CIT 2002

Proceedings of
Fifth International Conference on

Information Technology

December 21–24, 2002

Editors
Sridhar Iyer
Sagar Naik

Organized by
• Orissa Information Technology Society (OITS), Bhubaneswar

QoS Routing in Communication Networks

Krishnaiyan Thulasiraman[1] and Ravi Ravindran[2]

[1]University of Oklahoma, Norman, USA

[2]Nortel Networks, Ottawa, Canada

Abstract

Graph theory and combinatorial optimization techniques play an important role in the design, analysis and control of telecommunication networks. These techniques include algorithms for finding shortest paths, determining a maximum flow, finding a minimum cost tree etc. Though most problems encountered in telecommunication network studies are computationally intractable, the above graph theoretic algorithms have served as building blocks in designing efficient heuristics for these computationally hard problems. Finding a minimum cost delay-constrained routing is one such problem. In this paper we give a broad overview of the current state of the art in this area. We give a detailed exposition of two recent heuristics and provide a comparative evaluation of these heuristics with certain other approaches.

I. Introduction

Recently, there has been considerable emphasis on designing communication protocols which deliver certain performance guarantees. This has been the result of an explosive growth in applications such as digital video and audio that require stringent quality of service (QoS) guarantees. In this paper we are concerned with a study of algorithms than could be used in routing protocols for computing paths that guarantee certain QoS requirements. These heuristics can easily be incorporated into link state protocols like OSPF and ISIS where there is more need to source route using explicit paths in order to support various traffic engineering schemes. One of the promising architectures has been MPLS-based traffic engineering. Specifically, we will be studying routing mechanisms that aim at minimizing the cost of a path from a source to a destination node subject to the total delay of the path being within a certain limit. The discussion is also applicable to paths involving other link parameters which are additive (along the path). Several papers discussing different aspects of this problem have appeared in the literature [1]-[15]. We shall now review the salient features of some of these algorithms. In designing QoS routing protocols two min-cost path algorithms play an important role. They are: Dijkstra's algorithm and the Bellman-Ford-Moore (BFM) algorithm [16]. Whereas Dijkstra's algorithm is inherently sequential in nature, the BFM algorithm is amenable for an elegant distributed implementation [17]. The problem we are interested in is the design of routing protocols that satisfy multiple constraints: minimizing cost subject to the delay requirement. It has been shown that this problem is NP-complete [2]. So heuristics approaches and approximate schemes have been investigated. Chen and Nahrstedt [1] have given an overview of the several heuristics for the QoS routing problem. Jaffe [5] presented a pseudo-polynomial time algorithm for constructing paths satisfying multiple constraints. He also presented approximation algorithms and discussed several theoretical issues. Hassin [8] presented a strongly polynomial ε-approximation scheme which has formed the basis of later works [10-11] in which approximation schemes with better time complexity as well as schemes applicable to more general situations have been reported. But the approximation schemes in general have very high time complexity even for reasonable values of ε. In a recent work [12] a simple and efficient approximation scheme which improves upon Hassin's algorithm is presented. Reeves and Salama [4] discussed, perhaps the only, distributed algorithm for the delay constrained routing problem. Widyono [3] discussed several aspects of the design and evaluation of multicast and unicast algorithms for the bounded delay min-cost path problem. Though this algorithm finds the optimal paths it does not scale well with increase in the size of the network. A very efficient heuristic

based using the Lagragian relaxation method is presented in [13]. Luo, Huang, Wang, Hobbs and Munter presented in [7] a heuristic which we call the LHWHM algorithm for the bounded-delay minimum cost path problem. A heuristic based on the BFM algorithm is discussed in [14]. Several theoretical issues relating to the constrained shortest path are discussed [15].

In this paper we first present the LHWHM algorithm and examine in detail some of the factors that impact the performance of this algorithm. We then present what is called the Multi-Depth-Heuristic, which is a generalization of the LHWHM algorithm. These algorithms are adaptations of Dijkstra's algorithm. We then present the BFM-BDMCP algorithm [14] which is based on the BFM algorithm. Several issues arising in the implementation of this heuristic are discussed. The BFM-BDMCP algorithm is also amenable for distributed implementation. We conclude with an experimental comparative evaluation of these heuristics with Hassin's ε-approximation scheme and the Lagrangian relaxation based heuristic LARAC of [13].

II. Bounded Delay Min-Cost Path Problem (BDMCP)

Consider a point-to-point communication network represented as a directed graph $N = (V, E)$, where V is the set of nodes and E is the set of links in N. A link directed from node u to node v is denoted by $e = (u, v)$. Each link e is associated with two non-negative real numbers, cost $c(e)$ and delay $d(e)$. The link cost $c(e)$ may be either a monetary cost or some measure of the link's utilization. The link delay $d(e)$ is a measure of the delay a packet experiences when traversing the link e. We also specify two nodes, s and t, as the source node and the destination node, respectively. An undirected network may be viewed as a directed network with each link $e = (u, v)$ replaced by two oppositely oriented links $e_1 = (u, v)$ and $e_2 = (v, u)$. In this case $c(e_1) = c(e_2)$ and $d(e_1) = d(e_2)$. If $e = (u, v)$, then $c(e)$ and $d(e)$ are also denoted as $c_{u,v}$ and $d_{u,v}$ respectively.

