of the q upgrading arcs, then we can evaluate the objective value with each node as the center and compare the results.

Next, we will prove that both Q-D and Q-R on a general graph are NP-hard. We will first create a decision version of Q-D, which we will refer to as *DQ-D*, and define it as follows:
Instance: A graph $G = (V, E)$, travel time $t_e \in Z^+$ for each $e \in E$, a positive integer q, a positive integer K and a discount factor α $(0 \leq \alpha < 1)$.
Question: Is there a subset $F \subseteq E$ such that $|F| = q$ and if we let $t_e = \alpha t_e \, \forall e \in F$, then G has diameter K or less?

We will prove that DQ-D is NP-complete by a reduction from a special case of the weighted diameter problem (problem GT65 in [10]) which is known to be NP-complete.

The special case of weighted diameter problem (SWD) is defined as follows:
Instance: Graph $G' = (V, E)$, collection C of 0's and 1's where $|C| = |E|$, and a positive integer K'.
Question: Is there a one-to-one function $f : E \rightarrow C$ such that, if $f(e)$ is taken as the length of edge e, then G' has diameter K' or less?

Lemma 1 *DQ-D is NP-complete even if* $\alpha = 0$ *and* $t_e = 1, \forall e \in E$.

Proof: For an instance of SWD, we reduce SWD to DQ-D as follows.
Let $G = G'$, $q =$ the number of $0's$ in C, $K = K'$, $t_e = 1, \forall e \in E$ in G, and $\alpha = 0$. The instance of SWD will have a yes answer if and only if the constructed DQ-D has a yes answer. □

Since DQ-D is NP-complete, Q-D is NP-hard on a general graph even if $\alpha = 0$ and $t_e = 1, \forall e \in E$.

We can replace "diameter" by "radius" in DQ-D to create a decision version of Q-R which we will refer to as *DQ-R*. Since the variant of SWD in which "diameter" is replaced by "radius" for general graphs is also NP-complete [10], DQ-R is NP-complete even if $\alpha = 0$ and $t_e = 1, \forall e \in E$, by a similar proof as for Lemma 1. It follows that Q-R also is NP-hard.

Note that a node-specific version of Q-R which selects q upgrading arcs such that the maximum shortest path from a given node v to the other nodes is minimized is also NP-hard since Q-R can be polynomially transformed to this problem. Denoting the node-specific version by Q-R-N, we observe that we can solve a Q-R problem by solving a total of n Q-R-N problems.

By the above arguments, we have the following theorem:

Theorem 1 *Q-D, Q-R, and Q-R-N are NP-hard.*

We remark that a similar complexity argument can be made based on the bounded-cardinality-minimum-diameter edge addition problem (BCMD) which has been shown to be NP-hard ([16], [27]). In this problem, all arcs of $G = (V, E)$ are of unit length, and the problem is to add no more than q additional arcs (E') so that the diameter of the resulting graph $G = (V, E \cup E')$ is minimized.

3 Polynomially solvable cases

In Section 2, we demonstrated that Q-D, Q-R-N, and Q-R are NP-hard. In this section, we will show that these problems are polynomially solvable on tree graphs. Telecommunication and distribution networks are often sparse due to the high cost of establishing connections between nodes. Thus, the study of simpler graph structures, such as trees, is applicable and useful.

A key difference between tree graphs and general cyclic graphs that will be helpful in establishing our results is that there is one and only one path between every o-d pair on a tree graph. Also, to minimize the diameter of a tree graph, it is sufficient to minimize the maximum shortest path between every pair of leaf nodes. A leaf node of a tree graph is a node whose degree is equal to 1.

Before describing our results for general tree graphs, we first discuss some of the simpler graph structures where we have found lower order polynomial algorithms.

3.1 Path graph and star tree graph

If T is a path graph or a star tree graph, Q-D can be easily solved by a greedy algorithm. Furthermore, the same greedy algorithm can be used to solve Q-R if T is a star tree, but not if T is a path graph. A path graph is a special tree graph where each node has degree one or two. A star tree graph is a special tree graph where the (vertex) center of the star tree has degree $n-1$ and all of the other nodes have degree 1.

Theorem 2 *Q-D on a path or star tree graph can be solved optimally in* $O(n)$ *time.*

Proof: In a path graph, there are clearly only two leaf nodes. The maximum shortest path between any o-d pair is the one between the two leaf nodes since it travels through all of the arcs. In a star tree graph, the maximum shortest path between any o-d pair consists of the largest and the second largest arcs of the star tree. Greedily selecting the q largest arcs as upgrading arcs clearly minimizes the diameter of both path and star tree graphs.

To solve Q-D on a path or star tree graph, we must identify the q largest elements of the set of $n-1$ arc lengths. This requires $O(n)$ time [2]. □

For a star tree, the same $O(n)$ algorithm solves Q-R since the best vertex center is always the center node of the graph. For a path graph, greedy is

not as successful. Consider the following graph:

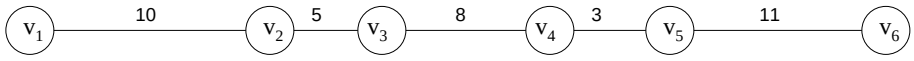

Figure 1: An example of Q-R on a path graph where $\alpha = 0.1$ and $q = 1$

In this example, if $q = 1$, a greedy algorithm would choose to discount the arc from v_5 to v_6 since 11 is the largest individual travel time. If this arc is discounted and $\alpha = .1$, the best center is at v_3. The distance from v_1 to v_3 is 15, and the distance from v_3 to v_6 becomes 8+3+1.1 = 12.1. Thus, if a greedy approach is used, the radius is 15. If instead the arc from v_1 to v_2 is discounted ($10 < 11$), the best vertex center is now at v_4. The distance to the left of v_4 is now 1 + 5+8=14, and the distance to the right is also 3+11=14. Even though the second solution has a larger diameter, the midpoint of the diameter is precisely on a vertex which creates the savings.

