GOLD: An Overlay Multicast Tree with Globally Optimized Latency and Out-Degree

Jun Guo, Sanjay Jha
School of Computer Science and Engineering
The University of New South Wales, Australia
Email: {jguo, sjha}@cse.unsw.edu.au

Suman Banerjee
Department of Computer Sciences
University of Wisconsin-Madison, USA
Email: suman@cs.wisc.edu

Technical Report UNSW-CSE-TR-0615

August 2006

Abstract

End-to-end delay and interface bandwidth usage are two important performance metrics in overlay multicast networks. This paper demonstrates that, by jointly optimizing the placement of multicast service nodes and the routing strategy for overlay multicast networks, it is possible to find an overlay multicast tree that both minimizes the maximum end-to-end delay between the source and the destinations, and balances the interface bandwidth usage within the set of multicast service nodes. Motivated by this important observation, we propose in this paper a joint optimization problem for overlay mutlicast networks, where we wish to find a globally optimized overlay multicast tree with minimum average end-to-end delay subject to two stringent constraints: 1) The maximum end-to-end delay is bounded by the unicast latency from the source to the farthest destination in the physical topology; 2) The interface bandwidth usage is balanced within the set of multicast service nodes. This problem is shown to be NP-hard. We present a low complexity greedy algorithm that obtains good quality approximate solutions to the problem. We further show how the greedy algorithm enables the design of a weight-coded genetic algorithm that achieves closer-to-optimal solutions with reasonable computational complexity.

1 Introduction

Several two-tier overlay multicast network infrastructures have been recently proposed as feasible solutions to support scalable inter-domain multicast services for real-time applications [3, 13, 20]. In an overlay multicast network, a set of dedicated devices called *multicast service nodes* (MSNs) are distributed in the network, typically collocated at the sites of a selected number of access routers. Each MSN functions not only as a multicast server for the access router it is collocated with, but more importantly as a transit point for replicating and forwarding real-time streaming traffic to other MSNs or access routers in the overlay multicast network by means of routine unicast mechanisms. The topology of the overlay multicast network is hence considered as a fully connected mesh.

End-to-end delay and interface bandwidth usage are two important performance metrics in overlay multicast networks [3, 20]. An overlay routing path from the source to a destination likely passes through a series of MSNs. As a consequence, it is important to minimize the end-to-end delay so as to reduce the jitter experienced by real-time streaming traffic on each overlay routing path. This is essential to the provision of good quality delay-sensitive multicast services in the overlay multicast network. On the other hand, each MSN in the overlay multicast network has limited access bandwidth. Thus, it is important to manage the utilization of interface bandwidth resources efficiently. It was shown in [20] that it is useful to balance interface bandwidth usage among MSNs for each multicast session. This way, the overlay multicast network is able to support a larger number of concurrent multicast sessions with good service qualities.

Earlier proposals for building an overlay multicast tree typically aim to optimize the backbone tree that contains MSNs only [3, 20]. They assume that the MSN placement solution is given and the access subtrees rooted at each particular MSN are also given. The routing protocols proposed in [3] and [20] addressed three constrained spanning tree problems: 1) Minimize the maximum end-to-end delay within the set of MSNs subject to the interface bandwidth constraint at each MSN; 2) Minimize the average end-to-end delay within the set of MSNs subject to the interface bandwidth constraint at each MSN; and 3) Balance the interface bandwidth usage at each MSN subject to the bound on the maximum end-to-end delay. Recently, Lao *et al.* [12] proposed to optimize exclusively the access subtrees rooted at each specific MSN. They formulated the MSN placement problem to find optimal placement solutions such that the average latency between access routers and their corresponding MSNs is minimized.

These existing approaches were designed to achieve optimal solutions in their respective problem domains. However, as will be demonstrated in this paper, they are not able to fully utilize the network resources and to achieve effectively the best routing performance from the global perspective of the overlay multicast network. We shall see that, by jointly optimizing the placement of MSNs and the routing strategy for both the backbone tree and the access subtrees, it is feasible to obtain an overlay multicast tree that globally minimizes the maximum end-to-end delay, balances the interface bandwidth usage within the set of MSNs, and yet obtains good performance results on average end-to-end delay.

The contributions of this paper include:

1. We motivate and propose a new and challenging constrained spanning tree problem for the overlay multicast network (Section 2). We demonstrate that, by solving the proposed problem, it is feasible to obtain an overlay multicast tree with globally optimized end-to-end delay of overlay routing paths and interface bandwidth usage of MSNs.

2. We develop a low complexity greedy algorithm that obtains good quality approximate solutions to this problem (Section 3). For all the large size network topologies tested in our simulation experiments, the worst quality solution obtained from the greedy algorithm is below 16% from the optimal solution.

3. We design a weight-coded genetic algorithm (GA) which can readily utilize the greedy algorithm and obtain closer-to-optimal solutions to this problem (Section 4). For all the large size network topologies tested in our simulation experiments, the worst quality solution obtained from the weight-coded GA is below 8% from the optimal solution.

4. We provide an integer linear programming (ILP) formulation for this problem. The ILP model can be used to compute the optimal solutions for small size network topologies (Section 5). Moreover, we believe that our proposed ILP model may pave the way for future

research into the development of possible mathematical approaches for solving this challenging problem [18].

The remainder of this paper is organized as follows: Section 2 deals with the rigorous formulation of the joint optimization problem. We also discuss the motivation of the problem using an explicit numerical example. In Section 3, we present the greedy algorithm, and we provide intuitive explanations to the working methodology of the greedy algorithm and an analysis of its computational complexity. The details of the weight-coded GA are presented in Section 4. The ILP formulation of the problem is given in Section 5. Simulation experiments are reported in Section 6. Finally, we provide concluding remarks in Section 7.

2 Joint Optimization Problem

Consider an overlay multicast network in the form of a complete directed graph $G = (V, E)$, where V is the set of N nodes and $E = V \times V$ is the set of edges. Let node r be the source. Each of the other $N - 1$ destination nodes in $V - \{r\}$ represents an access router. The directed edge $\langle i, j \rangle$ in E from node i to node j represents the unicast path of latency $l_{i,j} > 0$ from node i to node j in the physical topology. In this paper, we assume $l_{i,j} = l_{j,i}$ for all $i, j \in V$, $i \neq j$. By $\{l_{i,j}\}$, we denote the matrix of unicast latency quantities between each pair of nodes in G and we assume that it has been given. M, $M < N$, nodes (including the source node r by default) are selected as MSNs which constitute the set P. For one particular multicast session, each outgoing edge directed away from node i, $i \in P$, indicates one unit of interface bandwidth usage of node i, and is equivalent to one count towards the out-degree of node i in the graph representation. Note that we shall hence use the terms "interface bandwidth usage" and "out-degree" interchangeably in the rest of this paper.

The overlay multicast tree can be represented by a directed spanning tree of G rooted at the source node r. A directed edge from node i to node v, $i \in V$, $v \in V - \{r\}$, can be included in the spanning tree if and only if node i is in P. Consequently, the set of internal nodes of the spanning tree is composed of MSNs only, which constitutes the backbone tree. If we shall further enforce at least one unit of interface bandwidth usage of each MSN, the set of leaf nodes of the spanning tree is thus exclusively composed of non-MSN nodes, which constitute the access subtrees rooted at MSNs.