We define a path P from node v_o to node v_k as an alternating sequence of distinct nodes and links such that $P(v_0, v_k) = v_0, e_1, v_1, e_2 \ldots\ldots, e_k, v_k$ where $e_i = (v_{i-1}, v_i) \in E$, for $1 \le i \le k$. The cost $c(P)$ and delay $d(P)$ of the path P are defined as :

$$c(P) = \sum_{e \in P} c(e) \text{ and}$$

$$d(P) = \sum_{e \in P} d(e)$$

Suppose we are given a real number T which serves as a measure of the maximum allowable delay on any s-t path in N, then we call an s-t path P feasible if $d(P) \le T$.

The Bounded Delay Minimum Cost Path Problem (BDMCP) is to find a feasible s-t path, which has the smallest cost. For example, in the network N in Fig.1 with delay constraint of $T = 5$, the shortest s-t path is the path $P_0 = \{s, 2, 4, t\}$. But this path does not satisfy the delay constraint of $T = 5$. The s-t paths $P_1 = \{s, 1, 4, t\}$ and $P_2 = \{s, 2, 3, t\}$ are feasible with cost 5 and cost 8 respectively. P_1 is the feasible min-cost path.

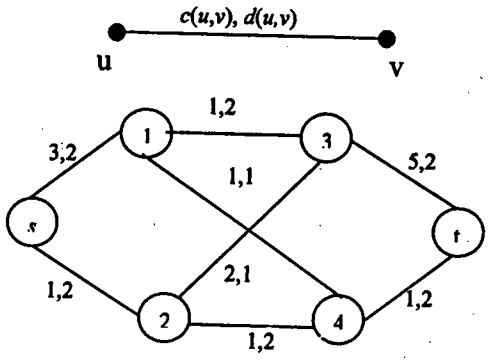


Fig.1

III. A Key Technique in the Design of Heuristics for the BDMCP Problem

The heuristics discussed in the following three sections employ a simple, yet, powerful idea discussed below.

Given a communication network with source node s and destination node t. For each node u, let us define

$D(u)$ = the delay of a minimum delay path from u to t.

In other words, $D(u)$ is the smallest delay of any path from the node u to the destination node t.

Consider a path P from node s to node u. Assume that Delay(P) $\leq T$, where T is the upper limit on allowable delay from s to t. At each step, the heuristics face the problem of extending the path P to a node x (see Figure 2). To determine the node x, the following condition is tested.

$$\text{Delay}(P) + d(u,x) + D(x) \leq T. \qquad \ldots(1)$$

If this condition is not satisfied, it can be concluded that there exists no feasible s-t path which contains P as a sub-path. Thus, in this case, the heuristics would not extend P to x. Otherwise, P will be extended to x. Incorporating this idea in efficient min-cost path algorithms results in different heuristics.

Note that satisfaction of the above condition does not guarantee the existence of a feasible s-t path. The impact of this important fact on the quality of routes produced by the heuristics will be discussed extensively in section V.

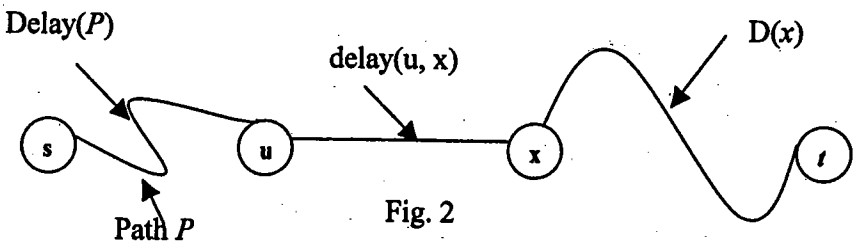


Fig. 2

IV. The LUO, HUANG, WANG, HOBBS and MUNTER (LHWHM) Algorithm

We now present and illustrate a heuristic for the Bounded Delay Min-Cost Path Problem (BDMCP) proposed by Luo, Huang, Wang, Hobbs and Munter in [7], which is to be called the LHWHM algorithm.

The LHWHM algorithm is based on the well-known Dijkstra's algorithm for the single source min-cost path problem [16]. In adapting Dijkstra's algorithm to find a sub-optimal solution, the LHWHM algorithm first computes D(u), for each node u. Recall that D (u) is the delay of a minimum delay u-t path. This computation can be done easily by applying Dijkstra's algorithm to the network N^* which is obtained by reversing the directions of all links in the given network, using the link delays instead of link costs and using node t as the source. Note that in the case of undirected graphs each link can be viewed as two oppositely directed links connecting the end nodes of the link.

The LHWHM algorithm assigns the variables CLABEL(u), PERM(u), PRED(u) and DLABEL(u) to each node u. At any step in the algorithm, the value of DLABEL(u) indicates the delay of an s-u path, and the cost of this path is given by the value of CLABEL(u). We initialize these variables as follows.

