

# Minimizing Average Path Cost in Colored Trees for Disjoint Multipath Routing

Ravi Balasubramanian and Srinivasan Ramasubramanian  
 Department of Electrical and Computer Engineering  
 University of Arizona, Tucson, AZ 85721  
 E-mail: {ravib, srini}@ece.arizona.edu

**Abstract**—Multi-path routing (MPR) is an effective strategy to achieve robustness, load balancing, congestion reduction, and increased throughput by transmitting data over multiple paths. Disjoint multi-path routing (DMPR) requires the multiple paths to be link- or node-disjoint. Implementation of both MPR and DMPR poses significant challenges in obtaining loop-free multiple (disjoint) paths and effectively forwarding the data over the multiple paths, the latter being significant in data-gram networks.

In this paper, we develop a disjoint multipath routing strategy using colored trees with an objective to minimize the total cost of the routing paths in a network. Two trees, namely red and blue, rooted at a given drain is formed. We demonstrate through extensive simulations that the developed technique is extremely effective in optimizing the average cost of the paths. In addition, we also observe that the developed approach minimizes the average minimum (minimum of the two paths) cost, which is lower than that obtained by earlier algorithms. The colored tree approach simply doubles the size of the routing table when two link- or node-disjoint paths to a specific node is needed.

## I. INTRODUCTION

Multipath routing (MPR) is an effective strategy to achieve robustness, load balancing, congestion reduction, lower power consumption and increase throughput by transmitting data over multiple paths. In general, the multiple paths from a source to a destination may have common links or nodes as long as the shared links (or nodes) have sufficient resources. The shared link (or node) must also be highly reliable, as its failure will render the multiple paths useless.

Alternatively, multiple paths can be selected to be link- or node-disjoint. In this case, the MPR approach referred to as *disjoint multipath routing* (DMPR). Having multiple disjoint paths is beneficial because wireless networks are prone to route breaks resulting from fading environment, signal interference and packet collision. The various approaches for finding multiple disjoint path routing are reviewed.

### A. Approaches for Disjoint Multipath Routing

The approaches for DMPR may be classified into three broad categories depending on the information used for forwarding individual packets.

**Explicit forwarding.** In this approach, every node may find two link/node-disjoint paths to the drain independently through known disjoint-path algorithms [1]. Once the disjoint paths are obtained, the path information is embedded in every packet. An intermediate node then routes individual packets based on the path information in the packet. No routing

tables are needed at intermediate nodes. MPR algorithms based on Dynamic source Routing fall under this category. The embedding of path information in packet header is a significant overhead and its associated routing cost limits the application of this approach.

**Source-based forwarding.** The communication cost of explicit routing may be reduced by transferring the disjoint path information embedded in the packets to the intermediate nodes along the paths. Every node then maintains a routing table and forwards a packet based on the destination and source addresses provided in the packet header. Figure 1 shows two node-disjoint paths computed by nodes A and B. Consider the situation at node C. This node receives packets from one incoming link and forwards them to two distinct outgoing links. The forwarding decision then has to be based on the source address. The routing table at node C will have two entries for the drain, corresponding to the two sources.

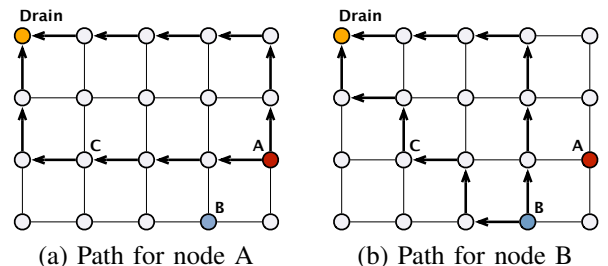


Fig. 1. Node-disjoint paths from nodes A and B to the drain computed independently by each node.

If the average sum of the lengths of two paths from a source to drain is  $\overline{H}_d$ , then the total routing table entries in the network for a given drain is  $(N - 1)\overline{H}_d$ . Note that a typical single path routing would only have a total of  $(N - 1)$  routing table entries for a particular drain. The sum of any two disjoint paths between two node pairs must be at least of length three, hence  $(N - 1)\overline{H}_d \geq 3(N - 1)$ . The value of  $H_d$  may range anywhere from  $O(1)$  (for a fully connected network) to  $O(N)$  (for a sparsely connected network). Given that there may be  $O(N)$  entries for each drain, in the worst-case, routing a packet based on destination and source address will increase the table lookup time significantly.

