NEAR-OPTIMAL ALLOCATION OF DELAY REQUIREMENTS ON MULTICAST TREES

Hieu T. Tran
CATT Centre
School of Electrical and Computer Engineering, RMIT University, Australia
hieu@catt.rmit.edu.au

Richard J. Harris
CATT Centre
School of Electrical and Computer Engineering, RMIT University, Australia
richard@catt.rmit.edu.au

Abstract Knowing the QoS requirement for each link involved in a multicast connection, such that overall QoS requirement is satisfied, would greatly assist both QoS-based multicast routing and resource reservation processes. In the case of delay, the question of what delay requirements should be imposed on each link of a source-based multicast tree, such that the overall source-to-destination delay and inter-destination delay variation requirements are satisfied at minimum total tree cost, is the focal point of this paper. A major contribution of this paper is the development of a number of heuristic algorithms for (near-)optimal allocation of delay requirements using Genetic Algorithms (GAs). Initial tests, with multicast trees of different sizes (10- and 30-node trees), configurations (in terms of number of destinations and destination distribution), and overall delay requirements, show the ability of the algorithms in providing good solutions within a reasonable amount of time.

Keywords: Multicasting, QoS, resource allocation, source-to-destination delay, inter-destination delay variation, Genetic Algorithms, Mixed integer programming.

1. Introduction

QoS-based multicast routing has attracted a lot of interest due to increasing demand in group-based real-time applications that require stringent quality of service constraints, such as teleconferencing and distance learning. For applications involving group communication, multicasting

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35673-0_28 is more efficient than unicasting as it allows for transmission of a single copy of data to a group of destinations instead of sending a separate copy to each destination as in unicast routing.

As far as real-time applications are concerned, delay is one of the most important QoS parameters. Commonly, the delay from a source to all destinations should be bounded [Kompella et al., 1993][Salama et al., 1997][Rouskas and Baldin, 1997][Wang and Hou, 2000][Ergun et al., 2000][Lorenz et al., 2000][Lorenz and Orda, 2002]. In addition, the need for an upper bound on inter-destination delay variation (the maximum time difference between delay values from the source to different destinations) also arises for applications that require a certain level of group synchronization among various destinations [Akyildiz and Yen, 1996] [Rouskas and Baldin, 1997]. In exchange for having a delay bounded connection, the users must incur a connection cost and minimizing this cost is an important issue. This paper deals with the problem of optimal allocation of delay requirements over a source-based multicast tree given upper bounds on source-to-destination delay and inter-destination delay variation, and under the assumption that link cost function is non-increasing and (weakly) convex with delay requirements allocated to the link.

The rest of the paper is organized as follows. Section 1.2 describes models for link delays and link costs, and formally states the delay requirement allocation problem. Related work is provided in Section 1.3. The mathematical basis and three heuristic algorithms are presented in Section 1.4. Section 1.5 discusses the test results for the algorithms on a number of different test trees and different overall delay requirements. Section 1.6 concludes the paper.

2. Problem Formulation

A *source-based multicast tree* is represented as a directed tree $\mathbf{T} = \{\mathbf{V}, \mathbf{E}\}$ rooted at a *source* node s, where $\mathbf{V}$ and $\mathbf{E}$ represent the set of nodes and the set of directed links in the tree. $\mathbf{M}$ is the set of destinations ($\mathbf{M} \subseteq \mathbf{V} - \{s\}$). In the tree $\mathbf{T}$, a unique *path* from a node u to a node v, if one exists, is denoted by p_{uv}. A node v is defined as a *downstream node* of a node u if $u \in p_{sv}$; in that case, u is the *upstream node* of v. A *branch point* between p_{su} and p_{sv}, denoted by $\mathbf{B}(u, v)$, is defined as a node t on the two paths of which all the downstream nodes lying on this path must not be on the other path. A destination node having no upstream destination nodes is said to be the *multicast branch root* of a *multicast subgroup* that consists of the branch root and all of the branch root's downstream destination nodes. The subtree of $\mathbf{T}$ rooted at the branch root u spanning the multicast subgroup is defined as a *multicast branch* and denoted by $\mathbf{T}_u$. Set of all branch roots in $\mathbf{T}$ is denoted by $\mathbf{R}$. The i-th branch root is $\mathbf{R}(i)$. The number of multicast branches in the tree $\mathbf{T}$ is denoted by b (i.e., $|\mathbf{R}| = b$).

In a real network, packets traversing a link l experience 3 types of delay: propagation delay, queueing delay, and transmission delay. *Propagation delay* for a given link l is constant, and is denoted by δ_l. Queueing and transmission delays may vary from time to time as they depend on network load. Assuming that it is possible to impose on a link l a delay requirement d_l, then all packets traversing the link would take *no more* than d_l time. Hence, d_l may be considered as the *maximum delay* of a link l. Furthermore, in order to account for the variation of packet delay across the link l, we define a *delay variation bound* Δ_l: packets will take *no less* than $d_l - \Delta_l$ time to traverse link l. It is also assumed that the value of Δ_l for a given link l is constant during the time of interest (i.e., Δ_l may have a different value at some other time). In our model, all delays are assumed to take *integral values* only. A *delay partition* is defined as $\mathbf{S} = \{d_l\}_{l \in \mathbf{E}}$.