CLABEL (s) = 0 and CLABEL (u) = ∞ for all $u \neq s$;

DLABEL (u) = 0 for all u;

PERM (s) = 1 and PERM (u) = 0 for all $u \neq s$;

If PERM(u) = 1, then we call u as permanently labeled. Initially, the node s is permanently labeled. It may be recalled that in Dijkstra's algorithm, at each step at most one node is permanently labeled. Also, once a node is permanently labeled, it is not considered for labeling again. With these preliminaries, we are now ready to present the general step of the LHWHM algorithm.

The *general step* in the LHWHM algorithm :

From the nodes not yet permanently labeled pick a node, say u, which has the minimum CLABEL value. Set PERM(u) =1.Then, for each node v which is adjacent to node u and which is not permanently labeled, do:

If DLABEL(u) + $d_{u,v}$ + D(v) $\leq T$ and CLABEL(v) > CLABEL(u) + $c_{u,v}$, then set

DLABEL(v) = DLABEL(u) + $d_{u,v}$
CLABEL(v) = CLABEL(u) + $c_{u,v}$ and
PRED(v) = u.

This *general step* is repeated until PERM(t)=1.

Note the modification to the general step in Dijkstra's algorithm to arrive at the general step of the LHWHM algorithm. Basically the LHWHM algorithm, before extending the *s-u* path to a neighbor v, checks to see if there exists a feasible *s-t* path through u and v. In doing so, it uses the value of D(v) computed at the beginning of the algorithm.

V. The Multi-depth Heuristic for the BDMCP Problem

In this section we shall first discuss the factors which impact the performance of the LHWHM algorithm introduced in the previous section. Based on these factors we shall first present some improvements and variants to the basic LHWHM algorithm. We then propose a new heuristic called the ***Multi-depth Heuristic*** which is an extension of the LHWHM algorithm.

5.1 Variants of the LHWHM algorithm

We now present two factors that impact the performance of the LHWHM algorithm and suggest schemes that will provide partial remedies.

Factor 1:

Consider Fig.4. Here, P_1 and P_2 are two paths with equal costs but with delay of P_1 much higher than the delay of P_2.

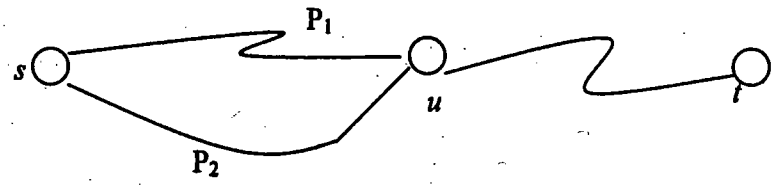


Fig 4

Suppose there is no feasible *s-t* path along P_1 and that there is an optimum feasible *s-t* path, which has P_2 as a sub-path. Assume that at some point u gets labeled along P_1 and that in some later iteration it is detected that there is no feasible *s-t* path along P_1. This means that the labeling of u along P_1 has not served any useful purpose. But it is now too late to correct the situation, because the LHWHM algorithm does not allow relabeling of u along P_2 (note that the costs of P_1 and P_2 are assumed to be equal). Hence the algorithm will not be able to look for the optimum feasible path containing P_2. Thus the LHWHM algorithm would fail to find an optimum solution. This could be remedied if we allow the LHWHM algorithm to relabel a labeled node u if it is approached by another path of equal cost but with lower delay. This requires a simple modification to the general step of the LHWHM algorithm.

Factor 2:

Consider again the situation in Fig. 4. Suppose that node u gets permanently labeled along P_1 and that because of the delay constraint no relabeling of any neighbor node of u is possible. Since node u is permanently labeled, it cannot be relabeled again along P_2. This means that node u blocks the search for optimum paths along P_2. This can be remedied if we incorporate " *node unlabeling*" in the LHWHM algorithm. *Node unlabeling* requires setting CLABEL(u) =∞, DLABEL(u) =0, PERM(u)=0 and PRED(u)= u, once it is detected that no update of the labels of the neighbor nodes of u is possible .

The question that we need to answer is whether the revised version of the LHWHM algorithm incorporating *node unlabeling* is an improvement over the original LHWHM algorithm. Unfortunately this is not so. We can construct an example to demonstrate this. This is not surprising, since for any heuristic one can construct an example where the heuristic performs very poorly. Improvements are possible if the *node unlabeling* operation is modified as follows.

If no update of the labels of the neighbor nodes of the permanently labeled node u is possible, then set
CLABEL(u) = Immediate past value of CLABEL(u) and,
DLABEL(u) = Immediate past value DLABEL(u).

Implementation of this new version of *node unlabeling* will require maintaining at each node past label values and thus the space complexity of the algorithm will be increased considerably. Time complexity will also increase and would depend on the number of node unlabelings done during the course of the algorithm. However, we can keep these complexities within practical limits, if we set a bound on the number of times a node is allowed to be unlabeled.

Summarizing, *node unlabeling* does not necessarily result in an improvement over the LHWHM algorithm. This is because, though *node unlabeling* allows new paths to be explored, these new paths themselves may block certain paths that would have otherwise been explored by the original LHWHM algorithm. However, except in some rare cases, *node unlabeling* did not help improve upon the cost of the produced by the LHWHM algorithm.