**Colored trees.** To reduce the routing table overhead, hence reduce lookup time, this paper develops a novel multipath routing strategy called Color Trees (CT). Every node in the

network has two preferred neighbors to reach the drain: *red* and *blue*. A packet transmitted from a source is marked with one of the two colors. An intermediate node that receives the packet forwards it to its preferred neighbor based on the color of the packet. Thus, the routing table at a node has only two entries (for every drain node). The network may be viewed as two trees (red and blue) rooted at the drain where the paths are directed towards the drain. The two trees have the property that the two paths from a given source to the drain on the two trees are link/node-disjoint. The number of routing table entries for a drain in a network with  $N$  nodes is  $2(N - 1)$ . Hence, the colored tree approach reduces the routing table by a factor of  $\frac{H_d}{2}$ . Figure 2 shows the colored trees for the example network.

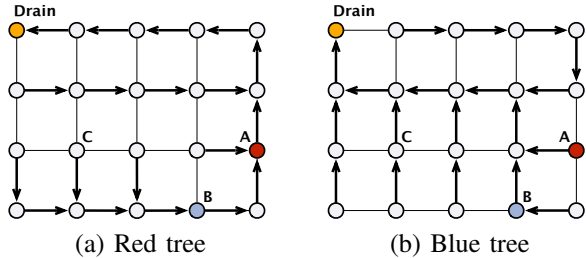


Fig. 2. Two trees obtained such that the paths from a node to the drain on the two trees are node-disjoint.

**Prior work.** The works most relevant to our approaches are [2], [3], [4] and [5]. In [2], an  $O(|\mathcal{N}|^3)$  algorithm for multi-tree construction was developed based on path augmentation. In [3], Chen et al. introduced the concept of quality of service of recovery scheme and developed an algorithm for multi-tree construction with an objective to minimize path cost in the primary path alone. Zhang et al. [4] later developed an  $O(|\mathcal{L}|)$  algorithm based on depth first search numbering. In [5], a distributed algorithm was developed based on depth first search numbering. The authors in [5] developed the concept of “generalized” low point value and showed that their approach is effective in minimizing the average path cost compared to the traditional lowpoint concepts employed in [4]. A drawback common to all the above algorithms is that they do not explicitly minimize the average path cost, which is an important metric when multiple paths are employed simultaneously.

**Motivation.** In networks where multipath routing is employed for improving throughput or load balancing, the multipath paths are used simultaneously. Data is split across the two paths. In such cases, the objective of colored tree construction is to minimize the sum of the two path costs, averaged over all the nodes. When multipath routing is employed for robustness, every packet is transmitted over both the paths. In such a case, the end-to-end delay (assuming that the link costs reflect delay) of a packet is determined by the length of the shortest path in the two trees. Hence, the objective is to minimize the minimum path length from a source to destination, averaged over all the nodes. It is worth noting that while some nodes may use the path on the red tree as the primary path, the others might use the path on the blue tree as their primary paths.

**Contribution.** In this paper, we develop an  $O(|\mathcal{N}|^3)$  centralized algorithm that is shown to achieve better performance compared to all the earlier known techniques with respect to: (1) the average sum of the cost of the two paths; and (2) the average minimum path cost. The centralized algorithm developed in this paper is aimed at networks that already employ link-state protocol, hence the entire network topology information is already available at every node. We show that the algorithm developed in this paper achieves a reduction of around 50% compared to the earlier algorithms for the metrics discussed above.

The rest of the paper is organized as follows. Section II provides a formal definition of the problem. Section III describes the related work in developing heuristic solutions to obtain the two trees that were developed in the context of robust multicasting. Section IV describes two algorithms that attempt at minimizing the average cost of red and blue paths; one developed by Chen et al. while the other is our contribution in this paper. Section V provides the performance results obtained using the two heuristic approaches and compares them to the optimal results obtained by solving the ILP. Section VI concludes the paper.

## II. PROBLEM FORMULATION

Consider a network  $\mathcal{G}(\mathcal{N}, \mathcal{L})$  composed of a set of nodes  $\mathcal{N}$  and a set of links  $\mathcal{L}$ . The links are assumed to be bi-directional, if one exists. The terminology of *arc* is used to refer to a directed link between two nodes. An arc from node  $i$  to  $j$  is represented as  $i \rightarrow j$ . For wireless networks, the set of links indicates that two nodes are in the receiving range of each other. Let  $C_{ij}$  refer to the cost of the arc  $i \rightarrow j$ . The cost is assumed to be symmetric,  $C_{ij} = C_{ji}$ . The cost of a link may reflect many parameters of interest, such as propagation delay, minimum transmission power required for receiving the signal with a pre-defined signal-to-noise ratio (SNR). The colored tree construction for link-disjoint paths is referred to as CT-LD and that for node-disjoint paths as CT-ND.