Note that for more complicated tree graphs, the greedy selection of arcs will not always find the optimal solution to Q-D or Q-R. We illustrate this for Q-D with the graph in Figure 2.

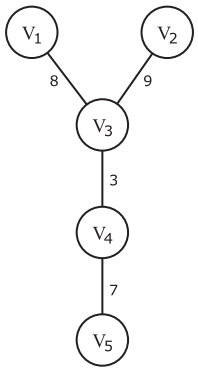

Figure 2: An example of Q-D on a tree graph where $\alpha = 0.1$

Suppose $\alpha = 0.1$. When $q = 1$, the optimal solution to Q-D on the tree graph in Figure 2 is to upgrade $\{(v_4, v_5)\}$, and the optimal objective value is $\theta_{1,2} = 17$. When $q = 2$, the optimal solution is to upgrade $\{(v_1, v_3), (v_2, v_3)\}$ and the optimal objective value is $\theta_{2,5} = 10.9$. From this example, we can see that it is not always the case that selecting the largest arc on a tree graph is optimal (for example, when $q = 1$), and it is not always the case that the optimal solution for a smaller value of q is included in the optimal solution for a larger value of q. This example also shows that the greedy selection of arcs will not always find the optimal solution to Q-D on a tree graph. In [32], we discuss several graph structures, such as the extended star tree graph, where the Q-D can be solved in low order polynomial time even without a greedy approach.

3.2 General tree

We now present an algorithm for solving Q-D, Q-R-N, and Q-R on a general tree, T. We arbitrarily select some node v_r of T and declare it to be the root of T. To simplify our presentation, we will transform T into an equivalent binary tree. A binary tree is a rooted tree where each non-leaf node has exactly two children. Tamir [28] shows how to make the transformation using at most $n - 3$ additional nodes and $n - 3$ additional arcs. The added arcs needed to construct the binary tree all have a length of zero. Solving

each of these problems on the the original tree is equivalent to solving it on the transformed binary tree since distances between nodes are preserved in the transformation, and zero length edges would not be upgraded in an optimal solution. Since the transformed binary tree has at most $2n-3$ nodes, the complexity analysis of our algorithms will not be affected by the transformation. Thus, henceforth we will assume T is a binary tree, rooted at node v_r. Furthermore, for each node v_j, we denote its children by $v_{j(1)}$ and $v_{j(2)}$ and the set consisting of v_j and its descendants by V_j. Also, let T^j be the subtree of T spanning V_j. We also assume $T \backslash T^j$ includes node v_j so that $T^j \cap \{T \backslash T^j\} = v_j$. In Figure 2, if T is rooted at v_1, then T^{v_4} is $\{v_4, v_5\}$ while $T \backslash T^{v_4}$ is $\{v_1, v_2, v_3, v_4\}$ along with the edges of T spanning these nodes. Finally, if $\mathcal{T}$ is a subtree of T, let $E(\mathcal{T})$ be the set of arcs of $\mathcal{T}$.

Our algorithm is a dynamic programming approach. The algorithm finds the best arcs to discount for each T^j and each $T \backslash T^j$ and builds solutions to Q-D and Q-R based on these subproblems. In the first pass through T, we compute a set of $F_j(k)$ values for each node v_j. $F_j(k)$ is the distance from the farthest node in T^j to v_j given that exactly k arcs in T^j are optimally upgraded to minimize this maximum distance and where $0 \leq k \leq \min\{q, \mid E(T^j) \mid\}$. Thus, $F_j(k)$ solves $\min\{\max\{\theta_{i,j} : v_i \in V_j\} : k \text{ arcs of } T^j \text{ are upgraded}\}$. To compute all $F_j(k)$, first set

$$F_j(0) = 0 \tag{11}$$

for all leaf nodes of T. Then, working from the leaf nodes of T toward v_r, we have for each non-leaf node v_j:

$$F_j(0) = \max\{t_{j,j(1)} + F_{j(1)}(0), t_{j,j(2)} + F_{j(2)}(0)\}. \tag{12}$$

$F_j(1)$ is the minimum of the following values:

$$\max\{\alpha t_{j,j(1)} + F_{j(1)}(0), t_{j,j(2)} + F_{j(2)}(0)\} \tag{13}$$
$$\max\{t_{j,j(1)} + F_{j(1)}(0), \alpha t_{j,j(2)} + F_{j(2)}(0)\} \tag{14}$$
$$\max\{t_{j,j(1)} + F_{j(1)}(1), t_{j,j(2)} + F_{j(2)}(0)\} \tag{15}$$
$$\max\{t_{j,j(1)} + F_{j(1)}(0), t_{j,j(2)} + F_{j(2)}(1)\}. \tag{16}$$