2.1 Performance metrics

For each destination node v in the set $V - \{r\}$, we define $R_{r,v}$ as the set of directed edges that form its overlay routing path from the source node r. Let $L_{r,v}$ denote the latency of the overlay routing path from the source node r to the destination node v. Given the unicast latency matrix $\{l_{i,j}\}$, we readily have

$$L_{r,v} = \sum_{\langle i,j\rangle \in R_{r,v}} l_{i,j} \ . \tag{1}$$

Let L_{max} denote the maximum end-to-end delay, given by

$$L_{max} = \max_{v \in V-\{r\}} L_{r,v} \ . \tag{2}$$

Let $\bar{L}$ denote the average end-to-end delay, given by

$$\bar{L} = \frac{1}{N-1} \sum_{v \in V-\{r\}} L_{r,v} \ . \tag{3}$$

Consider the case where the unicast latency matrix $\{l_{i,j}\}$ satisfies the triangle inequality (see e.g. [1], page 11). For each destination node v in the set $V-\{r\}$, the latency $L_{r,v}$ of its overlay routing path from the source node r can not be smaller than the latency $l_{r,v}$ of its unicast path from the source node r. Thus, we can establish the lower bound L^{lb}_{max} on L_{max} as

$$L^{lb}_{max} = \max_{v \in V-\{r\}} l_{r,v} \tag{4}$$

and the lower bound $\bar{L}^{lb}$ on $\bar{L}$ as

$$\bar{L}^{lb} = \frac{1}{N-1} \sum_{v \in V-\{r\}} l_{r,v} \ . \tag{5}$$

Let $d(i)$ denote the out-degree of an internal node i, $i \in P$. The following proposition states a property on the sum $\sum_{i \in P} d(i)$ of out-degrees in an overlay multicast tree.

Proposition 2.1 *The sum of out-degrees in an overlay multicast tree is $N-1$.*

Proof. It follows from Corollary 1.5.3 of [6] that a spanning tree with N nodes has exactly $N-1$ edges. By the definition of an overlay multicast tree, between each pair of nodes i and j, the edge is either directed from node i to node j, or directed from node j to node i. In either case, it contributes one count towards the sum of out-degrees. ■

We define an out-degree balancing index F, given by

$$F = \max_{i \in P} d(i) - \min_{i \in P} d(i) \ . \tag{6}$$

A smaller value of F indicates a more balanced interface bandwidth usage within the set of MSNs. Let F^{lb} denote the lower bound on F. Clearly, $F^{lb} = 0$ if

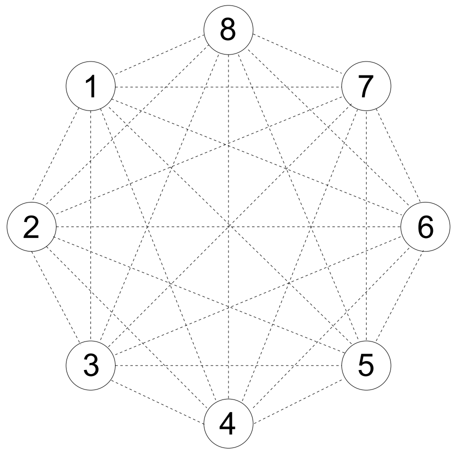
$$N - 1 = kM, \ \ k = 1, 2, \ldots \tag{7}$$

and we require each internal node to have an out-degree of exactly k. In situations where (7) does not hold, $F^{lb} = 1$. In such cases, letting $k = \lfloor \frac{N-1}{M} \rfloor$ and $n = N - 1 - kM$, we require n internal nodes to have an out-degree of exactly $k+1$, and the remaining $M - n$ internal nodes to have an out-degree of exactly k.

2.2 Problem formulation

Before we formally present the statement of the joint optimization problem, we shall first clarify the motivation of such a problem by considering an eight-node example illustrated in Figure 1. Let node 3 be the source node and assume three MSNs to be placed. A routine computation of (4) and (5) on the unicast latency matrix $\{l_{i,j}\}$ provided in Figure 1 gives $L_{max}^{lb} = 4$ and $\bar{L}^{lb} = 2.14$ for this particular instance. Since $N - 1 = 7$ and $M = 3$, we have $F^{lb} = 1$, and for achieving this we require one MSN to have an out-degree of exactly three, and the other two MSNs to have an out-degree of exactly two.

Using the ILP formulation presented in [12], we obtain the MSN placement solution, as shown in (a) of Figure 2, where the three MSNs are placed at nodes 6 and 7 as well as node 3. In addition, the access subtrees are constructed in such a way that nodes 1, 2, 4 and 5 are connected to node 3, and node 8 is connected to node 6. This gives a minimum total distance of six between each node and its associated MSN. Given this solution for the access subtrees, all the three possible



	①	②	③	④	⑤	⑥	⑦	⑧
①	-	1	2	1	3	5	5	6
②	1	-	1	2	2	4	4	5
③	2	1	-	1	1	3	3	4
④	1	2	1	-	2	4	4	5
⑤	3	2	1	2	-	3	2	4
⑥	5	4	3	4	3	-	3	1
⑦	5	4	3	4	2	3	-	2
⑧	6	5	4	5	4	1	2	-

Figure 1: Overlay topology and unicast latency for the eight-node example.

solutions for the backbone tree are listed as (b), (c) and (d) in Figure 2. For each solution, we use (1) to compute for each of the seven destination nodes the latency of its overlay routing path from the source node. These results are provided in the figures for the corresponding nodes. We then use (2) to find the maximum end-to-end delay L_{max}, compute the average end-to-end delay $\bar{L}$ using (3), and obtain the out-degree balancing index F from (6). We observe in these three solutions that both (b) and (c) incur a large maximum end-to-end delay and highly imbalanced interface bandwidth usage within the set of MSNs. On the other hand, (d) achieves both L_{max}^{lb} and $\bar{L}^{lb}$ for this particular instance. Nevertheless, it results in an extremely imbalanced out-degree distribution. Interestingly, if we shall enforce $F = 1$ with the MSN placement solution of (a), the best solution for backbone tree and access subtrees, as (e) of Figure 2 shows, can merely achieve $L_{max} = 7$, which also leads to a large average end-to-end delay. This implies that the issue of strategic placement of MSNs itself is not trivial and indeed has significant impact on the routing performance of the overlay multicast tree.

If we go one step further, and jointly optimize the entire overlay multicast tree, but without explicitly considering the constraint on balancing the interface bandwidth usage, the best solution on F that can achieve both L_{max}^{lb} and $\bar{L}^{lb}$ is shown in (f) of Figure 2. Such a solution again suffers a highly imbalanced out-degree distribution. On the other hand, if we shall enforce the constraint on balancing the interface bandwidth usage, the best solution to achieve L_{max}^{lb} with the MSN placement solution of (f) is shown in (g) of Figure 2. The interface bandwidth usage is more