**Problem definition.** Given a network  $\mathcal{G}(\mathcal{N}, \mathcal{L})$ , where  $\mathcal{N}$  denotes the set of nodes, a drain node  $d \in \mathcal{N}$ , and  $\mathcal{L}$  denotes the set of links, construct colored trees  $\mathcal{R}$  and  $\mathcal{B}$  (referred to as red and blue trees, respectively) rooted at the drain  $d$  that minimizes the total sum of red and blue path cost from every source to the drain such that the CT-LD and CD-ND versions of the problem satisfy the link-disjoint and node-disjoint path constraints, respectively.

Let  $\mathcal{P}_{sd}^{\mathcal{R}}$  and  $\mathcal{P}_{sd}^{\mathcal{B}}$  denote the paths from node  $s$  to drain  $d$  on trees  $\mathcal{R}$  and  $\mathcal{B}$ , respectively.

**Link-disjoint path constraint:** If the path from  $s$  to  $d$  on the red tree traverses  $i \rightarrow j$ , then the path from  $s$  to  $d$  on the blue tree traverses neither  $i \rightarrow j$  nor  $j \rightarrow i$ , i.e.  $\forall s \in \mathcal{N} \setminus \{d\}; \forall i, j \in \mathcal{N}$

$$i \rightarrow j \in \mathcal{P}_{sd}^{\mathcal{R}} \Rightarrow (i \rightarrow j \notin \mathcal{P}_{sd}^{\mathcal{B}}) \wedge (j \rightarrow i \notin \mathcal{P}_{sd}^{\mathcal{B}})$$

**Node-disjoint path constraint:** If the path from  $s$  to  $d$  on the red tree traverses node  $i$ , then the path from  $s$  to  $d$  on the

blue tree does not traverse node  $i$ , i.e.  $\forall s \in \mathcal{N} \setminus \{d\}$  and  $i \in \mathcal{N} \setminus \{s, d\}$

$$i \in \mathcal{P}_{sd}^{\mathcal{R}} \Rightarrow (i \notin \mathcal{P}_{sd}^{\mathcal{B}})$$

A solution to the CT-LD (CT-ND) problem exists whenever a network is two edge-connected (node-connected). The colored tree construction problem maybe modeled as an Integer Linear Program(ILP). The readers are referred to [6] for a detailed description of the formulation. The ILP formulation helps in identifying optimal solution for small networks. However, its application to large networks is impractical due to its prohibitively large solution time.

### III. RELATED WORK

Two centralized approaches have been proposed in the literature for multicasting, one based on  $s$ - $t$  numbering and the other on generalized path augmentation.

#### A. The $s$ - $t$ Numbering Approach

Given a graph and an edge  $s$ - $t$ , an  $s$ - $t$  numbering of the graph is a numbering of the  $N$  vertices of the graph such that the following two properties are obeyed:

- Node  $s$  has the number 1 and node  $t$  has the number  $N$ .
- Every node  $i$  other than  $s$  and  $t$  has a number  $v_i$ , such that  $1 < v_i < N$ . In addition, every node  $i$  except  $s$  and  $t$  has at least one neighbor with a higher and lower  $s$ - $t$  number.

Itai et al. [7] employ the  $s$ - $t$  numbering technique to obtain a solution to the node disjoint problem. The readers are referred to [7] for a detailed description of the algorithm. The major drawback of the  $s$ - $t$  approach is that it is applicable only to networks that are two-node-connected. Hence, it is not possible to obtain a solution to the link disjoint problem using this approach. Secondly this approach does not minimize the cost of the paths in the network.

#### B. The Generalized Path Augmentation Approach

The path augmentation algorithm starts by choosing an arbitrary directed cycle  $(d, v_1, \dots, v_k, d)$  of  $\mathcal{G}$  with at least three nodes ( $k \geq 2$ ). If this cycle does not include all nodes in the graph, then a path that starts and ends on the cycle and that passes through at-least one node not on the cycle is chosen for augmentation. The algorithm continues with path augmentation until all the nodes in the network are considered.

Medard et al. [2] developed an algorithm that selects a cycle and successive paths at random. Chen et al. [3] presented a generalized version of the approach by specifying certain criteria for selecting paths for augmentation, which we refer to as the XCT algorithm.