The delay requirement from a node u to a downstream node v is simply $D_{p_{uv}} = \sum_{l \in p_{uv}} d_l$. Delay variation bound of p_{uv} is computed by $\Delta_{p_{uv}} = \sum_{l \in p_{uv}} \Delta_l$. Instantaneous delay of a path p is guaranteed to be within the range $[D_p - \Delta_p, D_p]$. A *longest path*, p^*, in the tree $\mathbf{T}$ is defined as a path from the source to a destination that has the largest maximum delay, i.e. $p^* \equiv p_{sv}$ such that $D_{p_{sv}} \geq D_{p_{sv'}}$ $(\forall v' \in \mathbf{M})$.

Given source-to-destination delay bound D^e, and inter-destination variation delay bound D^i, a *feasible delay partition*, $\mathbf{S}_{D^eD^i} = \{d_l\}_{l \in \mathbf{E}}$, is a set of delay requirements allocated to the links of the multicast tree $\mathbf{T}$ such that:

$$D_{p_{su}} \leq D^e \quad \forall u \in \mathbf{M} \tag{1}$$

$$D_{p_{tu}} - (D_{p_{tv}} - \Delta_{p_{tv}}) \leq D^i \quad \forall u, v \in \mathbf{M}, t = \mathbf{B}(u,v) \tag{2}$$

$$\delta_l + \Delta_l \leq d_l \quad \forall l \in \mathbf{E} \tag{3}$$

The constraints above have the following interpretation: (1) requires that the delay from the source to a destination must not exceed the source-to-destination delay bound, (2) ensures that the difference in delays from the source to any pair of destinations will not be greater than the inter-destination delay variation bound, (3) simply requires that the delay requirement allocated to a link is feasible.

Total cost of the multicast tree $\mathbf{T}$ for a given $\mathbf{S}_{D^eD^i}$ is defined by $c(\mathbf{S}_{D^eD^i}) = \sum_{l \in \mathbf{E}} c_l(d_l)$ where $c_l(d_l)$ is the cost for imposing a delay requirement d_l on a link l. We assume that $c_l(d_l)$ is a *non-increasing* *convex function* (i.e., the link cost for a smaller bound is higher than the cost for a larger bound and cost savings due to requesting larger bound will diminish as the bound increases).

The problem of optimal delay allocation over a source-based multicast tree (MODA) can be formally stated as follows:

Problem MODA. *Given a multicast tree* $\mathbf{T} = (\mathbf{V}, \mathbf{E})$ *rooted at* s, $\{\delta_l, \Delta_l, c_l(d_l)\}_{l \in \mathbf{E}}$, *a set of destinations* $\mathbf{M} \subseteq \mathbf{V} - \{s\}$, *a source-to-destination delay bound* D^e, *and an inter-destination delay variation bound* D^i, *find a* feasible *partition* $\mathbf{S}^*_{D^e D^i}$ *such that* $c(\mathbf{S}^*_{D^e D^i}) \leq c(\mathbf{S}_{D^e D^i})$ *for all (other) feasible partition* $\mathbf{S}_{D^e D^i}$.

3. Related Work

The problem of partitioning end-to-end QoS requirements on unicast paths and multicast trees has been investigated in [Lorenz and Orda, 1998][Ergun et al., 2000][Lorenz et al., 2000][Lorenz and Orda, 2002]. Exact and approximate algorithms have been developed. Among those, [Lorenz and Orda, 2002]is most relevant to our problem, hence, we present a brief overview of their problem in this section.

Several assumptions , which are similar to ours, were made: 1) additive QoS; 2) integer QoS; 3) convex cost functions (link cost increases with level of QoS imposed on the link).

The problems of optimally partitioning QoS over a unicast path and a multicast tree are defined below. Please note that although [Lorenz and Orda, 2002]considers general additive QoS, in our discussion, we assume that the QoS is *delay*, which makes the following discussion more relevant to our problem without loss of generality.

Problem OPQ (Optimal Partition of QoS). *Given a path* p *and an end-to-end delay requirement* D, *find a feasible partition* $\mathbf{S}^*(p) = \{d_l^*\}_{l \in p}$, *such that* $c(\mathbf{S}^*(p)) \leq c(\mathbf{S}(p))$ *for all feasible partitions* $\mathbf{S}(p)$, *where a feasible partition is a partition for which* $\sum_{l \in p} d_l \leq D$.

Problem MOPQ (Multicast OPQ). *Given a multicast tree* $\mathbf{T}$ *rooted at* s *spanning a group of destinations* $\mathbf{M}$ *and an end-to-end delay requirement* D, *find a feasible partition* $\mathbf{S}^*(\mathbf{T})$, *such that* $c(\mathbf{S}^*(\mathbf{T})) \leq c(\mathbf{S}(\mathbf{T}))$ *for all feasible partitions* $\mathbf{S}(\mathbf{T})$, *where a feasible partition is a partition for which* $\sum_{l \in p_{sv}} d_l \leq D$ $(\forall v \in \mathbf{M})$.