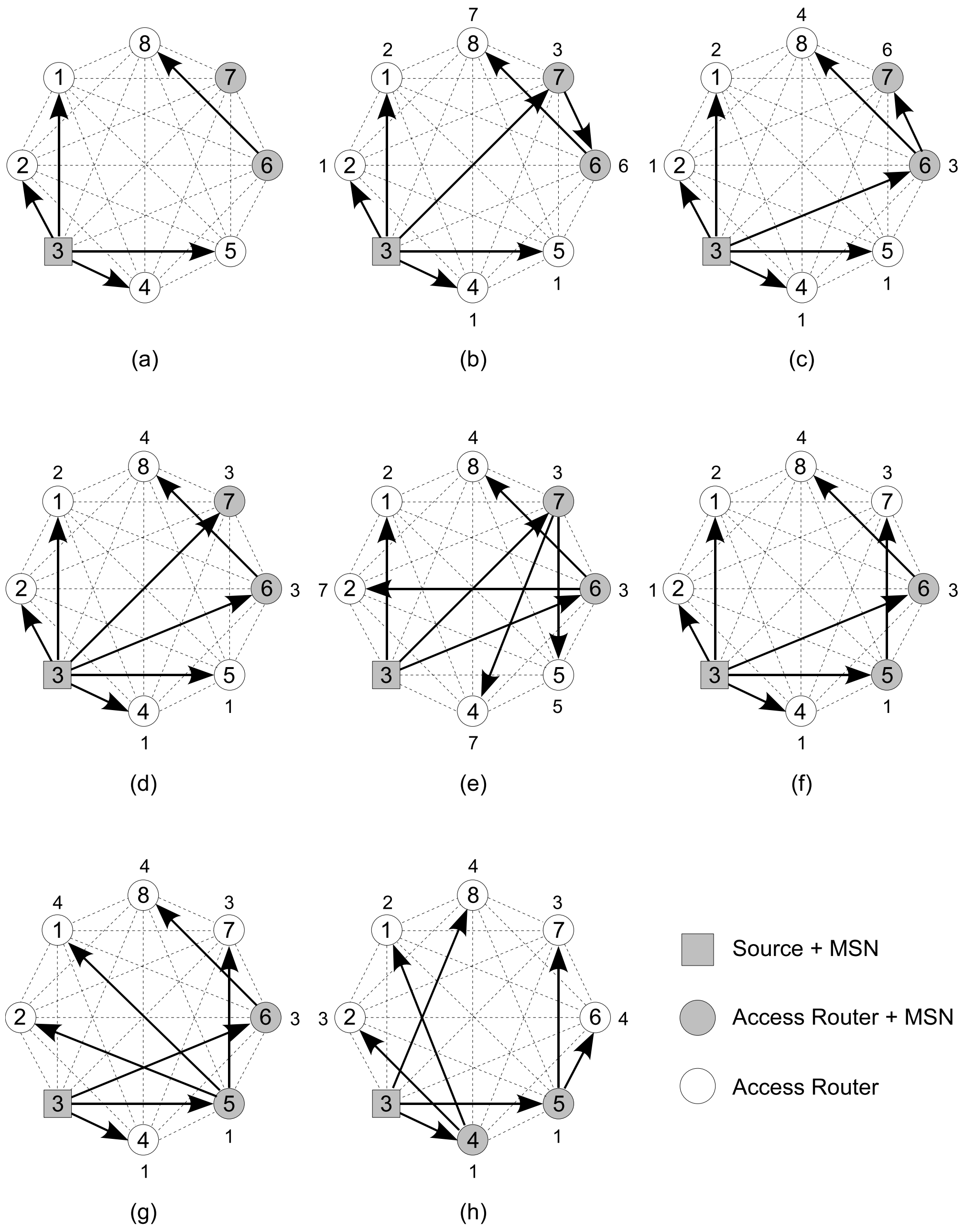


Figure 2: Overlay multicast tree solutions for the eight-node example: (a) Solution for MSN placement and access subtrees due to [12]. All three possible solutions for the backbone tree of (a) are shown in (b), (c) and (d); (b) $L_{max} = 7$, $F = 4$, $\bar{L} = 3$; (c) $L_{max} = 6$, $F = 5$, $\bar{L} = 2.57$; (d) $L_{max} = 4$, $F = 6$, $\bar{L} = 2.14$; (e) The best solution on L_{max} to achieve $F = 1$ using the MSN placement of (a) leads to $L_{max} = 7$ and $\bar{L} = 4.43$; (f) The best solution to achieve $L_{max} = 4$ and $\bar{L} = 2.14$ requires $F = 4$; (g) The best solution on F to achieve $L_{max} = 4$ using the MSN placement of (f) leads to $F = 2$ and $\bar{L} = 2.71$; (h) The best solution to achieve $L_{max} = 4$ and $F = 1$ results in $\bar{L} = 2.57$.

balanced in this way with $F = 2$, but it leads to a poor performance result on $\bar{L}$. Solution (h) in Figure 2 achieves both L_{max}^{lb} and F^{lb}, and yet obtains a good $\bar{L}$ result.

Clearly, we see from this example that any isolated approach optimizing exclusively either the backbone tree [3, 20] or the access subtrees [12] is less likely able to utilize the network resources efficiently. On the other hand, with joint optimization on the entire overlay multicast tree, it is feasible to achieve minimal possible maximum end-to-end delay, balanced interface bandwidth usage, and yet good performance results on average end-to-end delay. As we have discussed before, the two most important performance metrics for overlay multicast networks are maximum end-to-end delay and interface bandwidth usage. Therefore, we propose the following joint optimization problem for overlay multicast networks:

Given a complete directed graph $G = (V, E)$ of N nodes, find a constrained directed spanning tree T of G rooted at the source node r, such that $\bar{L}$ is minimized, and T is subject to constraints on: 1) M, $M < N$, nodes (including the source node r by default) are selected as internal nodes, all other $N - M$ nodes are for leaf nodes; 2) L_{max} is bounded by $L_{max}^{B} \geq L_{max}^{lb}$; 3) F is restricted to F^{lb}.

If we shall enforce $L_{max}^{B} = L_{max}^{lb}$, the optimal solution to any instance of this problem, should it exist, is certainly an overlay multicast tree with Globally Optimized Latency and out-Degree (GOLD). This is particularly true if it would also realize $\bar{L} = \bar{L}^{lb}$. In the rest of this paper, we shall refer to this problem as the GOLD spanning tree problem. Such a heavily constrained spanning tree problem is clearly different from any existing constrained spanning tree problem (see e.g. [3, 4, 15, 20] and references therein). This problem is NP-hard, which can be established by the following theorem:

Theorem 2.2 *The GOLD spanning tree problem is NP-hard.*

Proof. It is easy to see that the decision version Π of the GOLD spanning tree problem is in the class of NP. This is because a non-deterministic algorithm needs only to guess a directed spanning tree rooted at the source node and check in polynomial time if it satisfies all constraints of the GOLD spanning tree problem. Having $M = N$ and allowing $L_{max}^{B} = L_{max}^{ub}$, this restricts Π to a variant of the traveling salesman problem respecting mean arrival time of the tour starting from the source node ([8], page 211), which is NP-complete according to [19]. ■

We thus resort to heuristic methods to find near-optimal solutions for this challenging problem.

3 Greedy Algorithm

The greedy algorithm that we propose for the GOLD spanning tree problem contains three stages. The first stage deals with the issue of strategic placement of MSNs to constitute the set of internal nodes for the overlay multicast tree T. The second stage is dedicated to the task of forming the backbone tree of T. Finally in the third stage, we manage the connections between leaf nodes and internal nodes to resolve the access subtrees for T. We shall see that our greedy algorithm ensures that T satisfies all constraints of the GOLD spanning tree problem except the bound L_{max}^B on L_{max}. Nevertheless, at any intermediate step during the process of incrementally building T, we bear in mind the goal of greedily minimizing L_{max} as well as $\bar{L}$. A procedure that implements this greedy algorithm is provided in Figure 3. We shall next explain each module in detail.

3.1 MSN placement

Given the unicast latency matrix $\{l_{i,j}\}$, we compute s_v for each node v in the set $V - \{r\}$ by

$$s_v = \sum_{j \in V - \{r,v\}} (l_{r,v} + l_{v,j}) \ . \tag{8}$$

We then identify $M - 1$ nodes in $V - \{r\}$ with the smallest values on s_v. These $M - 1$ nodes together with the source node r constitute the set P of internal nodes of the overlay multicast tree T.

