On Finding Multi-constrained Paths [*]

Shigang Chen Klara Nahrstedt
Department of Computer Science
University of Illinois at Urbana-Champaign

Abstract

New emerging distributed multimedia applications provide guaranteed end-to-end quality of service (QoS) and have stringent constraints on delay, delay-jitter, cost, etc. The task of QoS routing is to find a route in the network which has sufficient resources to satisfy the constraints. The delay-cost-constrained routing problem is NP-complete. We propose a heuristic algorithm for this problem. The idea is to first reduce the NP-complete problem to a simpler one which can be solved in polynomial time, and then solve the new problem by either an extended Dijkstra's algorithm or an extended Bellman-Ford algorithm. We prove the correctness of our algorithm by showing that a solution for the simpler problem must also be a solution for the original problem. The performance of the algorithm is studied by both theoretical analysis and simulation.

1 Introduction

Quality of Service (QoS) routing has been attracting considerable attention in the research community recently [6, 10, 11, 12, 13]. The routing requests are typically specified in terms of constraints. For example, a delay (cost) constraint requires the total delay (cost) of a path to be not greater than a given upper bound. The multi-constrained routing problem is difficult because different constraints can conflict with one another. In particular, the delay-cost-constrained routing, i.e., finding a route between two nodes in the network with both end-to-end delay and end-to-end cost bounds, can be formalized as a *multi-constrained path problem* (MCP), which is NP-complete [5, 13].

We propose a heuristic algorithm for the MCP problem with a polynomial time complexity. The algorithm first reduces the NP-complete problem to a simpler one which can be solved in polynomial time, and then solve the simpler problem by an extended Dijkstra's or Bellman-Ford algorithm to find a solution path. When the extended Dijkstra's algorithm is used, the total time complexity of the heuristic algorithm is $O(x^2V^2)$; when the extended Bellman-Ford algorithm is used, the time complexity is $O(xVE)$, where x is an integer defined solely by the algorithm, V is the set of nodes and E is the set of edges. The value of x, which can be set arbitrarily by the user, determines the performance and the overhead of the algorithm. The performance of the algorithm is predictable and adjustable. It is predictable in the sense that when certain condition is satisfied the algorithm is guaranteed to find a solution. It is adjustable in the sense that the probability of finding a solution can be increased when the value of x is increased.

The rest of the paper is organized as follows. In Section 2, the heuristic algorithm for the general MCP problem is first presented, based on that the delay-cost-constrained routing algorithm is proposed, and finally, the performance of the routing algorithm is studied by experiments. The related work is covered in Section 3. Section 4 draws the conclusion.

2 A Polynomial-time Heuristic algorithm

2.1 The heuristic algorithm

Let $\mathbf{R}_0^+$ be the set of non-negative real numbers and $\mathbf{I}$ the set of non-negative integers.

Definition 1 *Multi-constrained path problem* (MCP): Given a directed graph $G\langle V, E\rangle$, a source vertex s, a destination vertex t, two weight functions $w_1 : E \to \mathbf{R}_0^+$ and $w_2 : E \to \mathbf{R}_0^+$, two constants $c_1 \in \mathbf{R}_0^+$ and $c_2 \in \mathbf{R}_0^+$; the problem, denoted as MCP$(G, s, t, w_1, w_2, c_1, c_2)$, is to find a path p from s to t such that $w_1(p) \leq c_1$ and $w_2(p) \leq c_2$ if such a path exists.

A path p which satisfies $w_1(p) \leq c_1$ and $w_2(p) \leq c_2$ is called a *solution* for MCP$(G, s, t, w_1, w_2, c_1, c_2)$. We assume that both weight functions are *additive* — the weight of a path is equal to the summation of the weights of all edges on the path.

Definition 2 *w_1-weight and w_2-weight*: For a path $p = v_0 \to v_1 \to \ldots, \to v_k$, $w_1(p) = \sum_{i=1}^{k} w_1(v_{i-1}, v_i)$ and $w_2(p) = \sum_{i=1}^{k} w_2(v_{i-1}, v_i)$. $w_1(p)$ is called the *w_1-weight* and $w_2(p)$ the *w_2-weight* of the path p.