5.2 Multi-depth Heuristic: A Generalization of the LHWHM Algorithm

We next draw attention to yet another factor which causes the LHWHM algorithm to produce non-optimal solutions, and then propose methods to remedy the situation. First, we note that once a node u is permanently labeled in the LHWHM algorithm, then there exists a unique s-u path containing only permanently labeled nodes. We call this the s-u tree path. The LHWHM algorithm, before labeling a node x from a permanently node u, first checks to see if there exists a feasible s-t path containing the s-u tree path and the edge (u, x). This is done by testing (condition (1) in section II) if

$$\text{DLABEL}(u) + d_{u,x} + \text{D}(x) \leq T \qquad \ldots (2)$$

If this condition is satisfied, then it is concluded that there exists the required feasible s-t path passing through u and x, and so the algorithm proceeds to label x, provided the cost condition is also satisfied.
Let us now examine condition (2). Recall that $\text{D}(x)$ is the minimum delay of any x-t path in the original network N. In condition (2) what we really need , in place of $\text{D}(x)$, is the minimum delay of any x-t path containing no permanently labeled nodes at this stage of the algorithm. Since $\text{D}(x)$ is only a lower bound on the minimum delay of any x-t path containing no permanently labeled nodes, satisfaction of this condition alone would not guarantee the existence of the required feasible s-t path passing through u and x. Thus we have two important conclusions summarized in the following theorem.

Theorem 1.

a) If condition (2) *is not satisfied, then there exists no feasible s-t path containing the s-u tree path and the edge (u,x).*
b) Satisfaction of condition (2) *does not guarantee the existence of a feasible s-t path described in (a) above//.*

Condition (b) in the above theorem means that the LHWHM algorithm, on the satisfaction of condition (2), may conclude that the required feasible s-t path exists and hence will label x, but detect infeasibility in several iterations later. To remedy the problem we have the following options.

OPTIONS

Suppose u is the most recent permanently labeled node. Let x be a neighbor of u not yet permanently labeled. Let PL be the set of nodes permanently labeled at this stage, and N^* be the network containing node x and the nodes not in PL along with the links connecting them.

Option 1 (See Fig. 5(a)):

Perform Dijkstra's algorithm on N^* with x as the source node, and using link delays determine the delay of a minimum delay path from x to the destination t in the network N^*. This gives the new value of D(x) to be used in condition (2).

Option 2 (See Fig. 5(b)):

Run the LHWHM algorithm on N^* with x as the source node and t as the destination node. Also,let us initialize two new variables CLABEL* and DLABEL* as follows.

$\text{CLABEL}^*(x) = 0$,

$\text{DLABEL}^*(x) = \text{DLABEL}(u) + d_{u,x}$,

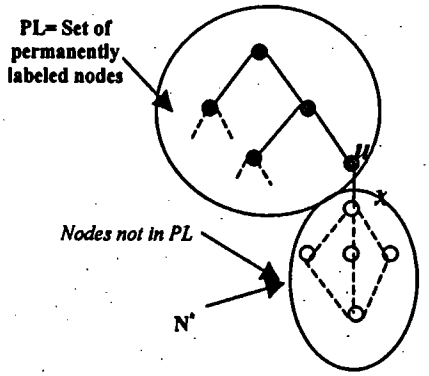
$\text{CLABEL}^*(v) = \infty$ and $\text{DLABEL}^*(v) = 0$ for all $v \notin$ PL and $v \neq x$,

where DLABEL(u) is the current delay value of u in the LHWHM algorithm applied on N. Note that the D(v) values to be used are the same as those used for the LHWHM algorithm running on N. If the algorithm on N^* is run to completion and at termination the node t is permanently labeled then this would guarantee the following:

There exits a feasible s-t path containing the s-u tree path and the edge (u,x). So, in this case, return to the original LHWHM algorithm and label x from u.

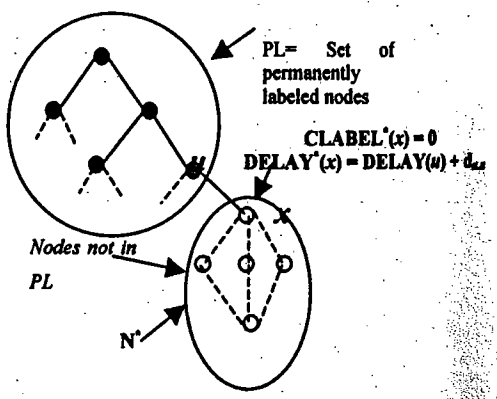
On the other hand, if the LHWHM algorithm running on N^* terminates without labeling t, then this would mean the following:

If the node x gets permanently labeled from u, then the LHWHM algorithm running on N would detect infeasibility at some later iteration. So, in this case, the LHWHM algorithm should not label x from u.



Find the min-delay from x to t in N^* and use this value in place of D(x).

Fig.5(a)



Run the LHWHM algorithm on N* with labels of x initialized as above.

Fig.5(b)

Running to completion the LHWHM algorithm on N^* will involve excessively large amount of computation time. So we propose to terminate this phase after the algorithm has labeled k nodes of N^* permanently. If the algorithm stops after labeling k nodes permanently or labeling t, then we would let LHWHM algorithm running on N label x from u. Otherwise, x will not be labeled from u. We call this heuristic the ***k-depth heuristic for the BDMCP problem.***

Replacing the general step in the LHWHM algorithm as discussed above, we get the *k-depth heuristic.* It is easy to see that the time complexity of the k-depth heuristic is $O(k\,n^3)$.