Our method of MSN placement is different from [12]. Their method aims to minimize the average latency between leaf nodes and internal nodes. However, their method may not find the optimal placement of MSNs from the global perspective of the overlay multicast tree. As we have observed previously from the eight-node example considered in Figure 2, none of the internal nodes (except the source node) in the optimal solution, (h) of Figure 2, can be identified by their method. Intuitively, our method aims to find nodes that (should they be chosen as internal nodes) would more likely result in smaller latency of the overlay routing path between each leaf node and the source node.

```
INPUT:
    Overlay topology $G = (V, E)$
    Source node $r$
    Unicast latency matrix $\{l_{i,j}\}$
    # of MSNs $M$
Output:
    GOLD spanning tree $T$
Procedure:
    /* Initialization */
    $T = \varnothing$; $N = |V|$; $k = \lfloor \frac{N-1}{M} \rfloor$;
    if ($kM = N - 1$)
        $n = M$;
    else
        $n = N - 1 - kM$; $k = k + 1$;
    /* MSN placement */
    for each $v \in V - \{r\}$ do
        $s_v = \sum_{j \in V - \{r,v\}} (l_{r,v} + l_{v,j})$;
    $P = \{M - 1$ nodes with the smallest $s_v\}$; $P = P + \{r\}$; $\bar{P} = \varnothing$;
    for each $i \in P$ do
        $d(i) = 0$;
    /* backbone tree */
    $P' = \{r\}$; $L_{r,r} = 0$;
    for each $v \in P - \{r\}$ in the non-decreasing order according to $s_v$ do
        find $u \in P' - \bar{P}$ with the smallest $L_{r,u} + l_{u,v}$;
        if ties exist on $u$
            choose $u$ with the largest $\sum_{j \in P - P' - \{v\}} (L_{r,u} + l_{u,j})$;
        $L_{r,v} = L_{r,u} + l_{u,v}$; $P' = P' + \{v\}$; $d(u) = d(u) + 1$; $T = T + \langle u, v \rangle$;
        /* out-degree balancing */
        if $d(u) = k$
            $\bar{P} = \bar{P} + \{u\}$; $n = n - 1$;
            if $n = 0$
                $n = M$; $k = k - 1$;
                for each $u \in P' - \bar{P}$ with $d(u) = k$ do
                    $\bar{P} = \bar{P} + \{u\}$; $n = n - 1$;
    /* access subtrees */
    $W = V - P$;
    while $W \neq \varnothing$ do
        for each $v \in W$ do
            $\delta_v = \min_{j \in P - \bar{P}} (L_{r,j} + l_{j,v})$;
        find $v \in W$ with the largest $\delta_v$;
        if ties exist on $v$
            choose $v$ with the largest $\sum_{j \in P - \bar{P}} (L_{r,j} + l_{j,v})$;
        $u = \text{argmin}_{j \in P - \bar{P}} (L_{r,j} + l_{j,v})$;
        if ties exist on $u$
            choose $u$ with the largest $\sum_{j \in W - \{v\}} (L_{r,u} + l_{u,j})$;
        $L_{r,v} = L_{r,u} + l_{u,v}$; $W = W - \{v\}$; $d(u) = d(u) + 1$; $T = T + \langle u, v \rangle$;
        /* out-degree balancing */
        if $d(u) = k$
            $\bar{P} = \bar{P} + \{u\}$; $n = n - 1$;
            if $n = 0$
                $n = M$; $k = k - 1$;
                for each $u \in P - \bar{P}$ with $d(u) = k$ do
                    $\bar{P} = \bar{P} + \{u\}$; $n = n - 1$;
```

Figure 3: Greedy algorithm for the GOLD spanning tree problem.

Apply (8) to the eight-node example considered in Figure 2. The s_v values are computed to be:

v	1	2	4	5	6	7	8
s_v	33	24	24	22	38	38	47

Node 5 clearly has the smallest value of s_v. A tie exists between nodes 2 and 4. Selecting either node, however, yields an optimal solution to this particular instance, which is confirmed by the ILP model to be presented in Section 5.

3.2 Out-degree balancing

An important objective of our greedy algorithm is to guarantee that the out-degree balancing index F of internal nodes of T is restricted to F^{lb}. As we have discussed in Section 2.1, if $F^{lb} = 0$, we require each of the M internal nodes to have an out-degree of exactly k, where $k = \frac{N-1}{M}$. This case can be trivially resolved. For each node v that has just been added to the partial tree of T, we check the cumulated out-degree $d(i)$ of the internal node i which connects node v. If $d(i) = k$, we mark node i so that node i will not be considered by any of the remaining unconnected nodes in the set $V - \{r\}$.

On the other hand, if $F^{lb} = 1$, we require n internal nodes to have an out-degree of exactly $k + 1$, and the remaining $M - n$ internal nodes to have an out-degree of exactly k, where in this case $k = \lfloor \frac{N-1}{M} \rfloor$ and $n = N - 1 - kM$. To achieve this, we make sure that once the n-th internal node whose out-degree reaches $k+1$ has been marked, we further check for each unmarked internal node if its out-degree has reached k. If so, we mark such nodes correspondingly, again to make sure that they will not be considered by any of the remaining unconnected nodes in the set $V - \{r\}$.

3.3 Backbone tree

Our greedy algorithm for constructing the backbone tree is an improved variant of the Compact Tree (CT) algorithm proposed in [20]. While CT was designed to address the problem of minimizing L_{max} of the backbone tree only, we aim for greedily minimizing L_{max} and $\bar{L}$ simultaneously for the entire overlay multicast tree.

For this purpose, we begin by arranging all $M-1$ internal nodes in the set $P-\{r\}$ in a non-decreasing order according to their s_v values computed from (8). Then, we add these internal nodes one-by-one in that order to form the backbone tree of T. Intuitively, nodes that are added to the backbone tree earlier in this stage are more likely to be connected by some other internal nodes later in this stage as well as some leaf nodes later in the access subtrees. Consequently, by adding internal nodes in such an order, we hope to achieve an overall smaller overlay latency of both internal nodes and leaf nodes.

Let P' denote the set of all internal nodes in the partial tree of T constructed so far. Let $\bar{P}$ denote the set of all internal nodes that have been marked. Starting from the initial tree with P' including the source node r only, for each unconnected internal node v in the ordered set $P-\{r\}$, we find an unmarked internal node u in the partial tree of T constructed so far such that $L_{r,u}+l_{u,v}$ is the smallest. Ties are broken by choosing node u such that $\sum_{j\in P-P'-\{v\}}(L_{r,u}+l_{u,j})$ is the largest. Intuitively, if we do not connect node v to node u at this iteration, later node u would more likely lead to larger overlay latency of some other internal nodes not yet in the tree. After node v is added to the tree, we update P', and we apply the out-degree balancing module to check if node u needs to be marked into the set $\bar{P}$.

3.4 Access subtrees

Given that the backbone tree of T is formed, we are now at the stage of describing how we connect leaf nodes one-by-one to the backbone tree to complete the access subtrees for T.

Let W denote the set of all leaf nodes not yet in the partial tree of T, and initialize W to $V-P$. At any iteration, for each node v in the set W, we find an unmarked internal node u in the backbone tree of T which minimizes $\delta_v = L_{r,u}+l_{u,v}$. We then identify the node v in W with the largest value of such δ_v, and add it to the partial tree of T by creating a directed edge from node u to node v. Intuitively, if we do not connect node v at this iteration, later it would more likely incur larger overlay latency and contribute towards a larger value of $\bar{L}$.

In cases where ties exist either on u or on v, we break the ties on u by choosing node u such that $\sum_{j\in W-\{v\}}(L_{r,u}+l_{u,j})$ is the largest. Intuitively, again, if we do not go for node u at this iteration, later node u would more likely lead to larger overlay latency of some other leaf nodes not yet in the tree. On the other hand, we break the ties on v by choosing node v such that $\sum_{j\in P-\bar{P}}(L_{r,j}+l_{j,v})$ is the largest. Similarly, if we do not connect node v at this iteration, later node v would be more likely connected to an internal node which yields larger overlay latency of node v.

Once node v is added to the tree, we update W, and we again apply the out-degree balancing module to check if node u needs to be marked into the set $\bar{P}$.