The algorithm developed in this paper is based on the path augmentation technique. Therefore, the generalized algorithm by Chen et al. for constructing multi-tree with node-disjoint path constraint is described in detail. The algorithm by Chen et al. is based on partial ordering of nodes in the network. Recall that  $\mathcal{R}$  and  $\mathcal{B}$  are the two trees to be formed. The partial order  $\prec$  on the nodes of the graph  $\mathcal{G}$  using the blue tree  $\mathcal{B}$  is

defined in the following way. If  $u \rightarrow v \in \mathcal{B}$ , then  $u$  precedes  $v$  in the partial order, represented as  $u \prec v$ . The algorithm described in [3] employs partial order on both red and blue trees, however the explanation here has been simplified based on the partial order on blue trees alone. The partial ordering satisfies the transitive relationship, i.e. if  $u \prec v \prec w$ , then  $u \prec w$ .

The algorithm for constructing the two trees satisfying node-disjoint path constraint follows four steps:

- 1) Initialize  $\mathcal{R}$  and  $\mathcal{B}$  to contain the root node  $d$  only. Initialize the partial order on the edges to be the empty set.
- 2) Find a cycle  $[d, v_1, \dots, v_k, d]$ . Let  $v_k \rightarrow v_{k-1} \rightarrow \dots \rightarrow v_1 \rightarrow d$  be the red chain and  $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k \rightarrow d$  be the blue chain. Add the blue chain to  $\mathcal{B}$  and red chain to  $\mathcal{R}$ . Use the precedence relation  $v_1 \prec v_2 \prec \dots \prec v_k \prec d$ .
- 3) If  $\mathcal{B}$  spans all the nodes in  $\mathcal{G}$ ; stop.
- 4) Find a path  $[x, v_1, \dots, v_k, y]$  connecting two distinct nodes  $x$  and  $y$  on  $\mathcal{B}$  and  $k$  ( $\geq 1$ ) nodes not on  $\mathcal{B}$ , such that  $x \prec y$ . Let  $v_k \rightarrow v_{k-1} \rightarrow \dots \rightarrow v_1 \rightarrow x$  be the red chain and  $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k \rightarrow y$  be the blue chain. Add the blue chain to  $\mathcal{B}$  and the red chain to  $\mathcal{R}$ . Go to Step 3. Use the precedence relation  $x \prec v_1 \prec v_2 \prec \dots \prec v_k \prec y$ .

The above algorithm may be applied to the link-disjoint case by relaxing the condition in Step 4 that node  $x$  and node  $y$  have to be distinct and maintaining partial ordering of edges instead of nodes. The algorithm is guaranteed to obtain two trees satisfying link-disjoint (node-disjoint) constraint if the network is two-edge-connected (two-node-connected).

**Observations:** The algorithm we propose stems from certain key observations made from the centralized path augmentation approach. Consider the example network shown in Figure 3.

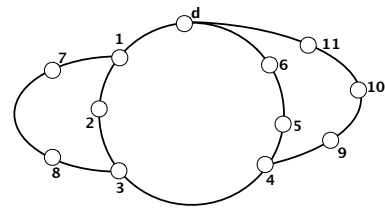


Fig. 3. Example network to illustrate the observations made in the centralized algorithm to develop a distributed algorithm.

The centralized algorithm first selects a cycle. Let the cycle formed be  $\{d, 1, 2, 3, 4, 5, 6, d\}$ . Considering one particular direction in the cycle (corresponding to the blue tree), the partial ordering of the nodes would be  $\{1, 2, 3, 4, 5, 6, d\}$ , implying  $1 \prec 2 \prec 3 \prec 4 \prec 5 \prec 6 \prec d$ . There are two options for selecting a path for augmenting:  $1-7-8-3$  or  $4-9-10-11-d$ . The algorithm by Medard et al. [2] selects a path at random while that by Chen et al. [3] selects a path based on a certain metric. Without loss of generality, assume that the path  $4-9-10-11-d$  is chosen for augmenting. The partial

ordering of these paths must be such that: (1) node 4 precedes node 9 ( $4 \prec 9$ ); node 11 precedes node  $d$  ( $11 \prec d$ ); and nodes 9, 10, and 11 must appear in the same order as in the path ( $9 \prec 10 \prec 11$ ). However, it is to be noted that there is no explicit ordering between the nodes 9, 10, and 11 in the new path and the nodes 5 and 6 in the old path. Some of the valid global ordering of the nodes that satisfy the above partial order are shown below.