MCP$(G, s, t, w_1, w_2, c_1, c_2)$ is NP-complete [13]. We provide a polynomial-time heuristic solution for this problem. The algorithm contains two steps:

*This work was supported by the ARPA grant under contract number F30602-97-2-0121 and the National Science Foundation Career grant under contract number NSF CCR 96-23867.

1. Create a new weight function $w_2' : E \rightarrow I$.

$$w_2'(u,v) = \lceil \frac{w_2(u,v) \cdot x}{c_2} \rceil \qquad (1)$$

   where x is a given positive integer. We reduce the original problem MCP$(G, s, t, w_1, w_2, c_1, c_2)$ to a new, simpler problem MCP$(G, s, t, w_1, w_2', c_1, x)$. [1]

2. Solve MCP$(G, s, t, w_1, w_2', c_1, x)$ in polynomial time.

The algorithms for Step 2 will be discussed in Section 2.2. We assume for the moment that a solution of MCP(G, s, t, w_1, w_2', c_1, x) can be found in polynomial time if there exists one.

Since MCP$(G, s, t, w_1, w_2, c_1, c_2)$ is NP-complete, we are not trying to find a solution for it whenever there exists one. The idea is to reduce it to a simpler and solvable problem, MCP$(G, s, t, w_1, w_2', c_1, x)$, which has a "coarser resolution" — x is a given finite integer and the range of w_2' is **I**. Theorem 1 guarantees that a solution for the simpler problem must be a solution for the original problem.

Theorem 1 A solution for MCP$(G, s, t, w_1, w_2', c_1, x)$ must also be a solution for MCP$(G, s, t, w_1, w_2, c_1, c_2)$.

Proof: Let p be a solution for MCP$(G, s, t, w_1, w_2', c_1, x)$. Hence, $w_1(p) \leq c_1$ and $w_2'(p) \leq x$. In order to prove p is also a solution for MCP$(G, s, t, w_1, w_2, c_1, c_2)$, it suffices to prove $w_2(p) \leq c_2$. By the definition of w_2', we have

$$w_2'(u,v) = \lceil \frac{w_2(u,v) \cdot x}{c_2} \rceil \qquad \text{see (1)}$$

which means $w_2(u,v) \leq \frac{w_2'(u,v) \cdot c_2}{x}$. Therefore, we have

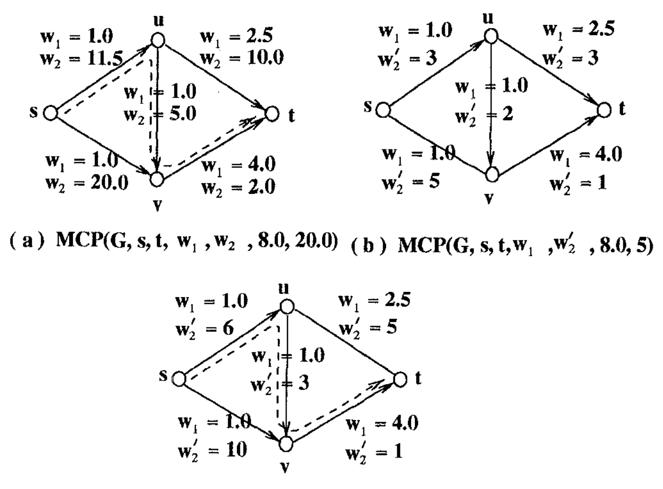
$$\begin{aligned} w_2(p) = \sum_{(u,v) \text{ on } p} w_2(u,v) &\leq \sum_{(u,v) \text{ on } p} \frac{w_2'(u,v) \cdot c_2}{x} \\ = \frac{c_2}{x} \cdot \sum_{(u,v) \text{ on } p} w_2'(u,v) &= \frac{c_2}{x} \cdot w_2'(p) \leq \frac{c_2}{x} \cdot x = c_2 \end{aligned}$$

$w_2(p) \leq c_2$ and hence the theorem holds. □

Corollary 1 Let P be the set of solutions of MCP(G, s, t, w_1, w_2, c_1, c_2) and P' be the set of solutions of MCP(G, s, t, w_1, w_2', c_1, x). Then, $P' \subseteq P$

Corollary 2 Let P' be the set of solutions of MCP(G, s, t, w_1, w_2', c_1, x). The heuristic algorithm succeeds in finding a solution for MCP(G, s, t, w_1, w_2, c_1, c_2) if and only if $P' \neq \emptyset$.