3.5 Computational complexity

Clearly, for the MSN placement, it first requires a total of $2(N-2)$ additions. Then, in order to identify $M-1$ nodes with the smallest values on s_v, it requires a total of $(M-1)(2N-M-2)/2$ comparisons of complexity $O(MN)$, given $N \geq M$. Alternatively, we apply a quick sort of complexity $O(N\log N)$ on s_v if $M > \log N$. For the backbone tree, it turns out that to add the m-th internal node, $m = 3, 4, \ldots, M-1$, it requires $m-1$ additions and $m-2$ comparisons and an extra $2(m-1)(M-m)$ additions and $m-1$ comparisons at most for tie-breaking. To add the M-th internal node, it merely requires $M-1$ additions and $M-2$ comparisons. Therefore, the complexity of this module is $O(M^3)$ in the worst. Similarly, it can be shown that the complexity of the module for the access subtrees is at most $O(MN^2)$. Since $N \geq M$, the greedy algorithm is $O(MN^2)$ in complexity.

4 Weight-Coded Genetic Algorithm

GAs are population-based stochastic search and optimization approaches inspired by the mechanism of natural selection which obeys the rule of "survival of the fittest" [10]. As shown in Figure 4, in a typical implementation of GAs, a population of chromosomes is processed. Each chromosome represents a candidate solution to the problem. Starting from an initial population of randomly created chromosomes, GAs perform multi-directional stochastic search through a genetic evolution process without the need for any problem information except the objective function values. During each generation of the evolution process, a selection operator is employed to introduce certain bias. As a result, chromosomes with higher fitness values have better chances to survive and reproduce through genetic operators. It is hoped that after a certain number of

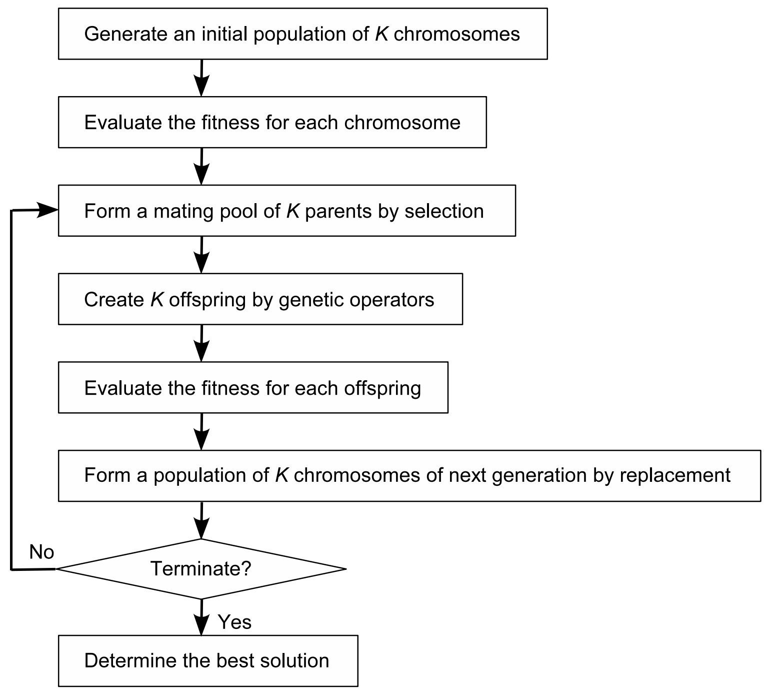

Generate an initial population of K chromosomes
Evaluate the fitness for each chromosome
Form a mating pool of K parents by selection
Create K offspring by genetic operators
Evaluate the fitness for each offspring
Form a population of K chromosomes of next generation by replacement
No
Terminate?
Yes
Determine the best solution

Figure 4: Flowchart of GAs.

generations, the best chromosome represents a good quality solution that is reasonably close to the optimal solution.

GAs have been extensively used for solving various real-world complex optimization problems due to their broad applicability, ease of use and global perspective [9]. In particular, they have found successful applications in a number of tree-based network optimization problems (see e.g. [21, 23] and references therein). In all cases, it was observed that GAs can achieve near-optimal solutions for fairly large size instances with reasonable computational effort. We choose to implement a weight-coded GA [16] for the GOLD spanning tree problem, since it uses a node-based encoding approach for chromosome representation rather than an edge-based encoding approach which is inefficient in the context of a complete graph [21]. Moreover, it readily utilizes the proposed greedy algorithm and thus effectively operate the stochastic search in the heavily constrained solution space without necessitating the design of any specialized GA operator [17].

4.1 Implementation

By weighted coding [16], we express a chromosome of GA in the form of a real-valued vector $\mathbf{w}$ of size N. Each gene w_i, $i = 1, 2, \ldots, N$, of the chromosome $\mathbf{w}$ holds a weight in the range $(0, 1)$. The string of N weights is used to produce an altered version of the unicast latency matrix $\{l_{i,j}\}$ by

$$l'_{i,j} = l_{i,j} \cdot w_i \cdot w_j \tag{9}$$

for $\langle i, j\rangle \in E$. The greedy algorithm is applied to identify an overlay multicast tree represented by $\mathbf{w}$ using the altered matrix $\{l'_{i,j}\}$, but the corresponding L_{max} and $\bar{L}$ values of this overlay multicast tree are computed using the original matrix $\{l_{i,j}\}$. Clearly, the nature of the greedy algorithm guarantees that such an overlay multicast tree satisfies all constraints of the GOLD spanning tree problem except the bound L^B_{max} on L_{max}. Consequently, we can effectively operate GA in the heavily constrained solution space of the GOLD spanning tree problem by simply exploring the space of weights and handling any constraint violation on L_{max} in each generation of the evolution process.

For the purpose of constraint handling, we use an efficient approach proposed in [5] to guide the weight-coded GA towards the constrained optimum without the need for any penalty parameter in the fitness function. Let $L_{max}(\mathbf{w})$ and $\bar{L}(\mathbf{w})$ respectively denote the L_{max} and $\bar{L}$ values of the overlay multicast tree represented by $\mathbf{w}$. Let $\bar{L}_{max}$ be the $\bar{L}$ value of the worst feasible chromosome in the population. Let L^{ub}_{max} be an upper bound on L_{max}. Let $\bar{L}^{ub}$ be an upper bound on $\bar{L}$. Note that both L^{ub}_{max} and $\bar{L}^{ub}$ can be safely set to $(N-1) \cdot \max_{\langle i,j\rangle \in E} l_{i,j}$ for the GOLD spanning tree problem.

If $\mathbf{w}$ is a feasible chromosome, i.e. $L_{max}(\mathbf{w}) \leq L^B_{max}$, we evaluate its fitness $f(\mathbf{w})$ by

$$f(\mathbf{w}) = \Big[L^{ub}_{max} - L^B_{max}\Big] + \Big[\bar{L}^{ub} - \bar{L}(\mathbf{w})\Big]. \tag{10}$$

On the other hand, if $\mathbf{w}$ is an infeasible chromosome, i.e. $L_{max}(\mathbf{w}) > L^B_{max}$, we evaluate its fitness $f(\mathbf{w})$ by

$$f(\mathbf{w}) = \Big[L^{ub}_{max} - L_{max}(\mathbf{w})\Big] + \Big[\bar{L}^{ub} - \bar{L}_{max}\Big]. \tag{11}$$

With the fitness function defined in this form, we ensure that the higher fitness value a chromosome has, the better solution it represents. Moreover, we also guarantee that 1) the fitness value of a feasible chromosome is higher than that of an infeasible chromosome, 2) the comparison between two feasible chromosomes is purely based on the objective function value $\bar{L}$, and 3) the comparison between two infeasible chromosomes uses the information of constraint violation on L_{max} alone.