- 1)  $\{1, 2, 3, 4, 5, 6, 9, 10, 11, d\}$
- 2)  $\{1, 2, 3, 4, 9, 10, 11, 5, 6, d\}$
- 3)  $\{1, 2, 3, 4, 9, 5, 10, 6, 11, d\}$

It is the choice of selecting one of these global orderings that distinguishes the various approaches. Medard et al. [2] algorithm selects one particular global ordering at the end of each path augmentation approach; while Chen et al. [3] approach does not explicitly impose a particular global ordering. However, their path augmentation step includes a constraint on the blue (or red) path explicitly, which in turn dictates a global ordering.

#### IV. MINIMIZING THE AVERAGE PATH COST

In this section, we consider minimizing the average path cost through heuristic approaches. One of the algorithms developed in [3] is to minimize the average path cost on the working tree. The working tree is chosen either as the red or the blue tree (whichever yields the minimum cost). We refer to the algorithm developed in [3] as the XCT algorithm and the steps are shown in Figure 4 for the CT-ND case. To obtain a solution to the CT-LD case, nodes  $x$  and  $y$  defined in the XCT algorithm need not be distinct.

---

##### XCT Algorithm:

- 1) Initialize  $\mathcal{R}$  and  $\mathcal{B}$  to contain only the root node  $d$ .
  - 2) Find a cycle  $[d, v_1, v_2, \dots, v_k, d]$  in the network such that the cost of the cycle minus the cost of its last edge  $[v_k \rightarrow d]$  is the minimum among all such cycles. Let  $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k \rightarrow d$  be the blue chain and  $v_k \rightarrow v_{k-1} \rightarrow \dots \rightarrow v_1 \rightarrow d$  be the red chain. Add the red chain and blue chain to  $\mathcal{R}$  and  $\mathcal{B}$  respectively and compute the red path length and blue path length for each node of the selected path. Use the precedence relation  $v_1 \prec v_2 \prec \dots \prec v_k \prec d$ .
  - 3) If  $\mathcal{B}$  spans all the nodes in  $\mathcal{G}$ , stop.
  - 4) Find an augmenting path  $[x, v_1, v_2, \dots, v_k, y]$  such that node  $x$  and node  $y$  (such that  $x \prec y$ ) are in the blue chain and the cost of red path of  $x$  plus the  $[x - v_k]$  cost is minimum among all such paths. Let  $v_k \rightarrow \dots \rightarrow v_1 \rightarrow x$  be the red chain and  $v_1 \rightarrow \dots \rightarrow v_k \rightarrow y$  be the blue chain. Use the precedence relation  $x \prec v_1 \prec v_2 \prec \dots \prec v_k \prec y$ . Go to Step 3.
- 

Fig. 4. Steps involved in the XCT algorithm

The XCT algorithm attempts to minimize the cost of the red path (assumed to be working) by considering the only the cost of red path during augmentation. While this is intuitively true, neglecting the cost of the last link (one that will be included in the blue path) directly affects the average path cost when blue tree is also included. In addition, we also show that the approach developed in this paper minimizes the red tree even

when we attempt to minimize the average cost of both the trees.

We refer to the algorithm developed in this paper as the *BR Algorithm* and is described as shown in Figure 5 for the node disjoint case. The key difference between XCT and BR algorithms is the selection of the path for augmentation.

---

##### BR Algorithm:

- 1) Initialize  $\mathcal{R}$  and  $\mathcal{B}$  to include only the drain  $d$ .
  - 2) Compute the least cost cycle  $[d, v_1, v_2 \dots v_k, d]$  in the network. Let  $v_k \rightarrow \dots \rightarrow v_1 \rightarrow s$  be the red chain and  $v_1 \rightarrow \dots \rightarrow v_k \rightarrow s$  be the blue chain. Add the blue chain to  $\mathcal{B}$  and red chain to  $\mathcal{R}$  and compute the red path length and the blue path length for each node of the selected path. Use the precedence relation  $v_1 \prec v_2 \prec \dots \prec v_k \prec d$ .
  - 3) If  $\mathcal{B}$  spans all the nodes in  $\mathcal{G}$ , stop.
  - 4) Find an augmenting path  $[x, v_1, v_2, \dots, v_k, y]$  such that node  $x$  and node  $y$  ( $x \prec y$ ) are in the blue chain and the cost of the augmenting path plus the cost of red count of  $x$  and the blue count of  $y$  is minimum among all paths. Let  $v_k \rightarrow \dots \rightarrow v_1 \rightarrow x$  be the red chain and  $v_1 \rightarrow \dots \rightarrow v_k \rightarrow y$  be the blue chain. Go to Step 3.
- 

Fig. 5. Steps involved in the BR algorithm.