VI. The Bellman-Ford-Moore Based Heuristic for the BDMCP Problem

As we pointed out earlier, there are two algorithms for the single source shortest path problem:- Dijkstra's algorithm and the Bellman-Ford-Moore (BFM) algorithm. The LHWHM heuristic for the BDMCP problem is based on Dijkstra's algorithm. We present in this section a heuristic based on the BFM algorithm to be called the BFM-BDMCP algorithm. We first present the BFM algorithm. We then present the BFM-BMCP algorithm. This is followed by a discussion of certain fundamental characteristics of this heuristic which distinguish it from the LHWHM algorithm and also certain fundamental implementational issues that arise because of these unique features. Finally, we prove that the BFM-BDMCP heuristic does find a feasible path, if one such path exists.

6.1 The BFM Algorithm

As in Dijkstra's algorithm, the BFM algorithm associates with each node u two variables CLABEL(u) and PRED(u). Initially CLABEL(s) =0, CLABEL(u) = ∞ ,for all $u \in$ V and $u \neq s$ and PRED(u) =u for all $u \in V$.

The *general step* in the algorithm :

Pick any link (u,v) such that CLABEL(u) $\neq\infty$ and CLABEL(v) > CLABEL(u) + $c_{u,v}$.
Set CLABEL(v) = CLABEL(u) + $c_{u,v}$ and PRED(v) =u.
If no such link is available, the BFM algorithm terminates.

At termination CLABEL(t) gives the cost of a minimum cost s-t path. This path can be traced starting at PRED(t) and working backwards towards the source node s.

We next present an efficient $O(m\ n)$ implementation of the BFM algorithm [16].
In the following discussion scanning a node u means examining all the edges (u,v) and labeling the neighbor nodes of u, if possible. A sweep of the BFM algorithm refers to the process of scanning all the nodes as 1,2,3,.....n in that order.
Following is an implementation of the BFM algorithm using the concept of the sweep.

The BFM ALGORITHM

1. (INITIALIZATION) Set CLABEL(s) = 0, CLABEL(u) = ∞ , for all $u \in V$ and $u \neq s$ and PRED(u) = u , for all $u \in V$.
2. Perform a Sweep.
3. If no node CLABEL value gets updated, STOP. CLABEL(t) gives the cost of the min-cost shortest s-t path. Otherwise, repeat Step 2.

It can be shown that after performing n sweeps the BFM algorithm will terminate, resulting in the time complexity of O(mn). The above implementation of the BFM algorithm permits two choices which are presented below.

Version 1 (*Asynchronous*)
While scanning a node u during a sweep, use the current value of CLABEL(u) to label the neighbors of node u.

Version 2 (*Synchronous*)
While scanning node u during a sweep, use the value of CLABEL(u) at the end of the previous sweep for updating the labels of the neighbors of node u.

Asynchronous distributed implementation of BFM algorithm uses *Version 1* and synchronous distributed implementation distributed implementation uses *Version 2* [16], [17]. So these two versions will be referred to asynchronous and synchronous versions, respectively.

6.2 BFM-BDMCP HEURISTIC

The BFM-BDMCP heuristic is based on the BFM algorithm. As in the case of the LHWHM algorithm, we first compute D(v) for each node v where, as we may recall, D(v) is the minimum delay of any v-t path. Again each node v is associated with the variables CLABEL(v), DLABEL(v) and PRED(v). Initialization of the BFM-BDMCP algorithm is as follows.

CLABEL(s) =0, CLABEL(u) = ∞ , for all $u \in V$ and $u \neq s$, and
DLABEL(u) =0 for all $u \in V$.

For reasons which will be made clear later, we are not at this point specifying the initialization of PRED(v) values or the mechanism for the update of these values as the algorithms proceeds. Scanning a node u involves the following:

For each neighbor v of u do:

If DLABEL(u) + $d_{u,v}$ + D(v) ≤ T and CLABEL(v) > CLABEL(u) + $c_{u,v}$ then set CLABEL(v) = CLABEL(u) + $c_{u,v}$.

A formal presentation of the BFM-BDMCP algorithm is as follows

BFM-BDMCP Algorithm

1. Initialize the variables CLABEL(v), and DLABEL(v) for all nodes $v \in V$.
2. Perform a Sweep.
3. If no node CLABEL value is updated during a sweep, STOP (the value of CLABEL(t) gives the cost of the final feasible s-t path and is a sub-optimal solution to the BDMCP problem). Otherwise repeat Step 2.

We can use the synchronous version or the asynchronous version while implementing the Step 2. We next proceed to highlight certain characteristics of the BFM-BDMCP algorithm.