[1]Note that the value of x is chosen by the algorithm. It does not depend on the input values of G, s, w_1, w_2, c_1 and c_2. This is the essential reason for Step 2 of the algorithm to be solvable in polynomial time. As we will see shortly, a larger x means a higher chance to find a satisfactory path and a higher overhead.



(a) MCP(G, s, t, w_1, w_2, 8.0, 20.0) (b) MCP(G, s, t, w_1, w_2', 8.0, 5)

(c) MCP(G, s, t, w_1, w_2', 8.0, 10)

Figure 1: (a) The original problem has a solution, $s \rightarrow u \rightarrow v \rightarrow t$. (b) If $x = 5$, the reduced problem does not have a solution. (c) If $x = 10$, the reduced problem has a solution.

The converse of Theorem 1 is not necessarily true — a solution for MCP$(G, s, t, w_1, w_2, c_1, c_2)$ may not be a solution for MCP$(G, s, t, w_1, w_2', c_1, x)$. Figure 1 gives an example. The original problem MCP$(G, s, t, w_1, w_2, 8.0, 20.0)$ has a solution $s \rightarrow u \rightarrow v \rightarrow t$ (Figure 1 (a)) . Suppose $x = 5$ and the problem is reduced to MCP$(G, s, t, w_1, w_2', 8.0, 5)$ (Figure 1 (b)). The path $s \rightarrow u \rightarrow v \rightarrow t$ is not a solution for the new problem. In fact, there is no solution for the new problem.

Hence, our heuristic algorithm may not find a solution for MCP$(G, s, t, w_1, w_2, c_1, c_2)$ even when such a solution exists, because the solution set P' of the new problem MCP(G, s, t, w_1, w_2', c_1, x) can be empty. Fortunately, whether P' is empty or not is to some extent predictable and adjustable — by assigning a larger x, we have a better chance for P' to be non-empty.

Theorem 2 Let a path p be a solution for MCP(G, s, t, w_1, w_2, c_1, c_2) and l be the length of p. If

$$w_2(p) \leq (1 - \frac{l-1}{x}) \cdot c_2 \qquad (2)$$

then p is also a solution for MCP$(G, s, t, w_1, w_2', c_1, x)$.

Proof: Since p is a solution for MCP$(G, s, t, w_1, w_2, c_1, c_2)$, we already have $w_1(p) \leq c_1$. In order to prove p is a solution for MCP$(G, s, t, w_1, w_2', c_1, x)$, we only need to prove $w_2'(p) \leq x$.

$$\begin{aligned} w_2'(p) &= \sum_{(u,v) \text{ on } p} w_2'(u,v) \overset{\text{by (1)}}{=} \sum_{(u,v) \text{ on } p} \lceil \frac{w_2(u,v) \cdot x}{c_2} \rceil \\ &< \sum_{(u,v) \text{ on } p} (\frac{w_2(u,v) \cdot x}{c_2} + 1) \\ &= \frac{x}{c_2} \cdot \sum_{(u,v) \text{ on } p} w_2(u,v) + \sum_{(u,v) \text{ on } p} 1 \\ &= \frac{x}{c_2} \cdot w_2(p) + l \overset{\text{by (2)}}{\leq} \frac{x}{c_2} \cdot (1 - \frac{l-1}{x}) \cdot c_2 + l = x + 1 \end{aligned}$$

Because both $w'_2(p)$ and x are integers, $w'_2(p) \leq x$. Therefore, the theorem holds. □

Theorem 2 means that if there exists a path p which is *overqualified* — not just $w_2(p) \leq c_2$ but $w_2(p) \leq (1-\frac{l-1}{x})c_2$ — then after we reduce the original problem to MCP(G, s, t, w_1, w'_2, c_1, x), the new problem still has solutions (p is one of them). Hence, we can solve MCP(G, s, t, w_1, w'_2, c_1, x) to find a solution, which must also be a solution for the original problem as stated by Theorem 1.