It is easy to see that OPQ is, in essence, a resource allocation problem: each link of the path can be considered as an "activity" and the end-to-end delay requirement can be seen as the available "resource". Hence, OPQ can be solved using any of the algorithms for the resource allocation problem SCDR described in Chapter 4 of [Ibaraki and Katoh, 1998]. A greedy algorithm GREEDY-ADD provided in [Lorenz and Orda, 2002]is exactly the same as the Algorithm INCREMENT in [Ibaraki and Katoh, 1998].

GREEDY-ADD starts with a partition $\mathbf{S}(p) = \{d_l = 0\}_{l \in p}$, and iteratively adds a unit of delay to a link such that the total cost is reduced the most at each iteration, until total delay along the path reaches D. A faster algorithm, GREEDY-MOVE, starts with any feasible partition and gradually changes the current partition (by moving a unit of delay from one link to another link such that the cost is reduced the most) until no move could further lower the cost. A truly polynomial algorithm, BINARY-OPQ, finds an optimal partition in truly polynomial time by repeatedly executing GREEDY-MOVE with units of delay being halved after each iteration until the unit reaches 1.

Lorenz and Orda showed that MOPQ can also be solved in a greedy fashion. Let r denote the only link originating from the source node. The *branches*[1] of $\mathbf{T}$ are denoted by $\hat{\mathbf{T}} = \mathbf{T} - \{r\}$. $c_{\mathbf{T}}(D)$ denotes the cost of optimally allocating a delay D on a (sub-)tree $\mathbf{T}$. They first showed that, for any tree having depth of 2, $c_{\mathbf{T}}(D)$ is convex as long as D is optimally allocated over $\mathbf{T}$. This property was then proven for a tree $\mathbf{T}$ of arbitrary depth. They then proved the optimal substructure of the problem by showing that the delay partition on each and every subtree must be optimal in order for the delay partition on the tree to be optimal. Consequently, the optimal delay for link r can be found by running the greedy algorithms for the Problem OPQ on the 2-link path $(r, \hat{T})$, where $\hat{T}$ is a "link" representing the optimally allocated $\hat{\mathbf{T}}$ (because $c_{\hat{\mathbf{T}}}(d)$ is convex).

Algorithm TREE-ADD, which performs an augmentation (i.e., optimally adds a unit of delay to a tree) or a removal (i.e., optimally removes a unit of delay off a tree) given the current delay partition, was provided.

Algorithm BALANCE solves MOPQ with the help of TREE-ADD. The algorithm starts with any feasible partition. It iteratively moves a unit of delay between parts of the tree such that tree cost is maximally reduced at each iteration (it uses TREE-ADD to compute the cost changes due to augmentation or removal of delay from each subtree).

[1]The term *branch* in this subsection is not the same as the term *multicast branch* defined previously.

4. An Optimal Delay Allocation Algorithm

4.1 Properties of an optimal solution to MODA

An optimal solution to the Problem MODA possesses several interesting properties as stated in a number of lemmas given below. We shall informally prove the straightforward ones and will provide more formal proofs for ones that are less obvious.

LEMMA 4.1 *If there are optimal solutions to MODA, there must be at least one optimal solution for which the longest path(s) of the tree has a maximum total delay of D^e, i.e., $D^*_{p^*} = D^e$.*

We now informally prove this lemma. Let us assume that there is an optimal solution for which the longest path(s) of the tree is less than D^e. By adding the same amount of delay to all of the links stemmed from the source node such that the longest path(s) in the tree now has a total delay of D^e, we shall have another optimal solution that has the aforementioned property. The reasons for that are: 1) since the longest path having total delay of D^e, other paths must not have larger delay, hence, the source-to-destination delay requirement is satisfied; 2) the addition of the same amount of delay to all of the links stemmed from the source node does not affect inter-destination delay variation between any pair of destinations, hence, inter-destination delay variation requirement is also satisfied; 3) adding more delay to those links does not increase the total cost as link cost functions are non-increasing.

LEMMA 4.2 *There must be an optimal partition (if optimal solutions exist) for which the delay requirement from the source node to all the leaf nodes within the same multicast branch are the same.*

Assuming there is an optimal solution that does not have that property, consider adding more delay to incoming links of every leaf node (except leaf node v currently having the largest total delay from the source), such that the total delay from source to every leaf node is the same as that from source to node v. The new solution is also optimal because: 1) source-to-destination delay is satisfied; 2) the inter-destination delay variation requirement between leaf nodes on different multicast branches is satisfied as delay variation between the branch roots and leaf nodes of other multicast branches has not been changed. Inter-destination delay variation requirement between leaf nodes on the same multicast branch is obviously satisfied; 3) adding more delay to those links does not increase the total cost as link cost functions are non-increasing.