To obtain a solution to the link disjoint case, nodes  $x$  and  $y$  described in the BR algorithm need not be distinct. The algorithm terminates if the graph is 2-vertex connected and the worst case running time of the algorithm is  $O(|\mathcal{N}|^2(|\mathcal{L}| + |\mathcal{N}| \log |\mathcal{N}|))$ .

**Time Complexity.** The least-cost cycle required in Step 2 of the BR algorithm may be computed as follows. For a link  $\ell$  that connects the drain  $d$  and another node  $v$ , compute the shortest path from  $d$  to  $v$  after removing link  $\ell$ . A cycle from  $d$  through  $v$  is obtained by adding the link  $\ell$  to the shortest path, whose cost is defined as the sum of all the links in the cycle. We may construct cycles similarly through all the neighbors of the drain and select the cycle with minimum cost. The complexity of computing the shortest path between two nodes is  $O(|\mathcal{L}| + |\mathcal{N}| \log |\mathcal{N}|)$ . The number of neighbors of drain is  $O(|\mathcal{N}|)$ . Therefore, the complexity of Step 2 is  $O(|\mathcal{N}| |\mathcal{L}| + |\mathcal{N}|^2 \log |\mathcal{N}|)$ .

The complexity computing the least-cost augmenting path in Step 4 of the BR algorithm is the same as that of step 2. Step 4 is executed  $O(|\mathcal{N}|)$  times as every execution of Step 4 adds at least one new node to the tree. Therefore, the complexity of Step 4, which defines the complexity of the BR algorithm, is  $O(|\mathcal{N}|^2 |\mathcal{L}| + |\mathcal{N}|^3 \log |\mathcal{N}|)$ .

As the XCT and BR algorithms differ only in the definition of cost of the path chosen for augmentation, the complexity of the XCT and BR algorithms are the same. The algorithms proposed is based on the path augmentation technique [2]. Hence, the proof of correctness of the algorithm follows from [2] and is not repeated in this paper due to space constraints.

##### A. Illustrative Example

We illustrate the difference between the XCT and BR algorithms using the example network shown in Figure 6.

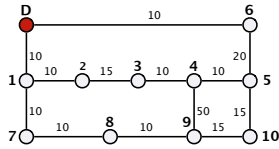


Fig. 6. Example network.

The XCT algorithm selects the cycle such that the cycle length minus its last edge is minimum. In the example network, the algorithm selects the cycle  $(D, 6, 5, 4, 3, 2, 1, D)$  with  $(6 \rightarrow 5 \rightarrow \dots \rightarrow 1 \rightarrow D)$  forming the blue tree and  $(1 \rightarrow 2 \rightarrow \dots \rightarrow 6 \rightarrow D)$  forming the red tree. In the next iteration, the XCT algorithm selects an augmenting path  $(x, c_1, c_2, \dots, c_k, y)$  such that the red cost of node  $x$  plus the augmenting path length minus the last edge  $[c_k, y]$  is minimum. There are three possible augmenting paths:

- 1)  $(5 - 10 - 9 - 4)$
- 2)  $(4 - 9 - 8 - 7 - 1)$
- 3)  $(5 - 10 - 9 - 8 - 7 - 1)$

The algorithm selects the first path  $(5 - 10 - 9 - 4)$  based on the constraint stated above.  $(9 \rightarrow 10 \rightarrow 5)$  forms the red tree and  $(10 \rightarrow 9 \rightarrow 4)$  forms the blue tree. In the next iteration, the algorithm selects the augmenting path  $(9 - 8 - 7 - 1)$  with  $(7 \rightarrow 8 \rightarrow 9)$  forming the red tree and  $(8 \rightarrow 7 \rightarrow 1)$  forming the blue tree. The algorithm terminates after the third iteration as all the nodes have been covered.

In the BR algorithm, we choose the shortest cycle  $(D, 6, 5, 4, 3, 2, 1, D)$  with  $(6 \rightarrow 5 \rightarrow \dots \rightarrow 1 \rightarrow D)$  forming the blue tree and  $(1 \rightarrow 2 \rightarrow \dots \rightarrow 6 \rightarrow D)$  forming the red tree. In the next iteration, our algorithm chooses an augmenting path  $(x, v_1, v_2, \dots, v_k, y)$  such that the sum of the red cost of node  $x$ , the augmenting path cost and the blue cost of node  $y$  is minimum. There are three possible augmenting paths as in XCT algorithm. However our algorithm selects the third path namely  $(5 - 10 - 9 - 8 - 7 - 1)$  as the red path cost of node 5, the augmenting path cost  $(5 - 10 - 9 - 8 - 7 - 1)$  and the blue path cost of node 1 is minimum among all augmenting paths.  $(7 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 5)$  forms the red tree and  $(10 \rightarrow 9 \rightarrow 8 \rightarrow 7 \rightarrow 1)$  forms the blue tree. The algorithm terminates after the second iteration, as all the nodes have been covered. Note that the difference in the XCT and BR algorithms lie in the paths chosen for augmentation. The trees formed by the two algorithms are shown in Figures 7 and 8.