Suppose while labeling from node u, the CLABEL(v) of neighbor node v is updated . We can view this as "node u initiating a wave to node v". Basically this means that node u has extended a current s-u path to an s-v path. In the case of the LHWHM algorithm, during an iteration only one node (the most recently permanently labeled node) is scanned. Thus at most n new waves (equivalently, paths) are initiated. And only one of these waves gets chosen for propagation of waves in the subsequent iteration. In fact, at most n waves will get a chance to propagate new waves during the entire algorithm . On the other hand, in the case of the BFM-BDMCP algorithm, an iteration corresponds to a sweep, that is, the scanning of all the nodes of the network. This means that during a sweep a number of waves are initiated. Therefore a large number of paths become considered for further extensions to feasible s –t paths. This is because every node whose CLABEL gets updated during a sweep gets a chance to initiate a wave (a new path) for exploration and does not have to wait until it becomes permanently labeled as in the case of the LHWHM algorithm. It is for this reason that we expect the BFM-BDMCP algorithm to outperform the LHWHM algorithm. This is illustrated further next. Consider again Fig. 4. Suppose node u is updated along path P_1 with new labels CLABEL(u) = x_1 and DLABEL(u) = y_1 .This causes a new wave, say WAVE1, to be initiated by u. Suppose at a subsequent sweep node u is updated along path P_2 with new CALBEL(u) = x_2 and DLABEL(u) = y_2. This causes a new wave, say, WAVE2, to be initiated by node u. Clearly $x_2 < x_1$. Suppose y_2 is much larger than y_1, then WAVE2 may detect in a subsequent sweep that it cannot extend itself to a feasible s-t path. On the other hand, WAVE1, because of the lower value of y_1, may be able to reach t and label it, thereby finding a feasible s-t path. In the case of the LHWHM algorithm, WAVE1 would not even have been initiated, thereby loosing an opportunity to discover a feasible s-t path.

We now examine the impact of the above scenario on the tracing of the s-t path using the PRED values, starting from node t and moving backwards towards s.

Suppose we initialize PRED(i)= i for all $i \in V$ and set PRED(i) = j whenever i gets updated while scanning node j. If we use this scheme for updating PRED values, then in the scenario (Fig.4) we have just considered PRED(u) will be first set to k (the node preceding u in the path P_1) when u is labeled by k along path P_1. In a subsequent step PRED(u) will again be reset to w, the node preceding u in P_2. WAVE1 reaches t and labels it. But WAVE2 does not reach t. So, when we trace the s-t path from t we reach node u and find node w as a predecessor and not k which was the cause for initiating WAVE1 and labeling t. Thus, using the scheme used in storing and updating PRED values as in the LHWHM algorithm will not help in correctly identifying the final s-t path. So we now propose an alternate scheme to remedy this problem.
In our scheme PRED(v) is a list of entries. Each entry in the list has three components
(x, y, z). Initially PRED(v) for every node v has only one entry, namely (v,0,0). When the labels of a node v are updated by node u then an entry (x,y,z) is added to the list PRED(v) where x ,y and z are defined as follows.

$x = u$,
y = CLABEL(u), and
z = DLABEL(u).

With this initialization of PRED(v) lists, our scheme for tracing the find s-t path is :

1. Let the final entry in PRED(t) be (x,y,z). Note that t received its labels from node x, and that y and z are, respectively, the values of the variables CLABEL(x) and DLABEL(x) when x labeled t. Also note that x is the predecessor of t in the final s-t path.
2. Search in the PRED(x) list for an entry (x', y',z') such that
   $y = y' + c_{x',x}$
   $z = z' + d_{x',x}$
   Then x ' is the predecessor of x in the final s-t path. Next set
   $x = x'$
   $y = y'$
   $z = z'$
3. Repeat step 2 until $x = s$.

It appears that to implement the above scheme for tracing the s-t path we need to store a large number of entries in PRED lists. This will be the case only if we keep adding an entry to PRED(v) list every time v is updated. This is not necessary because we need to keep only those entries which cause node v to propagate new waves. This means that during each sweep at most two new entries need to be added to each PRED(v) (In a careful implementation we need to add at most one new entry to PRED(v)).Thus the size of each PRED(v) list is at most $O(n)$. Hence the space complexity of the algorithm is $O(n^2)$. Since the BFM algorithm is known to terminate in n sweeps, the BFM-BDMCP algorithm also will terminate in n sweeps. Hence the time complexity of this algorithm is $O(m\,n)$.

We now prove that the BFM-BDMCP algorithm, at termination, finds a feasible s-t path if one such path exists. In the following we call a node u unlabeled if its CLABEL(u) = ∞.

Theorem 2 :
The BFM-BDMCP algorithm terminates with a feasible s-t path, if one such path exists.

PROOF
Proof is by contradiction. Assume that at termination of the BFM-BDMCP algorithm, node t is unlabeled. Let L be the set of all labeled nodes at termination and M be the set of all unlabeled nodes. So, $s \in L$ and $t \in M$. Also every node u in L is a labeled node with CLABEL(u) $\neq \infty$. Recall that for each labeled node u there exists an s-u path containing only labeled nodes, which can be traced by the PRED array entries. This path is called the s-u tree path.

Since there exists a feasible s-t path , select a min-delay s-t path P. Let v be the last labeled node in P as we traverse P from s towards t. Thus all successors of v in P are in the set M. Let u be the predecessor of v. This means that node v received its final CLABEL and DLABEL values while scanning node u. So,

DLABEL(u) + $d_{u,v}$ + D(v) $\leq$ T , and
DLABEL(u) + $d_{u,v}$ = DLABEL(v).