Corollary 3 Let P be the set of solutions of MCP(G, s, t, w_1, w_2, c_1, c_2) and P' be the set of solutions of MCP(G, s, t, w_1, w'_2, c_1, x). Then,

$$P' \neq \emptyset \text{ if } P \neq \emptyset \text{ and } \exists p \in P, w_2(p) \leq (1 - \frac{l-1}{x}) \cdot c_2$$

where l is the length of p.

Theorem 3 [2] Let P be the set of solutions of MCP(G, s, t, w_1, w_2, c_1, c_2). The heuristic algorithm succeeds in finding a solution for MCP(G, s, t, w_1, w_2, c_1, c_2) if

$$P \neq \emptyset \text{ and } \exists p \in P, w_2(p) \leq (1 - \frac{l-1}{x}) \cdot c_2$$

Proof: By combining Corollaries 2 and 3. □

The condition, $P \neq \emptyset$ and $\exists p \in P, w_2(p) \leq (1-\frac{l-1}{x})c_2$, is called the *heuristic condition*, where l is the length of p. Note that it is a sufficient but **not** a necessary condition in Theorem 3. With a larger x, the condition $w_2(p) \leq (1-\frac{l-1}{x})c_2$ has a better chance to be satisfied, which leads to a higher probability for the heuristic algorithm to find a solution.

Take the case $x = 10|V|$ as an example. Consider the worst case where the longest loop-free path has a length of $|V|-1$. The condition can be rewritten as $w_2(p) \leq 0.9c_2$. It means that, given an arbitrary problem MCP(G, s, t, w_1, w_2, c_1, c_2), if the problem has a solution p such that $w_2(p) \leq 0.9c_2$, then our heuristic algorithm is *guaranteed* to find a solution for the problem, provided x is as large as $10|V|$. A more detailed analysis of the relation between x and the heuristic condition can be found in [1]. In practice, how large should x be? This question will be discussed in Section 2.4 though experiments.

There remains another important question: Is the new problem MCP($G, s, t, w_1, w'_2, c_1, x$) in Step 2 solvable in polynomial time? We answer the question in Section 2.2.

2.2 The extended Dijkstra's and Bellman-Ford algorithms

An extended Dijkstra's shortest path algorithm (EDSP) and an extended Bellman-Ford algorithm (EBF) are presented below to solve MCP($G, s, t, w_1, w'_2, c_1, x$) in polynomial time. An algorithm similar to EBF, in its distributed implementation, has been proposed by Jaffe [5]. We will discuss the difference between our algorithm and the Jaffe's algorithm in Section 3.

```
     Initialize($G, s$)
     begin
(1)      for each vertex $v \in V[G]$, each $i \in [0..x]$ do
(2)          $d[v,i] := \infty$
(3)          $\pi[v,i] := NIL$
(4)      for each $i \in [0..x]$ do
(5)          $d[s,i] := 0$
     end

     Relax($u, k, v$)
     begin
(6)      $k' := k + w'_2(u,v)$
(7)      if $k' \leq x$ then
(8)          if $d[v,k'] > d[u,k] + w_1(u,v)$ then
(9)              $d[v,k'] := d[u,k] + w_1(u,v)$
(10)             $\pi[v,k'] := u$
     end

     EDSP($G, s$)
     begin
(11)     Initialize($G, s$)
(12)     $S := \emptyset$
(13)     $Q := \{\langle u,k\rangle | u \in V[G], k \in [0..x]\}$
(14)     while $Q \neq \emptyset$ do
(15)         find $\langle u,k\rangle \in Q$ such that $d[u,k] = \min_{\langle u',k'\rangle \in Q} \{d[u',k']\}$
(16)         $Q := Q - \{\langle u,k\rangle\}$
(17)         $S := S + \{\langle u,k\rangle\}$
             /* Note that the for loop iterates on
                 different adjacent vertices $v$. */
(18)         for each outgoing edge of $u$, $(u,v) \in E$ do
(19)             Relax($u, k, v$)
     end

     EBF($G, s$)
     begin
(20)     Initialize($G, s$)
(21)     for $i := 1$ to $|V[G]| - 1$ do
(22)         for $k := 0$ to $x$ do
(23)             for each edge $(u,v) \in E[G]$ do
(24)                 Relax($u, k, v$)
     end
```