If $j(1)$ is a leaf node, then equation (15) should be ignored. Likewise, if $j(2)$ is a leaf node, then equation (16) should be ignored. Finally, for $k \geq 2$, $F_j(k)$

is the minimum of the following values:

$$\min_{\substack{k_1\le|E(T^{j(1)})|\\ k_2\le|E(T^{j(2)})|\\ k_1+k_2=k}} \max\{t_{j,j(1)}+F_{j(1)}(k_1), t_{j,j(2)}+F_{j(2)}(k_2)\} \tag{17}$$

$$\min_{\substack{k_1\le|E(T^{j(1)})|\\ k_2\le|E(T^{j(2)})|\\ k_1+k_2=k-1}} \max\{t_{j,j(1)}+F_{j(1)}(k_1), \alpha t_{j,j(2)}+F_{j(2)}(k_2)\} \tag{18}$$

$$\min_{\substack{k_1\le|E(T^{j(1)})|\\ k_2\le|E(T^{j(2)})|\\ k_1+k_2=k-1}} \max\{\alpha t_{j,j(1)}+F_{j(1)}(k_1), t_{j,j(2)}+F_{j(2)}(k_2)\} \tag{19}$$

$$\min_{\substack{k_1\le|E(T^{j(1)})|\\ k_2\le|E(T^{j(2)})|\\ k_1+k_2=k-2}} \max\{\alpha t_{j,j(1)}+F_{j(1)}(k_1), \alpha t_{j,j(2)}+F_{j(2)}(k_2)\}. \tag{20}$$

Note that for each node v_j, the effort to compute $F_j(k)$ for all relevant k is $O(q^2)$. It would appear, then, that the total effort for all nodes of T would be $O(nq^2)$. Using the complexity analysis found in Tamir [28], however, it can be shown that the effort is actually only $O(nq)$. We will briefly outline the argument detailed in [28]. First, note that the effort to compute all $F_j(k)$ values for node v_j is $O(\min\{|E(T^{j(1)})|, q\} \cdot \min\{|E(T^{j(2)})|, q\})$. In Tamir's problem, for a fixed v_j the search is over nodes of subtrees and not edges of subtrees as is the case here. Since $|E(T^{v_j})|+1 = |V_j|$ for a tree, his approach with obvious modifications will hold for our problem. Tamir partitions the node set of T into two subsets. A node is called *rich* if it is not a leaf node and both of its children $v_{j(1)}$ and $v_{j(2)}$ satisfy $|V_{j(l)}| \le \frac{q}{2},\ l = 1, 2$. (Thus, we can define a node as rich if $|E(T^{v_j})|+1 \le \frac{q}{2}$.) If a node is not rich, it is *poor*. Tamir shows that the number of rich nodes in T is bounded above by $2\frac{n}{q}$. Given this, the total effort to compute $F_j(k)$ values for all of the rich nodes of T is $O(nq)$. For a fixed node v_j, he inductively computes a bound on the total computational effort required by all of the poor nodes in V_j. Using the obvious modifications for our problem, we can show that the total effort to compute $F_j(k)$ values for all of the poor nodes in V_r (all of the poor nodes of T) is also $O(nq)$. Thus, the total effort to compute all $F_j(k)$ of T is $O(nq)$.

Once all $F_j(\cdot)$ values are available, a "backward" pass finds a set of $B_j(k)$ values for each node v_j. $B_j(k)$ is the distance between the farthest node in $T\backslash T^j$ to v_j given that exactly k arcs of $T\backslash T^j$ are optimally upgraded to

minimize the maximum distance and where $0 \leq k \leq \min\{q, \mid E(T \backslash T^j) \mid\}$. The backward pass proceeds from v_r toward the leaves of T. First, set $B_r(0) = 0$. Then for $v_{j(1)}$, a child of v_j:

$$B_{j(1)}(0) = t_{j,j(1)} + \max\{B_j(0), t_{j,j(2)} + F_{j(2)}(0)\} \tag{21}$$

$B_{j(1)}(1)$ is the minimum of the following values:

$$\alpha t_{j,j(1)} + \max\{B_j(0), t_{j,j(2)} + F_{j(2)}(0)\} \tag{22}$$
$$t_{j,j(1)} + \max\{B_j(1), t_{j,j(2)} + F_{j(2)}(0)\} \tag{23}$$
$$t_{j,j(1)} + \max\{B_j(0), \alpha t_{j,j(2)} + F_{j(2)}(0)\} \tag{24}$$
$$t_{j,j(1)} + \max\{B_j(0), t_{j,j(2)} + F_{j(2)}(1)\}. \tag{25}$$