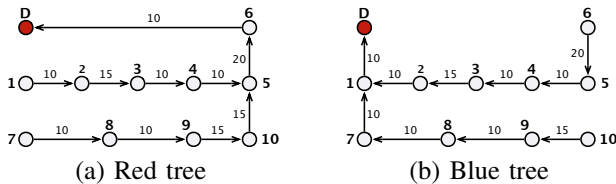


Fig. 7. Red and blue trees obtained using the BR algorithm.

We note that the red tree formed in both are the same (which need not be the case for all networks), while the blue trees formed in both the algorithms are different. The total path cost

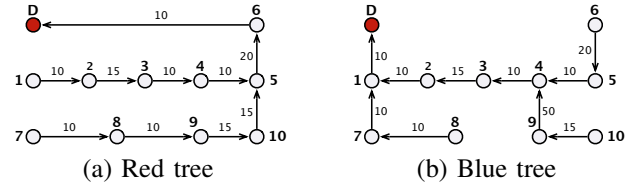


Fig. 8. Red and blue trees obtained using the XCT algorithm.

for all the nodes in the blue tree under the XCT algorithm is 495 while that under the BR algorithm is 385.

## V. PERFORMANCE EVALUATION

The effectiveness of the BR algorithm is analyzed by comparing the performance with the XCT algorithm and the optimal solution obtained by solving the ILP. As ILP solution times are prohibitively large for large networks, we compare the performance results of the heuristic approaches to the optimal solution only in small networks, such as NSFNET, ARPANET, NJ-LATA, and  $5 \times 5$  mesh networks. Due to space constraints the topologies of NSFNET, ARPANET, and NJ-LATA networks are not shown in this paper, the readers are referred to [8].

For each network, the simulations were performed ten times with random cost for links chosen uniformly between 10 and 50. The average of the ten simulation runs are presented here. The performance results presented here correspond to the CT-LD case. Similar results were obtained for the CT-ND case as well.

### A. Results and Discussion

Table I shows the total cost of red and blue paths from all the nodes to the drain under the XCT and BR algorithms along with the optimal results obtained by solving the ILP using CPLEX 8.0 solver [9]. We observe that the performance of the BR algorithm is closer to optimal compared to the XCT algorithm.

TABLE I  
COMPARISON OF THE SUM OF RED AND BLUE PATH COSTS OBTAINED USING THE XCT ALGORITHM, BR ALGORITHM, AND ILP.

Networks	XCT	BR	Optimal
NJLATA	1943	1613	1472
NSFNET	2322	2004	1885
ARPANET	3863	3607	3529
5x5 Mesh	6549	6104	5994

Table II show the performance of the XCT and BR algorithms over random topologies formed with 20, 50, 100, and 200 nodes. We observe that the BR algorithm achieves up to 42.36% reduction in the total cost of red and blue paths. Every node has a path to the drain on the red and blue trees. We compute the average of the minimum cost of the two paths over all the nodes (tabulated as average minimum path cost) and the average of the maximum of the two paths over all the nodes (tabulated as average maximum cost path).

TABLE II  
PERFORMANCE COMPARISON OF THE XCT AND BR ALGORITHMS.

Network	Average Red Path Cost		Average Blue Path Cost		Average Minimum Path Cost		Average Maximum Path Cost		Average Total Cost		Decrease in Average Total Cost	Decrease in Average Minimum Cost
	XCT	BR	XCT	BR	XCT	BR	XCT	BR	XCT	BR		
20-Node Random	889.2	710.5	656.6	483.2	544.4	391.9	994.4	739.4	1545.8	1193.7	22.77%	28.01%
50-Node Random	2277.4	1513.1	1404.8	1056.5	1150.3	1028.1	2531.9	1541.5	3682.2	2569.6	30.21%	10.6%
100-Node Random	4554.3	2664.5	2900.7	1714.2	2380.5	1826.1	5071.5	2522.6	7452	4378.7	41.24%	23.28%
200-Node Random	7961.5	4401.4	4971.1	3051.8	3576.1	2998.5	9356.5	4454.7	12932.6	7453.2	42.36%	16.15%

TABLE III  
PERFORMANCE COMPARISON OF THE BR ALGORITHM AND DISTRIBUTED ALGORITHM DEVELOPED IN [5] (REFERRED TO AS THE RHK ALGORITHM).