Since P is a minimum delay *s-t* path it follows that for every node x on P ,

$D(x)$ = delay of the *x-t* segment of P.

Let y be the node next to v in the *s-t* path P. Note that y belongs to M. Then

$$\begin{aligned} DLABEL(v) + d_{v,y} + D(y) &= DLABEL(v) + \text{delay of the } v\text{-}t \text{ segment of path P} \\ &= DLABEL(u) + d_{u,v} + \text{delay of the } v\text{-}t \text{ segment path P} \\ &= DLABEL(u) + d_{u,v} + D(v) \\ &\leq T \end{aligned}$$

So y is eligible for labeling from node v. That is, y satisfies the delay constraint while scanning node v. Thus while scanning node v, node y must have received a finite CLABEL value, and would have become labeled. This contradicts our assumption that node $y \in M$. Hence the theorem//.

The proof employed in the above theorem is also applicable to all the heuristics presented in earlier sections of this paper. In fact all these heuristics can be extended to cover cases involving multiple non-additive constraints such as bandwidth, besides the additive cost and delay constraints thatwe have already considered. In such cases, these extensions will still have the property proved in Theorem 4.1. However, the theorem is not applicable if more than two additive constraints are involved because the problem in that case is NP- complete [2].

VII. Simulation and Performance Evaluation

In this section we first present an evaluation of the performance of the LHWHM and the BFM-BDMCP algorithms and compare them with the performance of the LARAC algorithm [13]. We only report results for the regular graphs $H_{k,n}$, proposed by Harary [16]. Here k refers to the vertex degree and n refers to the number of vertices. The edge costs and delays are selected as follows:
Edge costs are randomly generated in the range 1 to 50 and delays are assigned values as follows: $d_{i,j} = 50 - c_{i,j}$.

The motivation for this choice is to test the heuristics under what we believe to be a worst-case scenario.

To generate the optimum solutions we used a dynamic programming algorithm. The results of the experiments comparing the LHWHM, BFM-BDMCP and LARAC algorithms are presented in Table 1. The simulations were carried out by generating source and destination nodes randomly for each graph. The delay T for the algorithms has been selected to be 10% more than the shortest delay path value. It is observed that for smaller networks all the three algorithms generate nearly optimal solutions, but as the graphs gets bigger the number of feasible paths also increase, in this case the BFM-BDMCP algorithm outperforms the LHWHM algorithm. Since the BFM algorithm inspects many more paths than the LHWHM algorithm, the time taken for the algorithm is also seen to increase. LARAC returns paths closer to the optimal in most of the cases, but the time taken for computing the paths was generally more than that taken by the LHWHM algorithm by a factor of 10. The OPT column in the tables refers to the optimal cost. We wish to note that the LARAC algorithm may not produce a feasible solution in all cases.

The results of our comparison with Hassin's FPAS (fully polynomial time approximation scheme) are presented in Table 2. We first ran the BFM-BDMCP algorithm the graphs mentioned above. For each graph, using the optimal values we computed the ε-value. These ε values are given as inputs to Hassin's algorithm [8]. Note that for a given value of ε, Hassin's algorithm guarantees a solution which is within (1+ε) of the optimum value. In Table 2, T1 refers to the time taken by Hassin's algorithm as proposed originally, and T2 refers to the time taken when the solution produced by the BFM-BDM heuristic is used as an upper bound in Hassin's algorithm. Though, as expected, Hassin's algorithm takes a considerably longer time than the BFM-BDMCP algorithm, it produces optimal solutions in most cases. This means that for small values of ε the algorithm produces optimal solutions. If we first run the BFM-BDMCP algorithm

and then use the result as an upper bound in Hassin's algorithm, the time taken (T2) gets reduced to a considerable extent.

NODES	LHWHM		BFM-BDMCP (Async)		BFM-BDMCP (Sync)		LARAC		OPT
	cost	time (sec)	cost	time	cost	time	cost	time	cost
100	168	1.54E-04	168	.0023	168	.0055	168	.00104	168
250	198	7.05E-04	198	.0268	198	0.046	198	.00514	198
400	552	2.08E-03	552	.0282	522	0.314	533	0.035	500
550	2650	7.60E-03	2650	.0986	2501	1.946	2423	0.233	2400
800	950	6.30E-03	950	0.145	807	1.83	900	0.088	751
1000	5651	3.25E-02	5550	9.75	5451	12.28	4702	0.68	4701
1400	10750	0.089	10400	41.12	10100	49.84	8765	1.91	8750
1800	16200	0.1709	14600	74.77	14600	100.8	13107	2.17	12890
2000	3801	0.028	3801	1.69	3450	13.09	3127	0.94	3100
2500	97	1.70E-02	97	23.47	97	33.15	97	0.04	97
3000	296	8.80E-02	296	41.91	296	78.15	296	.109	296

Table 1

Nodes	Eps(ε)	BFM-	FPAS			OPT
		cost	cost	T1 (sec)	T2 (sec)	cost
100	0.146	401	350	31.6	20.6	350
200	0.033	1553	1503	1020	306	1503
250	0.156	737	637	461	196	637
300	0.16	1451	1250	737	442	1250
350	0.13	3056	2703	1533	471	2703
400	0.151	1256	1091	1693	298	1091
450	0.01	859	851	806	603	851
500	0.15	5157	4855	2050	1382	4475

Table 2

VIII. Summary

In this paper we discussed one of the QoS routing problems formulated as the bounded delay minimum cost path problem (BDMCP). We presented a broad overview of the different heuristics and algorithms for this problem. We gave a detailed account of two recent heuristics called the LHWHM and BFM-BDMCP algorithms. We also provided an experimental evaluation of these heuristics in comparion of the LARAC algorithm of [13] and Hassin's approximation scheme. We summarize our conclusions as follows.