If $j = r$, then equation (23) should be ignored. Likewise, if $j(2)$ is a leaf node, then equation (25) should be ignored. Then, for $k \geq 2$, $B_{j(1)}(k)$ is the minimum of the following values:

$$\min_{\substack{k_1 \leq |E(T \backslash T^j)| \\ k_2 \leq |E(T^{j(2)})| \\ k_1 + k_2 = k-1}} \alpha t_{j,j(1)} + \max\{B_j(k_1), t_{j,j(2)} + F_{j(2)}(k_2)\} \tag{26}$$

$$\min_{\substack{k_1 \leq |E(T \backslash T^j)| \\ k_2 \leq |E(T^{j(2)})| \\ k_1 + k_2 = k-2}} \alpha t_{j,j(1)} + \max\{B_j(k_1), \alpha t_{j,j(2)} + F_{j(2)}(k_2)\} \tag{27}$$

$$\min_{\substack{k_1 \leq |E(T \backslash T^j)| \\ k_2 \leq |E(T^{j(2)})| \\ k_1 + k_2 = k-1}} t_{j,j(1)} + \max\{B_j(k_1), \alpha t_{j,j(2)} + F_{j(2)}(k_2)\} \tag{28}$$

$$\min_{\substack{k_1 \leq |E(T \backslash T^j)| \\ k_2 \leq |E(T^{j(2)})| \\ k_1 + k_2 = k}} t_{j,j(1)} + \max\{B_j(k_1), t_{j,j(2)} + F_{j(2)}(k_2)\} \tag{29}$$

To compute all $B_{j(2)}(k)$ values requires obvious interchanges of indices $j(1)$ and $j(2)$ in the above expressions. As with the $F_j(k)$ values, we can use the distinction between rich and poor nodes to find all $B_j(k)$ values in $O(nq)$ time. Once all $F_j(k)$ and $B_j(k)$ values are computed, it is easy to see that Q-R-N is solved for a given j via

$$\min_{\substack{k_1 + k_2 = q \\ k_1 \leq |E(T^j)| \\ k_2 \leq |E(T \backslash T^j)|}} \max\{F_j(k_1), B_j(k_2)\} \tag{30}$$

and Q-R is solved via

$$\min_j\{ \min_{\substack{k_1+k_2=q \\ k_1\leq|E(T^j)| \\ k_2\leq|E(T\setminus T^j)|}} \max\{F_j(k_1), B_j(k_2)\}\}. \tag{31}$$

The overall complexity for solving Q-R-N and Q-R is $O(nq)$.

We will now show that Q-D can also be solved using these $F_j(k)$ and $B_j(k)$ values in $O(nq)$. To do this, we make use of the following lemmas.

Lemma 2 *For any arc (v_i, v_l) with $v_l \in T\setminus T^{v_i}$ and k_1, let $\tilde{t_{i,l}}$ be the length of (v_i, v_l) after upgrading the arcs specified by the solutions to $F_i(k_1)$ and $B_i(q-k_1)$. If*

$$0 \leq B_i(q-k_1) - F_i(k_1) \leq 2\tilde{t_{i,l}}, \tag{32}$$

the length of the longest path in the resulting tree is $F_i(k_1) + B_i(q-k_1)$.

Proof: Let $v_{\bar{x}} \in T^{v_i}$ satisfy $\theta_{\bar{x},i} = F_i(k_1)$ and $v_{\bar{y}} \in T\setminus T^{v_i}$ satisfy $\theta_{\bar{y},i} = B_i(q-k_1)$. Relation (32) implies that the midpoint c of the path between $v_{\bar{x}}$ and $v_{\bar{y}}$ is on arc (v_i, v_l) , and c is in $T\setminus T^{v_i}$. (We note that c is possibly a point on the interior of some edge, but we will denote by $\theta_{i,c}$ the distance between c and some node v_i of T.) Letting $\bar{M} = F_i(k_1) + B_i(q-k_1)$ be the length of this path, we have that $\theta_{\bar{x},c} = \theta_{\bar{y},c} = \frac{\bar{M}}{2}$.

Consider arbitrary nodes v_a and v_b in T. If v_a and v_b are such that $v_a \in T^{v_i}$ and $v_b \in T\setminus T^{v_i}$, then $\theta_{a,i} \leq \theta_{\bar{x},i}$ and $\theta_{b,i} \leq \theta_{\bar{y},i}$ so $\theta_{a,b} \leq \theta_{\bar{x},i} + \theta_{\bar{y},i} = \bar{M}$. Now, suppose that v_a and v_b are both in T^{v_i}. We then have that $\theta_{a,c} = \theta_{a,i} + \theta_{i,c} \leq \theta_{\bar{x},i} + \theta_{i,c} = \theta_{\bar{x},c} = \frac{\bar{M}}{2}$. Similarly, $\theta_{b,c} \leq \theta_{\bar{x},c} = \frac{\bar{M}}{2}$. But then, $\theta_{a,b} \leq 2(\frac{\bar{M}}{2}) = \bar{M}$. A similar result follows if a and b are both in $T\setminus T^{v_i}$. Thus we have shown that for arbitrary a and b, $\theta_{a,b} \leq \bar{M} = \theta_{\bar{x},\bar{y}}$.□

Lemma 3 *There exists at least one arc where (32) holds.*

Proof: Let A^* denote the set of upgraded arcs in an optimal solution to Q-D. Let v_{x^*} and v_{y^*} denote the endpoints of the longest path in the resulting

tree, and let (v_i, v_l) denote an arc containing the midpoint c of this path where v_l is in $T \backslash T^{v_i}$. (Note that c may be at v_i or at v_l or in the interior of (v_i, v_l).) Without loss of generality, suppose that the root v_r and v_{y^*} are both in $T \backslash T^{v_i}$ and $v_{x^*} \in T^{v_i}$. Also, let $A_1^* = A^* \cap T^{v_i}$ and $A_2^* = A^* \cap T \backslash T^{v_i}$. Thus $A_1^*(A_2^*)$ is the subset of the optimal upgraded arcs in T^{v_i} (in $T \backslash T^{v_i}$). Finally, let $k_1^* = |A_1^*|$ so that $|A_2^*| = q - k_1^*$.