For each vertex $v \in V$ and each integer $k \in [0..x]$, a variable $d[v,k]$ is maintained, which is an **estimation** of the smallest w_1-weight of those paths from s to v whose w'_2-weights are k. Let

$$\delta(v,k) = \min_{p \in P(v,k)} \{w_1(p)\}$$

where $P(v,k) = \{p \mid w'_2(p) = k,\ p \text{ is any path from } s \text{ to } v\}$. The value of $d[v,k]$, initially $+\infty$, is always greater than or

[2]Theorem 3 can be rewritten as: The heuristic algorithm succeeds in finding a solution for MCP(G, s, t, w_1, w_2, c_1, c_2) if there exists a path p from s to t such that $w_1(p) \leq c_1 \ \wedge \ w_2(p) \leq (1 - \frac{l-1}{x}) \cdot c_2$

equal to $\delta(v,k)$. During the execution, EDSP (EBF) makes better and better estimation and $d[v,k]$ becomes closer and closer to, and eventually reach, $\delta(v,k)$.

When EDSP (EBF) completes, $d[v,k] = \delta(v,k), v \in V, k \in [0..x]$. There exists a solution, i.e. a path p from s to t such that $w_1(p) \le c_1$ and $w_2'(p) \le x$, iff $\exists k \in [0..x]$, $d[t,k] \le c_1$. The path is stored by the variable π. $\pi[v,k]$ keeps the immediate preceding vertex (called *predecessor*) of v on the path. Hence, the path can be recovered by tracing the variable π starting from t, through all intermediate vertices, till reaching the source s.

Two additional variables, S and Q, are required by EDSP.

$$S = \{\langle v,k\rangle \mid d[v,k] = \delta(v,k), v \in V, k \in [0..x]\}$$

$$Q = \{\langle v,k\rangle \mid d[v,k] > \delta(v,k), v \in V, k \in [0..x]\}$$

where the notation $\langle v,k\rangle$ simply means a pair of values, $v \in V$ and $k \in [0..x]$. Initially, $S = \emptyset$ and $Q = \{\langle v,k\rangle \mid v \in V, k \in [0..x]\}$. In the **while** loop (lines 14-19 of the algorithm), each iteration moves a pair from Q to S and adjusts the w_1-weight estimation by calling Relax(u,k,v). When $Q = \emptyset$, the algorithm completes.

A more detailed presentation of the original Dijkstra's and Bellman-Ford algorithms, which our algorithms are based on, can be found in [3].

The time complexity of ESDP is (x^2V^2). The maximum size of Q is $(x+1)V$. Hence, line 15 can be done within $O(xV)$. There can be at most $(x+1)V$ iterations of the **while** loop and thus the total time for line 15 is $O(x^2V^2)$. The **for** loop of lines 18-19 has $(x+1)E$ iterations in total [3] because Relax(u,k,v) is called once for every $(u,v) \in E$, $k \in [0..x]$. In each iteration, Relax(u,k,v) takes $O(1)$. Hence, the time complexity for this part is $O(xE)$. The total time complexity is $O(x^2V^2 + xE) = O(x^2V^2)$. The time complexity of EBF is $O(xVE)$, because line 23 is executed for at most $(x+1)(V-1)E$ times. The space complexities of both algorithms are $O(xV)$.

Let us consider the time complexity of our heuristic algorithm in Section 2.1. Step 1 of the algorithm takes $O(E)$. Step 2 of the algorithm is implemented by EDSP or EBF. Therefore, the total time complexity is $O(x^2V^2)$ when EDSP is used or $O(xVE)$ when EBF is used. The time complexity is polynomial because the value of x is given by the algorithm. For example, if we let $x = 10|V|$ and use EBF in Step 2, the time complexity is $O(V^2E)$.

We have studied the heuristic algorithm for MCP with two weight functions and two constraints so far. However, the heuristic algorithm together with EDSP and EBF can be easily generalized for more than two constraints. The generalized algorithms can be found in [1].

[3] $(x+1)E$ iterations are the combination result of the outer **while** loop and the inner **for** loop. The **while** loop iterates on u and k, and the **for** loop iterates on v.

2.3 Multi-Constrained Routing

Multi-Constrained routing is an important application of MCP. Consider *delay* and *cost* [4] as the two weight functions. Given a source node s and a destination node t, the delay-cost-constrained routing problem is to find a path p from s to t such that *delay*$(p) \le D$ and *cost*$(p) \le C$, where D and C are the required end-to-end delay bound and cost bound, respectively. The routing algorithm is presented below.