1. The LHWHM algorithm performs very well for sparse graphs.
2. In all cases, the BFM-BDMCP algorithm produces results better than those produced by the LHWHM algorithm.
3. The ε-approximation scheme of Hassin is computationally very expensive; but produces results very close to optimum for small values of ε.
4. If the cost of the final path produced by the BBFM-BDMCP algorithm is used as the initial upper bound in Hassin's algorithm, then the computation time of the approximation scheme is considerably reduced.
5. Even if a large value of ε (for instance, $\varepsilon = 0.9$) is used , Hassin's algorithm still produces results very close to the optimum in a significantly short time. But the execution time is still very excessive compared to that of the BFM-BDMCP algorithm. Due to space limitation the corresponding table is not included.

References

[1] S.Chen and K.Nahrsedt, " An Overview of Quality of Service Routing for Next Generation High Speed Networks: Problems and Solutions", *IEEE Network Magazine,* Vol.12 (6), 1998, pp.64-79

[2] Z.Wang and J.Crowcroft, "Quality-of-Service Routing for Supporting Multimedia Applications," *IEEE Journal on Selected Areas in Communications,* Vol. 14, no.7, September 1996, pp.1228-1234.

[3] R.Widyono ,"The Design and Evaluation of Routing Algorithms for Real-Time Channels," *Tech. Rep. ICSI TR-94-024,* University of California at Berkely, International Computer Science Institute, June 1994.

[4] Douglas S.Reeves, Hussein F.Salama, "A Distributed Agorithm for Delay-Constrained Unicast Routing". *IEEE/ACM Ttransactions on Networking,* Vol 8, no 2, April 2000, pp. 239-250.

[5] Jaffery M.Jaffe, "Algorithms for finding Paths with Multiple Constraints", *Networks,* Vol. 14, 1984, pp. 95-116.

[6] R. Guerin and A.Orda,"QoS Routing in Networks with Inaccurate Information: Theory and Algorithms," *IEEE/ACM Transactions on Networking.* Vol.7, no.3, June 1999.

[7] Gang Luo, Kaiyuan Huang, Jianli Wang,Chris Hobbs and Ernst Munter, "Multi-QoS Constraints Based Routing for IP and ATM Networks," in *Proceedings of IEEE Workshop on QoS Support for Real-Time Internet Applications,* June 1, 1999, Vancouver Canada.

[8] R.Hassin , "Approximation Schemes for the Restricted Shortest Path Problem", *Mathemetics of Operation Research,* 17(1), 1992, pp.36-42.

[9] Sung-Pi Hong, Sung-Jin Chung, Bum Hwan Park, "On "Strongly" and Fully Approximation Schemes for Restricted Shortest Path Problem", Technical Report, Seoul National University (csj@optima.snu.ac.kr)

[10] Danny Raz, and Yuval Shavitt, "Optimal Partition of QoS with Discrete Cost functions",In *Proceedings of IEEE INFOCOM 2000,* March 2000.

[11] Dean H. Lorenz, Ariel Orda, Danny Raz and Yuval Shavitt, "Efficient QoS Paritition and Routing of Unicast and Multicast", *IWQOS 2000,* June 2000, Pittsburgh, PA, USA, pp 75-83.

[12] D. Lorenz and D.Raz, " A Simple Efficient Approximation Scheme for the Restricted Shortest Path problem", *Operations Research Letters,* Vol. 28, No.5, pp. 213-219, 2001.

[13] Alpar Juttner, Balazs Szviatovszki, Ildiko Mecs, Zsolt Rajko, " Lagragian Relaxation Based Method for the QoS Routing Problem", *IEEE INFOCOM-2001,* pp. 859-868.

[14] Ravi Ravindran, K.Thulasiraman, Anindya Das, Kaiyuan Huang and Guoliang Xue," Quality of Service Routing in Communication Networks: Heuristics and Approximation Schemes with a Comparative Performance Evaluation", 2002. *Proceedings of the IEEE International Symposium on Circuits and Systems.*

[15] J.L.Sobrinho, "Algebra and Algorithms for QoS Path Computation and Hop-by-Hop Routing in the Internet ", *IEEE/ACM Transactions on Networking,* August 2002, pp.541-550.

[16] K. Thulasiraman and M.N.S.Swamy, Graphs: Theory and Algorithms (Book) , Wiley- Interscience, 1992.

[17] K. B. Lakshmanan and K. Thulasiraman, "On the Use of Synchronizers for Asynchronous Communication Networks", *2nd International Workshop on "Distributed Algorithms,* Amsterdam, July 1987.