LEMMA 4.3 *Let* $u_i = \mathbf{R}(i)(i \in [1,b])$. *Let* $v_i \in \mathbf{T}_{u_i}$ *be the node to which the delay from the source is largest compared to other destinations in* $\mathbf{T}_{u_i}$. *The* necessary and sufficient condition *for* $\mathbf{S}$ *to be feasible is:*

$$D_{p_{sv_i}} \le D^e \quad \forall i \in [1,b] \tag{4}$$

$$\begin{aligned}\sum_{l \in p_{u_iv_i}} (\delta_l + \Delta_l) \le D_{p_{u_iv_i}} \le \min\{D^i,\\ \min\{D_{p_{sv_i}} - D_{p_{sv_j}} - \Delta_{p_{tu_i}} + D^i | 1 \le i \le b,\\ 1 \le j \le b, i \ne j, t = \mathbf{B}(u_i,u_j)\}\}\end{aligned} \tag{5}$$

Proof: (4) is by definition the necessary and sufficient condition for $\mathbf{S}$ to meet the source-to-destination delay constraint.

Let us consider the inter-destination delay variation constraint for destinations within a multicast branch $\mathbf{T}_{u_i}$ (Please note that for all $u, v \in \mathbf{T}_{u_i}$ and $t = \mathbf{B}(u,v)$, $D_{p_{sv}} \le D_{p_{sv_i}}$ and $D_{p_{su}} - \Delta_{p_{tu}} \ge D_{p_{st}} \ge D_{p_{su_i}}$)

$$\begin{aligned}&D_{p_{tv}} - (D_{p_{tu}} - \Delta_{p_{tu}}) \le D^i \quad \forall u,v \in \mathbf{T}_{u_i}, u \ne v, t = \mathbf{B}(u,v)\\ \Leftrightarrow\ &D_{p_{sv}} - (D_{p_{su}} - \Delta_{p_{tu}}) \le D^i \quad \forall u,v \in \mathbf{T}_{u_i}, u \ne v, t = \mathbf{B}(u,v)\\ \Leftrightarrow\ &D_{p_{sv_i}} - D_{p_{su_i}} \le D^i \Leftrightarrow D_{p_{u_iv_i}} \le D^i\end{aligned} \tag{6}$$

Now let us consider the delay variation constraint for destinations in different multicast branches.

$$\begin{aligned}&D_{p_{tv}} - (D_{p_{tu}} - \Delta_{p_{tu}}) \le D^i \quad \forall u \in \mathbf{T}_{u_i}, \forall v \in \mathbf{T}_{u_j}, \forall i \ne j, t = \mathbf{B}(u,v)\\ \Leftrightarrow\ &D_{p_{sv}} - (D_{p_{su}} - \Delta_{p_{tu}}) \le D^i \quad \forall u \in \mathbf{T}_{u_i}, \forall v \in \mathbf{T}_{u_j}, \forall i \ne j, t = \mathbf{B}(u,v)\\ \Leftrightarrow\ &D_{p_{sv_j}} - (D_{p_{su_i}} - \Delta_{p_{tu_i}}) \le D^i \quad \forall i \ne j, t = \mathbf{B}(u,v) = \mathbf{B}(u_i,v_j)\\ \Leftrightarrow\ &D_{p_{sv_j}} - (D_{p_{sv_i}} - D_{p_{u_iv_i}} - \Delta_{p_{tu_i}}) \le D^i \ \forall i \ne j, t = \mathbf{B}(u_i,v_j) = \mathbf{B}(u_i,u_j)\\ \Leftrightarrow\ &D_{p_{u_iv_i}} \le D_{p_{sv_i}} - D_{p_{sv_j}} - \Delta_{p_{tu_i}} + D^i \quad \forall i \ne j, t = \mathbf{B}(u_i,u_j)\end{aligned} \tag{7}$$

Remember that for all $v \in \mathbf{T}_{u_j}$, $D_{p_{sv}} \le D_{p_{sv_j}}$, and the order of nodes on path p_{su} is $s \to t \to u_i \to u$, hence, $D_{p_{su}} - \Delta_{p_{u_iu}} \ge D_{p_{su_i}}$ ($\forall u \in \mathbf{T}_{u_i}$). (6) and (7) hold if and only if (5) holds. The result follows. ■

LEMMA 4.4 *Let* $u_i = \mathbf{R}(i)$. *The optimal delay requirement from the source node* s *to any leaf node* $v_i \in \mathbf{T}_{u_i}$ $(i \in [1,b])$, *for at least one of the optimal solutions (if optimal solutions exist), is either equal to* D^e

or bounded by

$$D^e - D^i + \Delta_{p_{tv_i}} + \sum_{l \in p_{u_i v_i}} \delta_l \le D^*_{p_{sv_i}} \le D^e + D^i - \Delta_{p_{tv_j}} - \sum_{l \in p_{u_j v_j}} \delta_l \quad (8)$$

$$\textit{and,} \sum_{l \in p_{sv_i}} (\Delta_l + \delta_l) \le D^*_{p_{sv_i}} \le D^e \quad (9)$$

where t is the branch point between path p_{su_i} and a path, p_{sv_j}, having total delay requirement of D^e.