It now follows that A_1^* solves $F_i(k_1^*)$ and A_2^* solves $B_i(q - k_1^*)$. To see this, note that if, for example, A_1^* does not solve $F_i(k_1^*)$, then there is an improving allocation of k_1^* upgrades in T^{v_i}, in which case the upgrade allocation A^* is not optimal for Q-D. Furthermore, since $c \in (v_i, v_l)$, (32) holds. $\square$

Based on Lemmas 2 and 3, a method for solving Q-D is now apparent. Let $\bar{N}$ be the subset of nodes where for each $v_i \in \bar{N}$, there is an adjacent node v_l and at least one $k_1(i)$ value where (32) holds. For each $v_i \in \bar{N}$, we let $\bar{K}_i$ represent the set of $k_1(i)$ values where (32) holds. The solution to Q-D is found via

$$\min_{v_i \in \bar{N}} \{ \min_{k_1(i) \in \bar{K}_i} \{F_i(k_1(i)) + B_i(q - k_1(i))\}\}. \tag{33}$$

As we have already established, computing all $F_i(k)$ and $B_i(k)$ values for all nodes v_i of T is $O(nq)$. For a given node v_i, identifying whether or not there are values of k such that $0 \leq B_i(q-k) - F_i(k)$ and $B_i(q-k) - F_i(k) \leq 2\tilde{t_{i,l}}$, i.e. where (32) holds, takes $O(q)$ effort. Since $O(n)$ nodes are checked for this condition, the total effort to identify $\bar{N}$ and $\bar{K}_i$ for each $v_i \in \bar{N}$ is $O(nq)$. Searching through these values in (33) is also $O(nq)$ in the worst case, so the overall complexity for solving Q-D is $O(nq)$.

It is important to note that the algorithm for the general tree is not dependent on α values being the same for each arc. It is straightforward to modify all of the proposed polynomial algorithms for heterogeneous α values without increasing computational complexity.

We conclude this section by introducing a new problem related to Q-R, which we will call the q-upgrading arc *absolute* radius problem (Q-AR). The problem is to select q upgrading arcs and to locate the absolute center of the resulting graph such that the absolute radius is minimized. In Q-AR, unlike Q-R, the absolute center can be a node or in the interior of an arc. The following lemma shows that Q-AR can be solved on a tree by solving Q-D.

Lemma 4 *Given an optimal set of q upgrading arcs that solves Q-D on a tree graph T, let v_{x^*} and v_{y^*} be the endpoints of the longest path in the resulting tree. The same set of upgrading arcs solves Q-AR where the absolute center c is the midpoint of the path between v_{x^*} and v_{y^*}.*

Proof: Let A^* be the set of arcs upgraded in the optimal solution to Q-D. With these arcs upgraded, note that with c defined as in the lemma, the radius of the resulting tree is $R^* = \theta_{x^*,y^*}/2$ [11]. Suppose that A^* does not solve Q-AR. Thus, there exists a set $\tilde{A}$ of q upgrading arcs and a center c' where the optimal radius has value $\tilde{R} < R^*$. For any two v_a and v_b in this resulting tree (with distances denoted by $\tilde{\theta}$ on the resulting tree), we have $\tilde{\theta_{a,b}} \leq \tilde{\theta_{a,c'}} + \tilde{\theta_{b,c'}} \leq 2\tilde{R} < 2R^* = \theta_{x^*,y^*}$. This contradicts the assumption that A^* solves Q-D. □

Theorem 3 *Q-R-N, Q-R, Q-D, and Q-AR can be solved optimally on a tree in $O(nq)$ time.*

4 Variants with a budget constraint

The budget constrained q-upgrading arc diameter problem *BC-D* is variant of Q-D and is defined as follows:
Instance: A graph $G = (V, E)$, travel time $t_e \in Z^+$ and cost $c(e) \in Z^+$ for each $e \in E$, a budget $B \in Z^+$, a positive integer K, and a discount factor α (where $0 \leq \alpha < 1$).
Question: Is there a subset $F \subseteq E$ such that $\sum_{e \in F} c(e) \leq B$ and if we let $t_e = \alpha t_e, \forall e \in F$, then G has diameter K or less?

Before we analyze the complexity of BC-D, we make the following observations: (1) Q-D is a simpler case of BC-D where $B = q$ and $c(e) = 1$ for each $e \in E$. (2) BC-D is in class NP since we can evaluate a given solution for BC-D in polynomial time.

Since Q-D is NP-hard on a general graph by Theorem 1, BC-D is NP-hard on a general graph. We will prove that BC-D is NP-hard even if G is a

path graph by a reduction from the knapsack problem which is known to be NP-complete (problem MP10 in [10]).

BC-D on a path graph G can be rephrased as follows:
Instance: A path graph $G = (V, E)$, travel time $t_e \in Z^+$ and cost $c(e) \in Z^+$ for each $e \in E$, a budget $B \in Z^+$, a positive integer K, and a discount factor α (where $0 \leq \alpha < 1$).
Question: Is there a subset $F \subseteq E$ such that $\sum_{e \in F} c(e) \leq B$ and such that $\sum_{e \in E} t_e - (1 - \alpha) \sum_{e \in F} t_e \leq K$?