The evolution process is initiated with a randomly generated population of size K, except for one in which all weights are set to one. This chromosome represents the approximate solution obtained from the greedy algorithm directly using the original matrix $\{l_{i,j}\}$. Seeding the population in this knowledge-augmented way can substantially improve the search efficiency of GA [16]. Starting from this initial population, in each generation of the evolution process, we form a mating pool by selecting K good chromosomes, some of which may be duplicate, from the current population. We use the tournament selection operator for this purpose [5]. For two chromosomes chosen for competition in a binary tournament, the winner goes to the one with a higher fitness value. Moreover, we set the rule such that any chromosome is made to participate in exactly two tournaments. As a result, any chromosome in the current population will have at most two copies as the parents in the new population. This is particularly important in our context to prevent the good quality approximate solution from becoming a predominant chromosome which would likely lead to premature convergence of GA.

For each pair of coupled chromosomes in the mating pool, a uniform crossover operator is used to promote random information exchange between the two parents. One such operation at rate r_c generates a pair of new chromosomes, each of which inherits some parts of genetic materials from both parents. To introduce greater variability into the offspring, for each gene w_i, $i = 1, 2, \ldots, N$, of an offspring $\mathbf{w}$, we further apply a uniform mutation operator with a small probability r_m. If w_i is chosen for mutation, it is altered with a weight randomly generated from the range $(0, 1)$.

We evaluate the fitness of each offspring using either (10) or (11) depending on if it represents a valid overlay multicast tree. The parents and the offspring are then combined into one transitional population (of size $\geq K$ and $\leq 2K$). The K chromosomes with highest fitness values are chosen and placed in the new population of the next generation.

The evolution process is terminated after a predefined number of generations have been completed. Alternatively, we presume convergence if the best chromosomes are not improved over a certain succession of generations. In either case, we decide the best overlay multicast tree with the highest fitness value from the last population.

4.2 Computational complexity

For each generation of the weight-coded GA, it is clear that the complexity of selecting K parents from the current population of K chromosomes is no more than $O(K)$. It then requires at most K crossover operations each of which has complexity $O(N)$, and is followed by KN mutation operations of overall complexity $O(KN)$ in the worst case. Since the evaluation of the fitness for each of the K offsprings requires exactly one computation of the greedy algorithm which is of complexity $O(MN^2)$, the overall computational complexity of one generation of GA is no more than $O(KMN^2)$.

5 Integer Linear Programming Formulation

Here we provide an ILP formulation for the GOLD spanning tree problem. Let the 0-1 variables x_i, $i \in V$, indicate if node i is selected as an internal node. Let the 0-1 variables $p^v_{i,j}$, $v \in V - \{r\}$, $\langle i,j \rangle \in E$, indicate if the directed edge $\langle i,j \rangle$ from node i to node j is included in the overlay routing path from the source node r to the destination node v. Let the 0-1 variables $t_{i,j}$, $\langle i,j \rangle \in E$, indicate if the directed edge $\langle i,j \rangle$ from node i to node j is used by the overlay multicast tree. Let the non-negative real variables d_{max} and d_{min} respectively identify the maximum out-degree and the minimum out-degree among the internal nodes. For the latter, it is useful to define a non-negative real variable y_i for each $i \in V$. The GOLD spanning tree problem aiming to find an overlay mutlicast tree that achieves L^B_{max} and F^{lb} and minimizes the average end-to-end delay $\bar{L}$ can now be formulated as:

$$\text{Minimize} \quad \frac{1}{N-1} \sum_{v \in V - \{r\}} \Big(\sum_{\langle i,j \rangle \in E} p^v_{i,j} l_{i,j} \Big) \tag{12}$$

subject to

$$x_r = 1 \tag{13}$$

$$\sum_{v \in V - \{r\}} x_v = M - 1 \tag{14}$$

$$\sum_{i:\langle i,r \rangle \in E} p^v_{i,r} = 0, \ \forall v \in V - \{r\} \tag{15}$$

$$\sum_{j:\langle r,j \rangle \in E} p^v_{r,j} = 1, \ \forall v \in V - \{r\} \tag{16}$$

$$\sum_{i:\langle i,v \rangle \in E} p^v_{i,v} = 1, \ \forall v \in V - \{r\} \tag{17}$$

$$\sum_{j:\langle v,j \rangle \in E} p^v_{v,j} = 0, \ \forall v \in V - \{r\} \tag{18}$$

$$\sum_{i:\langle i,m \rangle \in E} p^v_{i,m} = \sum_{j:\langle m,j \rangle \in E} p^v_{m,j}, \ \forall v \in V - \{r\}, \ \forall m \in V - \{r, v\} \tag{19}$$

$$\sum_{v \in V - \{r\}} p^v_{i,j} \leq (N-1) \cdot t_{i,j}, \ \forall \langle i,j \rangle \in E \tag{20}$$

$$\sum_{\langle i,j \rangle \in E} t_{i,j} = N - 1 \tag{21}$$

$$\sum_{\langle i,j \rangle \in E} t_{i,j} \leq (N-1) \cdot x_i, \ \forall i \in V \tag{22}$$

$$\sum_{\langle i,j \rangle \in E} p^v_{i,j} l_{i,j} \leq L^B_{max}, \ \forall v \in V - \{r\} \tag{23}$$

$$d_{max} \geq \sum_{j:\langle i,j \rangle \in E} t_{i,j}, \ \forall i \in V \tag{24}$$

$$y_i = (N-1) \cdot (1 - x_i) + \sum_{j:\langle i,j \rangle \in E} t_{i,j}, \ \forall i \in V \tag{25}$$

$$d_{min} \leq y_i, \ \forall i \in V \tag{26}$$

$$d_{max} - d_{min} \leq F^{lb} \tag{27}$$

The objective function presented in (12) is equivalent to (3) by the definition of $p^v_{i,j}$. Equations (13) and (14) restrict that M nodes (including the source node r by default) are selected as internal nodes. Equations (15) to (20) ensure that the solution is a directed spanning tree rooted at the source node r. More explicitly, they enforce one single overlay routing path for each source-destination pair. Equation (21) restricts that the sum of out-degrees counts $N - 1$, which is a necessary property of an overlay multicast tree established in Proposition 2.1. Equation (22) ensures that the directed edge $\langle i, j \rangle$ from node i to node j can be included in the overlay multicast tree if and only if node i is an internal node. Equation (23) restricts that the maximum end-to-end delay is bounded by L^B_{max}. Equations (24) to (27) enforces the balance of out-degree among nodes selected as internal nodes. All the equations jointly ensure that the solution is a directed spanning tree rooted at the source node r and satisfies all constraints of the GOLD spanning tree problem established in Section 2.

6 Simulation Experiments

We have tested our proposed heuristic methods for the GOLD spanning tree problem on many different network topologies. Here we report experimental results for three small size topologies and three large size topologies.

The various network topologies used in our experiments were obtained from the GT-ITM topology generator [22]. In particular, the three small size topologies ($N = 20,\ 25,\ 30$) were generated using the flat random graph model of GT-ITM with an average node degree between 3 and 5. Unicast latency between different pairs of nodes in these graphs varies between 1 and 80. The three large size topologies ($N = 100,\ 300,\ 500$) were generated using the transit-stub graph model of GT-ITM with an average node degree between 3 and 4. Unicast latency between different pairs of nodes in these graphs ranges from 1 to 500. It was reported in [22] that the transit-stub model was designed to generate a domain structure resembling that of the Internet.