Proof: Let us consider the optimal solution mentioned in Lemma 4.1. If $D^*_{p_{sv_i}} = D^e$ then the lemma holds. Now let us assume that $D^*_{p_{sv_i}} \ne D^e$. From Lemma 4.2, it is clear that $D^*_{p_{sv}} \ne D^e$ ($\forall v \in \mathbf{T}_{u_i}$). Lemma 4.1 shows that there exists a $v_j \in \mathbf{T}_{u_j}$ ($j \ne i$) such that $D^*_{p_{sv_j}} = D^e$. Let $t = \mathbf{B}(v_i, v_j)$; it is obvious that $t \equiv \mathbf{B}(u_i, u_j)$. From (7), we have

$$\sum_{l \in p_{u_i v_i}} (\delta_l + \Delta_l) \le D^*_{p_{u_i v_i}} \le D^*_{p_{sv_i}} - D^*_{p_{sv_j}} - \Delta_{p_{tu_i}} + D^i$$

$$\therefore D^e + \Delta_{p_{tu_i}} - D^i + \sum_{l \in p_{u_i v_i}} (\delta_l + \Delta_l) \le D^*_{p_{sv_i}}$$

$$\therefore D^e - D^i + \Delta_{p_{tv_i}} + \sum_{l \in p_{u_i v_i}} \delta_l \le D^*_{p_{sv_i}}$$

Similarly,

$$\sum_{l \in p_{u_j v_j}} (\delta_l + \Delta_l) \le D^*_{p_{u_j v_j}} \le D^*_{p_{sv_j}} - D^*_{p_{sv_i}} - \Delta_{p_{tu_j}} + D^i$$

$$\therefore D^*_{p_{sv_i}} \le D^e - \Delta_{p_{tu_j}} + D^i - \sum_{l \in p_{u_j v_j}} (\delta_l + \Delta_l)$$

$$\therefore D^*_{p_{sv_i}} \le D^e + D^i - \Delta_{p_{tv_j}} - \sum_{l \in p_{u_j v_j}} \delta_l$$

Consequently, (8) holds. In the meantime, (9) obviously holds. ■

4.2 Problem Bounded-MOPQ

Let $u_i = \mathbf{R}(i)$ ($i \in [1, b]$). Let v_i be the node to which delay from the source is largest compared to other destinations in the multicast branch $\mathbf{T}_{u_i}$. The Problem Bounded-MOPQ (MOPQ-B) is stated as follows.

Problem MOPQ-B. *Given a tree* $\mathbf{T}$ *rooted at* s, $\{\delta_l, \Delta_l, c_l(d_l)\}_{l \in \mathbf{E}}$, $\{D^{max}_{p_{sv_i}}\}_{i \in [1,b]}$, *and* $\{D^{max}_{p_{u_i v_i}}\}_{i \in [1,b]}$, *find* $\mathbf{S} = \{d_l\}_{l \in \mathbf{E}}$ *such that* $c(\mathbf{S})$ *is minimum, subject to:*

$$D_{p_{sv_i}} = D^{max}_{p_{sv_i}} \quad \forall i \in [1,b] \tag{10}$$

$$D_{p_{u_i v_i}} \leq D^{max}_{p_{u_i v_i}} \quad \forall i \in [1,b] \tag{11}$$

$$\delta_l + \Delta_l \leq d_l \quad \forall l \in \mathbf{E} \tag{12}$$

Problem MOPQ-B is different from MOPQ because not only does it specifies delay requirements from the source to all leaf nodes but also imposes a bound on the delay requirement from the root of each multicast branch to the leaf nodes of the branch.

An exact algorithm for MOPQ-B is given below. Please note that: 1) information including branch roots, multicast branches, branch points, and the total propagation delay and the total delay variation bound from the source to every node in the tree must be determined prior to running this algorithm; 2) $GAIN^{+}_{\mathbf{T}^l}$ denotes the (negative) gain in the cost of a subtree $\mathbf{T}^l$ stemmed from a link l (including l) due to optimal augmentation of a single unit of delay to the current delay partition.

```
Algorithm TREE-INCREMENT
  for each link $l \in \mathbf{E}$ (traversing in post-order)
    $d_l \leftarrow \delta_l + \Delta_l$
    mark $l$ as "not-full"
    compute $GAIN^{+}_{\mathbf{T}^l}$
  end loop
  if ($D_{p_{sv_i}} \leq D^{max}_{p_{sv_i}}$  $\forall i \in [1,b]$)
    if ($D_{p_{u_i v_i}} \leq D^{max}_{p_{u_i v_i}}$  $\forall i \in [1,b]$)
      if ($D_{p_{sv_i}} = D^{max}_{p_{sv_i}}$  for some $i \in [1,b]$)
        mark all links on those $p_{sv_i}$ as "full"
      if ($D_{p_{u_i v_i}} = D^{max}_{p_{u_i v_i}}$  for some $i \in [1,b]$)
        mark all links on those $p_{u_i v_i}$ as "full"
      while (not all links "full")
        call TREE-AUGMENT /*see below*/
      end while
      return cost of the solution
    end if
  end if
  return infinite cost
End.
```

Algorithm TREE-AUGMENT is a modified version of TREE-ADD. Different from TREE-ADD, TREE-AUGMENT: 1) marks links that have become "full" (i.e., can no longer be augmented); 2) considers increment gain of a link as 0 if the link is already "full".