Knapsack is defined as follows:
Instance: Finite set U, size $s(u) \in Z^+$ and value $v(u) \in Z^+$ for each $u \in U$, and positive integers B' and K'.
Question: Is there a subset $U' \subseteq U$ such that $\sum_{u \in U'} s(u) \leq B'$ and such that $\sum_{u \in U'} v(u) \geq K'$?

Theorem 4 *BC-D is NP-hard even if G is a path graph.*

Proof: The theorem is proved by a reduction from knapsack.

We can reduce an instance of knapsack to an instance of BC-D as follows: Create a path graph with $n = |U| + 1$ nodes and $|E| = |U|$ arcs where each arc $e \in E$ corresponds to an element $u \in U$. For each $e \in E$ and its corresponding $u \in U$, let $c(e) = s(u)$ and $t_e = v(u)$. Let $B = B'$ and $K = \sum_{e \in E} t_e - (1 - \alpha)K'$.

If the constructed BC-D above has a yes answer, then there is a subset F such that $\sum_{e \in F} c(e) \leq B$ and such that $\sum_{e \in E} t_e - (1 - \alpha) \sum_{e \in F} t_e \leq K$. We can use the solution to BC-D to construct a solution to the instance of knapsack by letting $U' = F$. By definition, $\sum_{u \in U'} s(u) \leq B'$ and $\sum_{u \in U} v(u) - (1 - \alpha) \sum_{u \in U'} v(u) \leq K = \sum_{u \in U} v(u) - (1 - \alpha)K'$, i.e., $\sum_{u \in U'} v(u) \geq K'$. Thus, the instance of knapsack has a yes answer. Further, the instance of knapsack will have a yes answer if and only if the constructed BC-D above has a yes answer. Since knapsack is NP-complete, BC-D is NP-hard even if G is a path. □

5 Heuristic algorithms

In this section, we briefly describe three heuristic algorithms for Q-D on complete graphs and summarize the computational results. Due to the difficulty of solving Q-D by exact methods, we explore the use of simpler graph structures to approximate the full problem and allow for quick solutions. A detailed discussion of heuristics for complete and general graphs, bounding results, improvement procedures, and more extensive tests will be found in [33]. Some of the key results are summarized here to demonstrate the usefulness of algorithms based on the polynomially solvable cases discussed in Section 3 and because the results provide insight into the characteristics of the arcs that should be discounted on complete graphs.

All of the heuristic algorithms which we will discuss in this section initially select the best q upgrading arcs from a spanning tree T of G. A *spanning tree* for a graph G is a tree graph extracted from G which connects all the n nodes on G. Finding the best q arcs to upgrade on a spanning tree of a graph may provide a reasonably good approximate solution to Q-D, and these solutions are polynomial to obtain.

The heuristic algorithms are based on selecting a spanning tree in three different ways and then solving Q-D on the resulting simpler graph structure. Since each of the approaches involves choosing the best q arcs from a subgraph of G, the objective value of the resulting solution will be an upper bound on the optimal objective value. The first approach involves finding a minimum spanning tree of G. A minimum spanning tree T on G is a spanning tree such that the total cost of the arcs (i.e., $\sum_{e \in T} t_e$) of T is minimized. There are many algorithms which find a minimum spanning tree on a graph G, for example, Kruskal's algorithm [15] or Prim's algorithm [26]. The second approach starts from a maximum spanning tree. A maximum spanning tree is a spanning tree T on G such that the total cost of the arcs (i.e., $\sum_{e \in T} t_e$) of T is maximized. The maximum spanning tree can be found by slightly modifying a minimum spanning tree algorithm. The approach outlined in Section 3.2 can be used to identify the q arcs to discount on the minimum and maximum spanning trees. The objective value of each solution can be found by applying Floyd's all pairs shortest paths algorithm to G', where G' is the graph G after the selected arcs are upgraded.

The third approach starts from star trees, so this method is applicable only on complete graphs, where the previous two approaches do not have this restriction. Here n star trees, $T_1, T_2, \ldots, T_n$, are considered, where T_i is centered at node $v_i \in V$. For each constructed star tree T, the q largest arcs of T are selected greedily, which is optimal by Theorem 2. The objective value of the solution for each constructed star tree is found by applying Floyd's all pairs shortest paths algorithm on the resulting graph G'. The q arcs yielding the lowest objective value among the constructed star trees are selected. Note that even though this third approach considers n different trees, where the first two approaches consider only one, the complexity to solve the problem on a star tree is much lower than on a general tree.

We can improve the solutions obtained by the above algorithms by a simple local search method which we will refer to as the Interchange Method. For each arc not in the optimal solution for the spanning tree, we determine if an improvement in the objective value will occur if this arc is added and an arc from the current solution is dropped. If an improvement is possible, the undiscounted arc is "swapped" with the one in the current solution that creates the biggest savings. The Interchange Method terminates after each arc not in the original set of q arcs has been considered.

We next compare the effectiveness of the three heuristic algorithms for Q-D. The results presented here are based on the CAB (Civil Aeronautics Board) data set which provides a complete graph. The CAB data set introduced by O'Kelly [20] has been used frequently in the literature to test IP formulations and algorithms for hub location problems. The data set is based on airline passenger flow between 25 US cities. All of the experiments required less than 0.5 seconds of CPU time.