1. Create two new functions *new-delay* : $E \to I$ and *new-cost* : $E \to I$.

$$\textit{new-delay}(u,v) = \lceil \frac{\textit{delay}(u,v) \cdot x}{D} \rceil$$

$$\textit{new-cost}(u,v) = \lceil \frac{\textit{cost}(u,v) \cdot x}{C} \rceil$$

   where $x = coef \times d_{s,t}$, *coef* is a given positive integer and $d_{s,t}$ is the distance from s to t. We reduce the original problem MCP$(G, s, t, \textit{delay}, \textit{cost}, D, C)$ to two simpler problems, MCP$(G, s, t, \textit{delay}, \textit{new-cost}, D, x)$ and MCP$(G, s, t, \textit{new-delay}, \textit{cost}, x, C)$.

2. First, solve MCP$(G, s, t, \textit{delay}, \textit{new-cost}, D, x)$ by EDSP or EBF. If a solution is found, return the found path and terminate; otherwise, solve MCP$(G, s, t, \textit{new-delay}, \textit{cost}, x, C)$.

The proposed routing algorithm applies the heuristic algorithm (Section 2.1) twice, reducing *delay* and *cost* to *new-delay* and *new-cost*, respectively. Hence, it guarantees to find a solution when *either* of the following two *heuristic conditions* is satisfied by a path p from s to t (see Theorem 3):

1. *delay*$(p) \le D \ \wedge\ $ *cost*$(p) \le (1 - \frac{l-1}{x}) \cdot C$.
   (Heuristic condition one)

2. *delay*$(p) \le (1 - \frac{l-1}{x}) \cdot D \ \wedge\ $ *cost*$(p) \le C$
   (Heuristic condition two)

We assume a source routing strategy, which was also adopted by routing algorithms in [7, 6, 9, 13]. It requires every node to maintain the state information of the network, which can be done by the link-state algorithm [8]. The routing path is determined locally at the source node.

2.4 Experiments

We know from the routing algorithm proposed in Section 2.3 that $x = coef \times d_{s,t}$. What is the relationship between *coef* and the performance of the algorithm and how large should *coef* be? We answer the questions by simulation.

The network topology used in our simulation is shown in Figure 2, which expends the major circuits in ANSNET [2] by inserting additional links to increase the connectivity. For each routing request, the values of s, t, *delay*, *cost*, D and

[4] The cost of an edge can be measured in dollars, or it can be a function of a given system metric such as bandwidth utilization or buffer utilization.

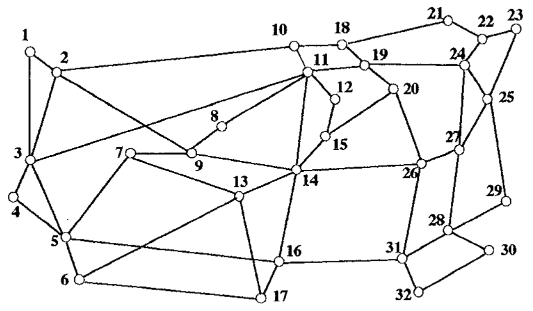


Figure 2: the topology of the experimental network

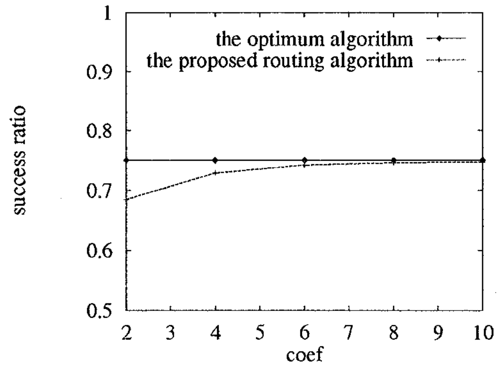


Figure 3: Success ratio with respect to *coef* when $D \in [100, 115ms]$ and $C \in [400, 460]$. Note that $x = coef \cdot d_{s,t}$. The average distance from the source to the destination of all experimental routing requests is 3.2 hops.