Network	Average Red Path Cost		Average Blue Path Cost		Average Minimum Path Cost		Average Maximum Path Cost		Average Total Cost		Decrease in Average Total Cost	Decrease in Average Minimum Path Cost
	RHK	BR	RHK	BR	RHK	BR	RHK	BR	RHK	BR		
10-Node Random	493.77	405.2	990.89	369.8	455.77	305.6	1020.89	469.4	1484.66	775	47.79%	32.95%
20-Node Random	932.84	600.2	2012.64	508.2	848.64	469.4	2096.84	699	2945.48	1108.4	62.36%	44.68%
50-Node Random	2253	1436.6	4592	1014.2	2037.5	951.8	4807.5	2096.84	6845	2450.8	64.12%	53.29%
100-Node Random	4506.5	3059	9184	1661	4075.5	1656	9615	4807	13690.5	4720	65.52%	59.36%

The rationale behind computing this metric is that if the cost reflects delay along the links, this metric reflects the end-to-end delay experienced when the same message is transmitted over both the trees (for fault tolerance purposes). The average minimum path cost reflects the delay in networks that achieve robustness. The average minimum path cost or delay was found to be significantly lower than that obtained by the XCT algorithm. Thus the heuristic is more effective in minimizing the end-to-end delay. The heuristic approach completed well under 10 ms for all scenarios. We also observe that the average red path cost under BR approach is lower than that obtained using the XCT approach, indicating optimizing over one tree alone is not an effective strategy.

In [5], the distributed algorithm is shown to be more effective than the algorithm developed in [4], hence we compare the algorithm developed in this paper with that developed in [5], referred to as the RHK algorithm. Table III shows the performance of the RHK and BR algorithms over random topologies formed with 10, 20, 50, and 100 nodes. We observe that the BR algorithm achieves around 60% percent reduction in the average total cost and average minimum cost.

## VI. CONCLUSION

This paper develops an effective algorithm for finding disjoint multipath routing with an objective to minimize the total average path cost. It is also effective in reducing the average minimum routing path delay of the network than previously known algorithms. This algorithm is useful especially in networks that require quality of service for delays. The algorithm achieves upto 50% reduction in total path cost and minimum path cost. The tradeoff, however being an increase in com-

putational complexity. This algorithm guarantees a link/node-disjoint path if the network is two link/node-connected.

## ACKNOWLEDGMENT

The research presented in this paper is supported by National Science Foundation under grants 0325979, 0435490, and 0333046.

## REFERENCES

- [1] R. Bhandari, *Survivable Networks: Algorithms for Diverse Routing*, Kluwer Academic Publishers, 1999.
- [2] M. Medard, R.A. Barry, S.G. Finn, and R.G. Gallager, "Redundant trees for preplanned recovery in arbitrary vertex- redundant or edge redundant graphs," *IEEE/ACM Transactions on Networking*, vol. 7, no. 5, pp. 641–652, October 1999.
- [3] G. Xue, L. Chen, and K. Thulasiraman, "Quality-of-service and quality-of-protection issues in preplanned recovery schemes using redundant-trees," *IEEE Journal on Selected Areas in Communication*, vol. 21, no. 8, pp. 1332–1345, October 2003.
- [4] W. Zhang, G. Xue, J. Tang, and K. Thulasiraman, "Linear time construction of redundant trees for recovery schemes enhancing QoP and QoS," in *Proceedings of IEEE INFOCOM*, Miami, FL, USA, March 2005, pp. 2702–2710.
- [5] S. Ramasubramanian, M. Harkara, and M. Krunz, "Distributed linear time construction of colored trees for disjoint multipath routing," in *Proceedings of IFIP Networking*, Coimbra, Portugal, May 2006, pp. 1026–1038.
- [6] S. Ramasubramanian, H. Krishnamoorthy, and M. Krunz, "Disjoint multipath routing using colored trees," *Technical Report, University of Arizona*, November 2005.
- [7] A. Itai and M. Rodeh, "The multi-tree approach to reliability in distributed networks," in *Foundations of Computer Science '84*, October 1984.
- [8] A. Chandak and S. Ramasubramanian, "Dual-link failure resiliency through backup link mutual exclusion," in *Proceedings of IEEE International Conference on Broadband Networks*, Boston, MA, USA, October 2005, pp. 258–267.
- [9] CPLEX Solver, <http://www.cplex.com>.