In the first results we present here, we experiment with different values of n and q to evaluate how the three heuristics perform in each scenario. For a given value of n, an n-node subgraph of the 25 node graph is used. The points on the x-axis in Figure 3 correspond to different combinations of n and q, where the y-axis corresponds to the gap relative to the optimal solution. Due to their small size, optimal solutions can be found for these problems using enumeration. The graph in Figure 3 highlights the dramatically better performance using the star tree method, as opposed to the minimum or maximum spanning tree methods. The objective values based on using the

star tree method are lower on all of the scenarios tested.

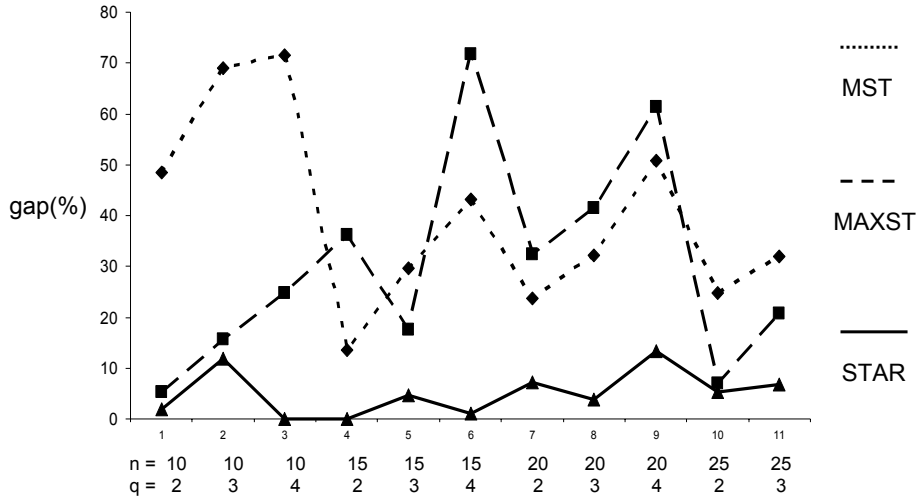

Figure 3: Test problems on CAB with α =0.5

After the Interchange Method is applied to these initial solutions, the objective values dramatically improve for the minimum and maximum spanning tree approaches, but the objective values based initially on star trees are almost always still better, as indicated in Figure 4. For the minimum and maximum spanning tree methods, we found that the improvement method either entirely or almost entirely replaces the q arcs chosen for these initial spanning trees. This seems to indicate that the arcs chosen in defining the minimum spanning tree are not the arcs creating the shortest paths of maximum length in the final graphs. Similarly, the arcs used in defining the maximum spanning trees are not likely to be used in the shortest paths between origin-destination pairs, because better, lower travel time options exist. Only with star trees are the initial q arcs more likely to closely resemble the final set of q arcs.

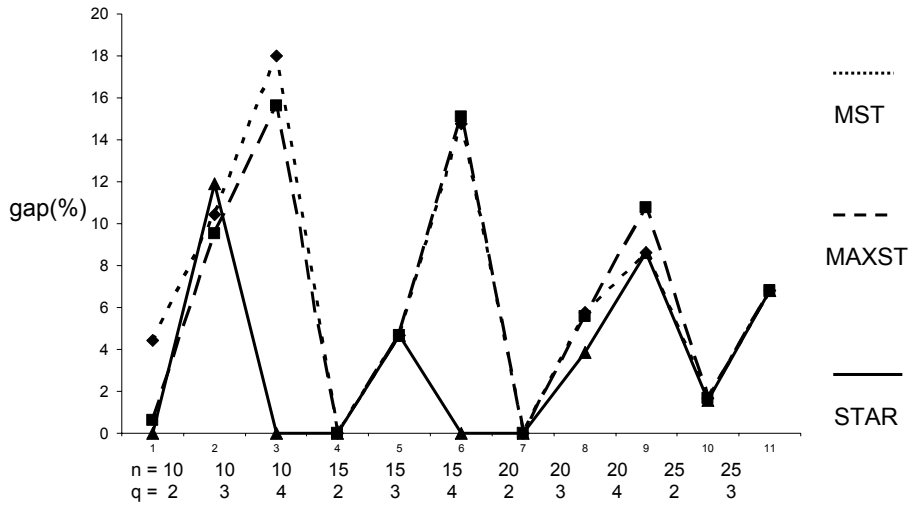

Figure 4: Test problems on CAB with α =0.5 with Improvement

6 Conclusions and Future Work

This paper introduced the q-upgrading arc problems Q-D, Q-R-N, Q-R, and Q-AR. We demonstrated that Q-D, Q-R-N, and Q-R are NP-hard on general graphs, but provided polynomial algorithms for all four on tree graphs. The q-upgrading arc problem is very interesting in that it is applicable in a variety of time-sensitive settings. There are many opportunities in creating heuristics for general graphs, but there are other research opportunities as well. Specifically, we are interested in considering capacity on arc flow in conjunction with this problem, which would constrain the solutions. One interesting variation would involve redefining upgrading an arc to involve increasing capacity as well as decreasing travel time. This would follow from our discussions with United Parcel Service (UPS), where high speed lanes correspond to where large volumes of packages are shipped.