THEOREM 4.1 *TREE-INCREMENT accurately solves Problem MOPQ-B in* $\mathcal{O}(|\mathbf{E}|(\max_{i\in[1,b]}\{D^{max}_{p_{sv_i}} - \min_{v\in\mathbf{T}_{u_i}} \sum_{l\in p_{sv}}(\delta_l + \Delta_l)\}))$ *time.*

Proof: The existence of an upper bound on delay requirement from a branch root to the leaf nodes in the branch does not affect convexity of the cost of an optimally allocated subtree. A bounded subtree is equivalent to a unbounded subtree with a (convex) cost function that is the same as the cost function in the unbounded case for delay requirement less than or equal to the bound but remains constant (equal to the cost at the bound) for delay requirement larger than the bound. Hence, the greedy algorithm TREE-INCREMENT does provide an optimal solution. Since TREE-AUGMENT requires a depth-first-search, the computational complexity of TREE-AUGMENT is obviously $\mathcal{O}(|\mathbf{E}|)$. Furthermore, because TREE-AUGMENT is called $\max_{i\in[1,b]}\{D^{max}_{p_{sv_i}} - \min_{v\in\mathbf{T}_{u_i}} \sum_{l\in p_{sv}}(\delta_l + \Delta_l)\}$ times before all edges become "full", the result follows. ■

4.3 Heuristic algorithms for MODA

THEOREM 4.2 *Let* $\{\mathbf{S}^*_{D^eD^i}\}$ *denote a set of optimal solutions to the Problem MODA having the same* $\{D^*_{p_{sv_i}}\}$. $\{\mathbf{S}^*_{D^eD^i}\}$ *must also be the set of all optimal solutions the following Problem MOPQ-B with the bounds constructed as follows (MOPQ-BE):*

$$D^{max}_{p_{sv_i}} = D^*_{p_{sv_i}} \tag{13}$$

$$D^{max}_{p_{u_iv_i}} = \min\{D^i, \min\{D^*_{p_{sv_i}} - D^*_{p_{sv_j}} - \Delta_{p_{tu_i}} + D^i | 1 \le i \le b, \\ 1 \le j \le b, i \ne j, t = \mathbf{B}(u_i, u_j)\}\} \tag{14}$$

Proof: Since $\mathbf{S}^*_{D^eD^i}$ is an optimal solution to the Problem MODA, $D^*_{p_{sv}} \le D^e$ ($\forall v \in \mathbf{M}$), hence, a feasible solution to the Problem MOPQ-BE, which satisfies (10), must satisfy the source-to-destination constraint of MODA. In addition, from Lemma 4.3 and by the way that upper bounds on delay requirements from u_i to v_i on multicast branches are constructed as in (14), a feasible solution to MOPQ-BE also meets inter-destination delay variation constraint of MODA. Thus, a feasible solution to MOPQ-BE is a feasible solution to MODA. *In other words, a set of feasible solutions to MOPQ-BE is a subset of the feasible solution set of MODA.*

Following the same argument, we could prove that $\{\mathbf{S}^*_{D^e D^i}\}$ is a subset of the set of feasible solutions to MOPQ-BE: 1) requirement (10) is obviously satisfied; 2) (11) is met due to Lemma 4.3.

As $\{\mathbf{S}^*_{D^e D^i}\}$ ensure that the tree cost is minimum (as compared to all other $\mathbf{S}_{D^e D^i}$), $\{\mathbf{S}^*_{D^e D^i}\}$ must also be the optimal solution set of MOPQ-BE. ■

Theorem 4.2 implies that Problem MODA can be solved by finding the set $\{D_{p_{sv_i}}\}_{i\in[1,b]}$ for which the optimal solution to the corresponding MOPQ-BE (as described in Theorem 4.2) would result in least cost compared to other $\{D'_{p_{sv_i}}\}_{i\in[1,b]}$. The optimal solution to that instance of MOPQ-BE is also an optimal solution to MODA. Based on that observation, we structure our heuristic algorithm for MODA as below.

```
Algorithm MODA
  determine branch roots, branches, branch points of the tree
  compute delay variation bound from the source to every node
  compute propagation delay from the source to every node
  $bestCost \leftarrow infinity$
  for $j = 1$ to $b$ do
    set upper and lower bounds on $D_{p_{sv_j}}$ to $D^e$
    for all $i \neq j$ do
      compute upper and lower bounds on $D_{p_{sv_i}}$ /*see (8,9)*/
    end loop $i$
    initialize D-GENERATOR with these bounds
    while (D-GENERATOR can still generate $\{D_{p_{sv_i}}\}_{i\in[1,b]}$)
      $\{D^{max}_{p_{sv_i}}\}_{i\in[1,b]} \leftarrow \{D_{p_{sv_i}}\}_{i\in[1,b]}$
      compute $\{D^{max}_{p_{u_i v_i}}\}_{i\in[1,b]}$ /*see (14)*/
      $cost \leftarrow$ TREE-INCREMENT($\{D^{max}_{p_{sv_i}}\}, \{D^{max}_{p_{u_i v_i}}\}$)
      if ($cost < bestCost$)
        $bestCost \leftarrow cost$
        save the solution
      end if
    end while
  end loop $j$
End.
```

In the above algorithm, we make use of a D-GENERATOR. Its task is to generate vectors $\{D_{p_{sv_i}}\}_{i\in[1,b]}$ and to terminate the algorithm (by stop generating the "next" vector). Based on the results of Lemmas 4.1, 4.2, and 4.4, we limit the search space, which the generator will explore, with the lower and upper bounds on $\{D_{p_{sv_i}}\}_{i\in[1,b]}$ and $\{D_{p_{u_iv_i}}\}_{i\in[1,b]}$ during initialization of the generator.