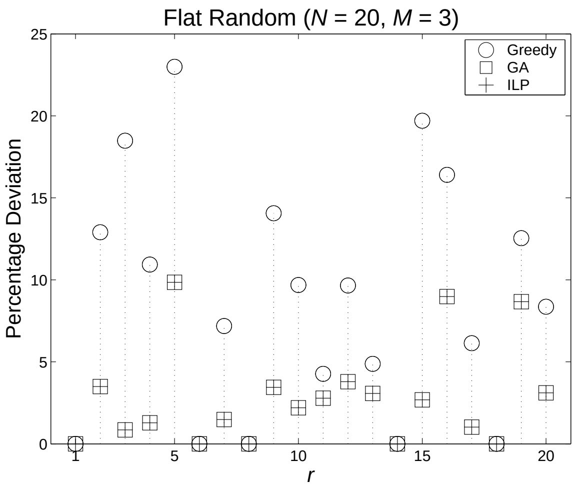
Based on these network topologies, we design the following experiments to study the performance of the proposed heuristic methods. For each of the small size topologies, we consider the various instances in which we set each different node to be the source node. In all such instances, the number M of MSNs is fixed to 3. Due to the small size of these topologies, we are able to compute the optimal solutions by solving the ILP model presented in Section 5. The CPLEX tool [11] can be in general used to solve ILP models for problems with a reasonable number of integer variables. With these concrete optimal results in hand, we are able to confirm the capability of GAs in our context, although it has been widely established in the literature.

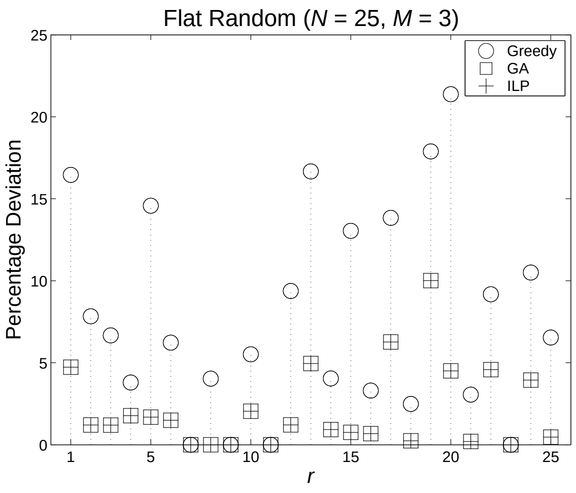
ILP is, however, known to have an exponential computational complexity and is unlikely to be solved for problems with a large number of integer variables. Consequently, we are not able to obtain the ILP results for the large size topologies. Nevertheless, given that we have established the lower bound $\bar{L}^{lb}$ on $\bar{L}$, we can use $\bar{L}^{lb}$ as the benchmark to verify the performance of the heuristic methods in such instances. Due to the limited space, we do not present experimental results for the large size topologies with respect to different source nodes. Instead, we randomly choose one node as the source node and vary M from 5 to 30 for each such topology.

In all experiments, we conduct the weight-coded GA with ten independent runs each of which starts from a different initial population. The crossover rate r_c is varied from 0.5 to 1.0 at the step size of 0.1. The mutation probability r_m is set to $1/N$ [2]. Each run repeats the evolution process with up to 100 generations for the small size topologies and with up to 1000 generations for the large size topologies. The population size K is set to N for the small size topologies and 100 for the large size topologies. We also compare the performance of the weight-coded GA with a random search. For that purpose, we use the weighted coding approach to randomly generate the same number of chromosomes explored by the weight-coded GA. For each string of weights so obtained, we compute its corresponding overlay multicast tree and check if it results in a valid solution. Similar to the implementation of GA, we include the approximate solution obtained from the greedy algorithm as one of the solutions in the random search.

The performance results are reported in Figure 5 for the small size topologies and in Figure 6 for the large size topologies. In both figures, we plot the $\bar{L}$ results in the form of a percentage deviation from the lower bound $\bar{L}^{lb}$ given by $(\bar{L} - \bar{L}^{lb})/\bar{L}^{lb}(\%)$. Note that in all the experiments we have conducted for the purpose of this paper, we have enforced $L^B_{max} = L^{lb}_{max}$. It is interesting to see that even with such a stringent constraint, our proposed greedy algorithm always yields a valid solution to the heavily constrained problem.

Results in Figure 5 clearly confirm the capability of the weight-coded GA. For all the various instances, this heuristic method has successfully found the optimal solutions within the limited number of generations. The performance of the greedy algorithm is reasonably good for these small size topologies, within 20% in the worst case from the optimal solution. Interestingly, it happens in these small size topologies that, in all cases where the greedy algorithm hits the optimal solution, it indeed achieves the lower bound $\bar{L}^{lb}$ on $\bar{L}$. This demonstrates that it is possible that a globally optimal solution to an instance of the GOLD spanning tree problem can truly realizes





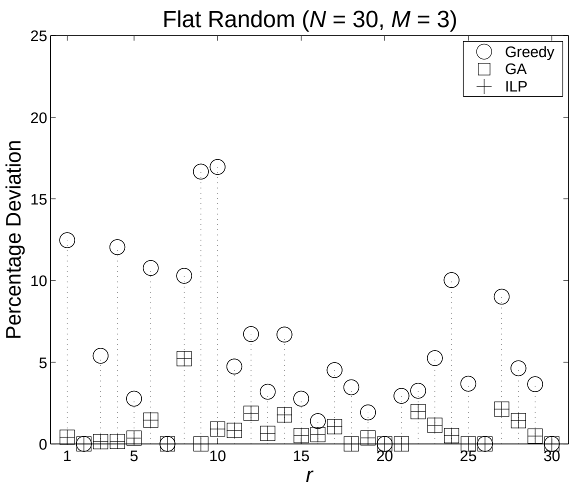


Figure 5: Quality of solutions measured by $(\bar{L} - \bar{L}^{lb})/\bar{L}^{lb}(\%)$ to the small size instances of the GOLD spanning tree problem, obtained from 1) the greedy algorithm, 2) the weight-coded GA and 3) the ILP formulation.

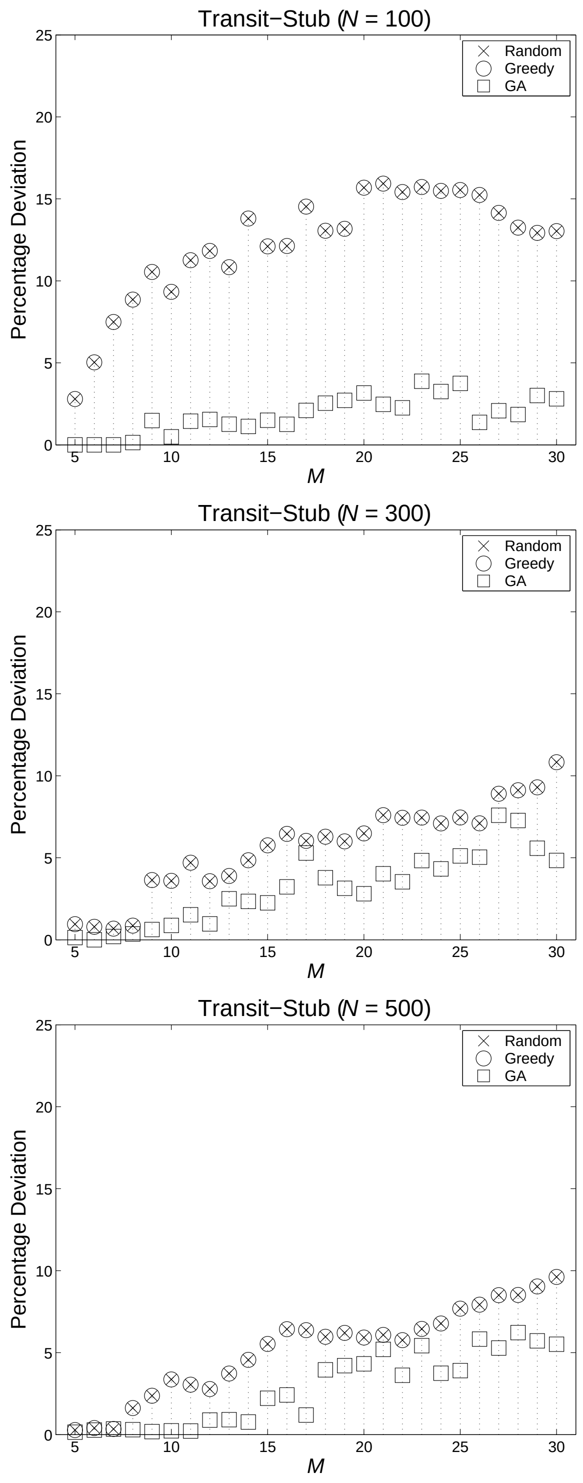


Figure 6: Quality of solutions measured by $(\bar{L} - \bar{L}^{lb})/\bar{L}^{lb}(\%)$ to the large size instances of the GOLD spanning tree problem obtained from 1) the random search, 2) the greedy algorithm and 3) the weight-coded GA.