7 Acknowledgments

We would like to express our extreme gratitude to Arie Tamir for his assistance in reducing the complexity of the algorithm on general trees. This work was partially supported by the National Science Foundation, through grant number 0237726(Campbell), and served as the basis of the dissertation for Li Zhang. We would also like to acknowledge the input of Ranga Nuggehalli of the UPS. The q-upgrading arc problem was inspired by a discussion with him about the challenges in designing a delivery network for UPS.

References

[1] T. Aykin and G. F. Brown. Interacting new facilities and location-allocation problems. *Transportation Science*, 79:501–523, 1994.

[2] M. Blum, R. Floyd, V. Pratt, R.L. Rivest, and R. E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7:448–461, 1973.

[3] J. F. Campbell, A. Ernst, and M. Krishnamoorthy. Hub arc location problems: part 1 - introduction and results. Working paper, 2000.

[4] J. F. Campbell, A. Ernst, and M. Krishnamoorthy. Hub arc location problems: part 2 - formulations and optimal algorithms. Working paper, 2000.

[5] J. F. Campbell, A. Ernst, and M. Krishnamoorthy. Hub location problems. In Z. Drenzner and H. Hamacher, editors, *Facility location: applications and theory*. Springer–Verlag, 2002.

[6] J.F. Campbell. Integer programming of discrete hub location problems. *European Journal of Operational Research*, 72:387–405, 1994.

[7] P. K. Chan. Algorithms for library-specific sizing of combinational logic. *Proc. 27th DAC Conf*, pages 353–356, 1990.

[8] I. Demgensky, H. Noltemeier, and H.-C. Wirth. On the flow cost lowering problem. *European Journal of Operational Research*, 137:265–271, 2002.

[9] R.W. Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5:345, 1962.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[11] G. Y. Handler. Minimax location of a facility in an undirected tree graph. *Transportaton Science*, 7:287–293, 1973.

[12] G. Y. Handler and P. B. Mirchandani. *Location on networks: Theory and algorithms*. MIT Press, Cambridge, Massachusetts, 1979.

[13] A. V. Iyer and H. D. Ratliff. Accumulation point location on tree networks for guaranteed time distribution. *Management Science*, 36(8):958–969, 1990.

[14] B. Y. Kara and B. C. Tansel. On the single-assignment p-hub center problem. *European Journal of Operational Research*, 125(3):648–655, 2000.

[15] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50, 1956.

[16] C.L. Li, S.T. McCormick, and D. Simchi-Levi. On the minimum-cardinality-bounded-diameter and the bounded-cardinality-minimum-diameter edge addition problems. *Operations Research Letters*, 11:303–308, 1992.

[17] P. McGeer, R. Brayton, R. Rudell, and A. Sangiovanni-Vincentelli. Extended stuck-fault testability for combinational networks. *Proceedings of the 6th MIT Conference on Advanced Research in VLSI*, April 1990.

[18] P. B. Mirchandani and R. L. Francis. *Discrete location theory*. John Wiley and Sons, Inc., New York, 1990.

[19] S. Nickel, A. Schobel, and T. Sonnebon. Hub location problems in urban traffic networks. In M. Pursula and J. Niittymäki, editors, *Mathematical Methods on Optimization in Transportation Systems*. Kluwer Academic Publishers, 2001.

[20] M. E. O'Kelly. A quadratic integer program for the location of interacting hub facilities. *European Journal of Operational Research*, 32:393–404, 1987.

[21] M. E. O'Kelly and H. J. Miller. Solution strategies for the single facility minimax hub location problem. *Papers in Regional Science: The Journal of the RSAI*, 70:367–380, 1991.

[22] D. Paik, S. Reddy, and S. Sahni. Vertex splitting in dags and applications to partial scan designs and lossy circuits. Technical report, University of Florida, Gainesville, Florida, 1990.

[23] D. Paik, S. Reddy, and S. Sahni. Heuristics for the placement of flip-flops in partial scan designs and for the placement of signal boosters in lossy circuits. *Sixth International Conference on VLSI Design*, pages 45–50, 1993.

[24] D. Paik, S. Reddy, and S. Sahni. Deleting vertices in dags to bound path lengths. *IEEE Transactions on Computing*, 43(9):1091–1096, 1994.

[25] D. Paik and S. Sahni. Network upgrading problems. *Networks*, 26:45–58, 1995.

[26] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[27] A.A. Schoone, H.L. Bodlaender, and J. van Leeuwen. Diameter increase caused by edge deletion. *Journal of Graph Theory*, 11(3):409–427, 1987.

[28] A. Tamir. An $o(pn^2)$ algorithm for the p-median and related problems on tree graphs. *Operations Research Letters*, 19:59–64, 1996.

[29] B. C. Tansel, R. L. Francis, and T. J. Lowe. Location on networks: A survey. part 1. the p-center and p-median problems. *Management Science*, 29:482–497, 1983.

[30] B. C. Tansel, R. L. Francis, and T. J. Lowe. Location on networks: A survey. part 2. exploiting tree network structure. *Management Science*, 29:498–511, 1983.

[31] B. Yetis and B. C. Tansel. The single-assignment hub covering problem: Models and linearizations. *Journal of the Operational Research Society*, 54:59–64, 2003.

[32] L. Zhang. *The p-hub center allocation problem and the q-discount arc problem*. PhD thesis, The University of Iowa, Iowa City, Iowa, 2004.

[33] L. Zhang, A. Campbell, and T. Lowe. Heuristics for q-arc upgrading problems. Working Paper, 2004.