This algorithm structure allows us to try different techniques in implementing the D-GENERATOR. A naive generator would enumerate all possible combinations $\{D_{p_{sv_i}}\}_{i\in[1,b]}$, which may work for small problems but will take a prohibitively long time for larger ones. We choose to implement a number of D-GENERATORs using Genetic Algorithms (GAs)[2] and a greedy approach.

In our GA implementation, we use a SIMPLE GA which is similar to that described in [Goldberg, 1989]. In order to code the vector $\{D_{p_{sv_i}}\}_{i\in[1,b]}$ as a binary string, we consider two alternative methods: 1) converting each element to a binary number, and combining these binary numbers together to create a binary string; 2) reducing the vector to a scalar value, then, converting this scalar value to a binary number. Besides, TREE-INCREMENT is used for evaluating of the "fitness" of a $\{D_{p_{sv_i}}\}_{i\in[1,b]}$.

Our greedy D-GENERATOR generates 2 vectors. The first vector represents a partition for which the delay requirement from the source to each destination is at a maximum, whereas the second one represents a partition for which the upper bounds on delay requirements along all multicast branches are the same (in most cases).

5. Numerical Results

We implemented the Algorithm MODA in C++ using the following libraries: 1) GAlib (GA component library) [Wall, 1999], 2) Graph Template Library (GTL) [Forster et al., 1999], 3) BRITE (topology generation library) [QNLB, 2001]. Employing the two coding schemes, we built two GA-based algorithms, namely, MODA-GA Coding 1 and MODA-GA Coding 2. MODA-Greedy is an algorithm using the greedy approach. For benchmarking, we also developed an LP-based algorithm, MODA-LP, that uses ILOG^{TM} CPLEX 7.1 LP solver in solving the MIP model of the Problem MODA.

Four randomly generated trees were used for testing: two 10-node trees, and two 30-node trees. Propagation delay and delay variation bound of each link were assigned random values from 1(ms) to 4(ms) and from 1(ms) to 10(ms) respectively. Links were randomly assigned a cost function from a set of 3 different cost functions. In each tree, the root represented the source node and all leaf nodes represented the des-

[2]For a full account of GAs, the reader is referred to the books by Holland [Holland, 1975]and Goldberg [Goldberg, 1989].

Table 1. Tree cost produced by the MODA algorithms. The percentage shows the difference between the cost and the optimal cost.

Tree Configuration	*LP optimal*	*Greedy*	*Coding 1*		*Coding 2*	
			Best	*Worst*	*Best*	*Worst*
$\lvert V\rvert = 10; \lvert M\rvert = 4; b = 4$ $D^e = 300; D^i = 30$	1.89 0.0%	1.89 0.0%	1.90 0.6%	1.91 1.0%	1.9 0.4%	1.91 1.0%
$\lvert V\rvert = 10; \lvert M\rvert = 5; b = 3$ $D^e = 300; D^i = 30$	1.92 0.0%	2.18 13.5%	1.94 0.9%	2.08 8.2%	1.92 0.0%	2.12 10.0%
$\lvert V\rvert = 30; \lvert M\rvert = 16; b = 7$ $D^e = 500; D^i = 200$	N/A	4.39	4.79	5.55	4.90	6.53
$\lvert V\rvert = 30; \lvert M\rvert = 17; b = 5$ $D^e = 500; D^i = 200$	N/A	4.22	4.54	5.66	4.77	6.29

tinations. In addition, we also randomly selected other in-tree nodes as destinations. The source-destination delay bound and inter-destination delay variation bound were chosen to be 300(ms) and 30(ms) for the 10-node trees and 500(ms) and 200(ms) for the 30-node trees respectively.

We tested the algorithms with the test trees and settings as described above on a Pentium II 550 MHz computer with 384 MB RAM. MODA-LP was run one for each tree in order to obtain the optimal solution. MODA-Greedy was run one, whereas MODA-GA Coding 1 and MODA-GA Coding 2 were run 5 times alternatively for each tree. The test results are summarized in the Table 1.

For the 10-node trees, MODA-LP took approximately 15 seconds to complete while the greedy algorithm took a fraction of a second and the two GA-based algorithms took about 10 seconds. As the tree size increased to 30 nodes, MODA-LP failed after running for 20 minutes (due to memory shortage) whereas the greedy algorithm took about 1 second and the other two algorithms needed about 60 seconds in order to provide good solutions.