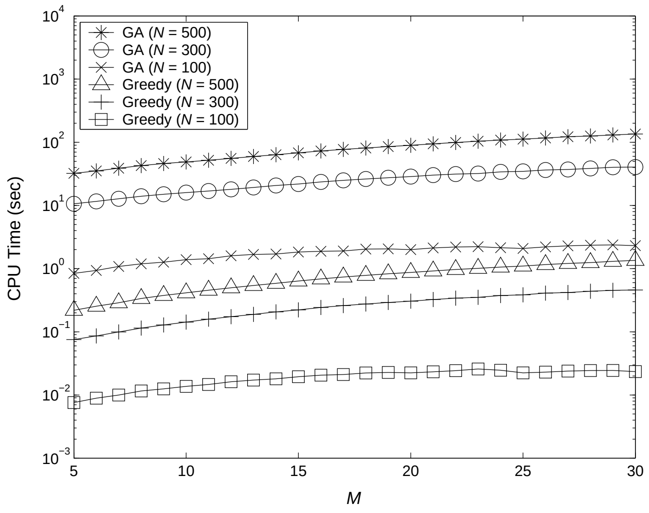


Figure 7: CPU time required on a 2.4 GHz Pentium 4 machine for one computation of the greedy algorithm and for one generation of the weight-coded GA.

the three lower bounds F^{lb}, L^{lb}_{max} and $\bar{L}^{lb}$ simultaneously.

Such perfect results of the heuristic algorithm do not recur in Figure 6 for the large size topologies. Nevertheless, the weight-coded GA yet hits the lower bound $\bar{L}^{lb}$ in four instances and approaches very closely to $\bar{L}^{lb}$ in several other instances, when the ratio between N and M is sufficiently large. The worst case performance of the weight-coded GA is below 8% from $\bar{L}^{lb}$. The random search is not even able to hit a valid solution given that it searches the same number of chromosomes as of the weight-coded GA. The worst case performance of the greedy algorithm occurs in the graph with $N = 100$, which is below 16% from $\bar{L}^{lb}$. Compared with the weight-coded GA, it is clear that the performance of the greedy algorithm is considerably good when the size of the network topology is large. This is especially true in consideration of its lower computational complexity and hence the smaller amount of CPU time it requires as demonstrated in Figure 7.

7 Conclusion and Future Work

We have proposed a new and challenging constrained spanning tree problem which we call GOLD for overlay multicast networks. We have demonstrated that by solving this problem it is feasible to obtain an overlay multicast tree with globally optimized latency and out-degree. We have presented two efficient heuristic methods, which have been shown by simulation experiments to be scalable for large size network topologies and yield near-optimal routing performance from the global perspective of overlay multicast networks.

The GOLD spanning tree problem proposed in this paper is based on a single-source multicast service model, since the driving multicast applications to date are one-to-many including real-time media streaming and file transfers [7]. In our future work, we plan to modify the formulation of the GOLD spanning tree problem and extend the methodologies developed in this paper to cover the multi-source multicast scenarios in overlay multicast networks. We are also interested in designing efficient restoration algorithms to provide resilience capabilities for overlay multicast trees [14].

Acknowledgments

This project is supported by Australian Research Council (ARC) Discovery Grant DP0557519.

References

[1] M. Abramowitz and I. A. Stegun, editors. *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables.* Dover, New York, 1972.

[2] T. Bäck. Optimal mutation rates in genetic search. In *Proc. 5th Int. Conf. Genetic Algorithms*, pages 2–8, San Mateo, CA, 1993.

[3] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. In *Proc. IEEE INFOCOM 03*, volume 2, pages 1521–1531, San Francisco, CA, USA, Mar. 2003.

[4] E. Brosh and Y. Shavitt. Approximation and heuristic algorithms for minimum delay application-layer multicast trees. In *Proc. IEEE INFOCOM 04*, volume 4, pages 2697–2707, Hong Kong, China, Mar. 2004.

[5] K. Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2–4):311–338, Jun. 2000.

[6] R. Diestel. *Grapy Theory.* Springer-Verlag, New York, 3rd edition, 2005.

[7] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment issues for the IP multicast service and architecture. *IEEE Network*, 14(1):78–88, Jan./Feb. 2000.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, San Francisco, 1979.

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

[10] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.

[11] ILOG. http://www.ilog.com/, 2006.

[12] L. Lao, J.-H. Cui, and M. Gerla. Multicast service overlay design. In *Proc. SPECTS 05*, Philadelphia, PA, USA, Jul. 2005.

[13] L. Lao, J.-H. Cui, and M. Gerla. TOMA: a viable solution for large-scale multicast service support. In *Proc. IFIP NETWORKING 05*, pages 906–917, Waterloo, Ontario, Canada, May 2005.

[14] W. Lau, S. Jha, and S. Banerjee. Efficient bandwidth guaranteed restoration algorithms for multicast connections. In *Proc. IFIP NETWORKING 05*, pages 1005–1017, Waterloo, Ontario, Canada, May 2005.

[15] S. C. Narula and C. A. Ho. Degree-constrained minimum spanning trees. *Computers and Operations Research*, 7:239–249, 1980.

[16] G. R. Raidl and B. A. Julstrom. A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem. In *Proc. ACM Symp. Applied Computing*, pages 440–445, Como, Italy, Mar. 2000.

[17] G. R. Raidl and B. A. Julstrom. Edge sets: an effective evolutionary coding of spanning trees. *IEEE Trans. Evol. Comput.*, 7(3):225–239, Jun. 2003.

[18] F. Safaei, I. Ouveysi, M. Zukerman, and R. Pattie. Carrier-scale programmable networks: wholesaler platform and resource optimization. 19(3):566–573, Mar. 2001.

[19] S. Sahni and T. Gonzalez. P-complete approximation problems. *J. ACM*, 23(3):555–565, Jul. 1976.

[20] S. Y. Shi and J. S. Turner. Routing in overlay multicast networks. In *Proc. IEEE INFOCOM 02*, volume 3, pages 1200–1208, New York, NY, USA, Jun. 2002.

[21] S.-M. Soak, D. W. Corne, and B.-H. Ahn. The edge-window-decoder representation for tree-based problems. *IEEE Trans. Evol. Comput.*, 10(2):124–144, Apr. 2006.

[22] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to model an internetwork. In *Proc. IEEE INFOCOM 96*, volume 2, pages 594–602, San Francisco, CA, USA, Mar. 1996.

[23] Q. Zhang and Y.-W. Leung. An orthogonal genetic algorithm for multimedia multicast routing. *IEEE Trans. Evol. Comput.*, 3(1):53–62, Apr. 1999.