We also tested our algorithms with a 10-node tree ($\lvert V\rvert = 10; \lvert M\rvert = 5; b = 3; D^e = 300$) for different value of D^i. As the problem became infeasible for $D^i < 19$(ms), we varied D^i from 19(ms) to 29(ms). The costs of the solutions found by the algorithms and the costs after normalization by the MODA-LP (optimal) cost, for different values of D^i, were shown in Figure 1.

As can be seen from the test results, GA-based algorithms consistently provided good solutions, whereas, the greedy algorithm was able to come up with very good solutions in some cases but provided less good solutions in other cases.

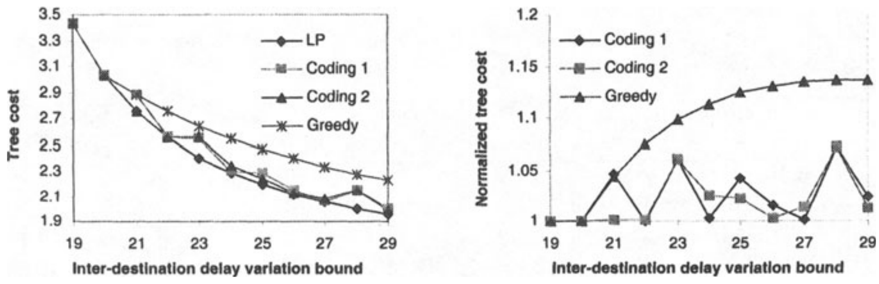

Figure 1. Costs of the solutions found by the MODA algorithms

6. Conclusions

This paper addressed the Problem MODA that effectively extends the Problem MOPQ in [Lorenz and Orda, 2002]by adding inter-destination delay variation constraint. We derived a number of lemmas and theorems that provide the foundation for the development of two GA-based algorithms and a greedy algorithm. Initial test showed that our algorithms are capable of providing good solutions.

This work is ongoing work. Firstly, although the GA-based algorithms could provide good solutions, they may converge too slowly for practical use; on the other hand, the greedy algorithm could run very fast but may not find feasible solutions in some cases. Hence, other fast heuristics should be developed to further shorten the computation time and to increase the chance of finding feasible solutions. Secondly, other algorithms for more complex link delay models could be developed. Thirdly, a distributed version of the algorithm needs to be developed as centralized algorithms, like ours, generally have scalability problem. Finally, the question of how this allocation algorithm could be incorporated into QoS routing algorithms also needs to be properly addressed.

References

[Akyildiz and Yen, 1996] Akyildiz, I. F. and Yen, W. (1996). Multimedia group synchronization protocols for integrated services networks. *IEEE JSAC*, 14(1):162 – 173.

[Ergun et al., 2000] Ergun, F., Sinha, R., and Zhang, L. (2000). Qos routing with performance-dependent costs. In *INFOCOM '2000*, pages 137 – 146.

[Forster et al., 1999] Forster, M. et al. (1999). Gtl - graph template library. *http://www.infosun.fmi.uni-passau.de/GTL/*.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

[Ibaraki and Katoh, 1998] Ibaraki, T. and Katoh, N. (1998). *Resource Allocation Problems.* The MIT Press.

[Kompella et al., 1993] Kompella, V. N., Pasquale, J. C., and Polyzos, G. C. (1993). Multicast routing for multimedia communication. *IEEE/ACM Transaction on Networking*, 1:286 - 292.

[Lorenz and Orda, 1998] Lorenz, D. H. and Orda, A. (1998). Optimal partition of qos requirements on unicast paths and multicast trees. Research Report EE 1167, Department of Electrical Engineering, Technion, Haifa, Israel, fpt://ftp.technion.ac.il/pub/suported/ee/ Network/lor.mopq98.ps.

[Lorenz and Orda, 2002] Lorenz, D. H. and Orda, A. (2002). Optimal partition of qos requirements on unicast paths and multicast trees. *IEEE/ACM Transactions on Networking*, 10:102 - 114.

[Lorenz et al., 2000] Lorenz, D. H., Orda, A., Raz, D., and Shavitt, Y. (2000). Efficient qos partition and routing of unicast and multicast. In *IWQoS '2000*, pages 75 - 83.

[QNLB, 2001] QNLB (2001). Brite: Boston university representative internet topology generator. *http://www.cs.bu.edu/brite/.*

[Rouskas and Baldin, 1997] Rouskas, G. N. and Baldin, I. (1997). Multicast routing with end-to-end delay and delay variation constraints. *IEEE JSAC*, 15:346 - 347.

[Salama et al., 1997] Salama, H. F., Reeves, D. S., and Viniotis, Y. (1997). Evaluation of multicast routing algorithms for real-time communication on high-speed networks. *IEEE JSAC*, 15:332–345.

[Wall, 1999] Wall, M. (1999). Galib. *http://lancet.mit.edu/ga/.*

[Wang and Hou, 2000] Wang, B. and Hou, J. (2000). Qos-based multicast routing for distributing layered video to heterogeneous receivers in rate-based networks. In *IEEE INFOCOM' 2000.*