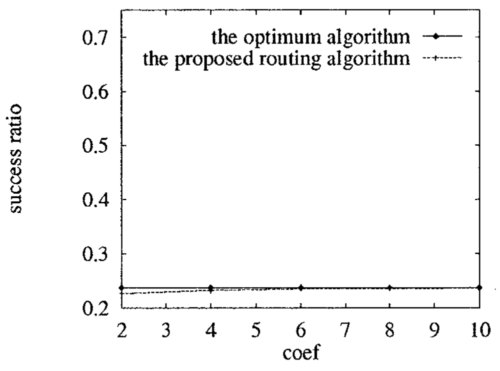


Figure 4: $D \in [50, 65ms]$, $C \in [200, 260]$

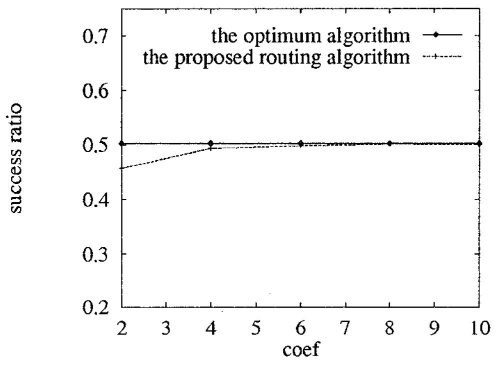


Figure 5: $D \in [75, 90ms]$, $C \in [300, 360]$

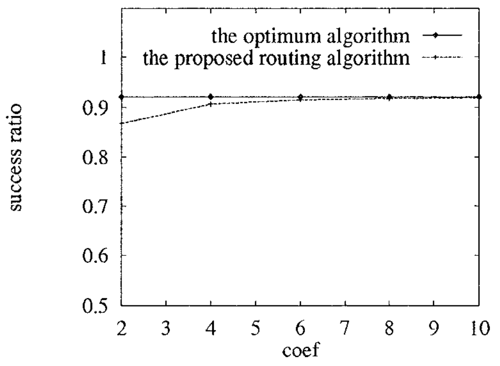


Figure 6: $D \in [125, 140ms]$, $C \in [500, 560]$

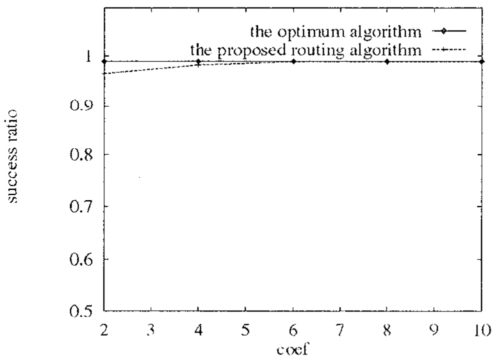


Figure 7: $D \in [150, 165ms]$, $C \in [600, 660]$

C are randomly generated. The *delay* values of the links are uniformly distributed in the range of $[0..50ms]$, and the *cost* values of the links are uniformly distributed in $[0..200]$. The performance metric we considered was *success ratio.*

$$success\ ratio = \frac{number\ of\ requests\ successfully\ routed}{total\ number\ of\ routing\ requests}$$

We studied the *success ratio* with respect to *coef*, D and C. The larger the value of *coef*, the higher the probability for the heuristic conditions to be satisfied, which leads to a higher *success ratio.* The smaller the values of D and C, the tighter the constraints of a routing request, which leads to a lower *success ratio.*

The experiment results are presented in Figures 3-7. The x axis represents *coef* and the y axis represents the *success ratio.* The dimensions of D and C are shown by different figures. Let us take Figure 3 as an example. Each point in the figure is taken by running one thousand randomly-generated routing requests. The values of D and C of all the requests are uniformly distributed in $[100, 115ms]$ and $[400, 460]$, respectively. For the purpose of comparison, we implemented an optimum algorithm, which searches all possible paths for a solution with an exponential time complexity. There are two lines in the figure. The upper horizontal line shows the *success ratios* of the optimum algorithm. The lower line shows the *success ratios* of the proposed routing algorithm. The *success ratio* of the proposed routing algorithm approaches that of the optimum algorithm when *coef* is increased. With *coef* ≥ 4, the performance of our algorithm is close to that of the optimum algorithm.

Figures 4-7 present the *success ratios* when $D \in [75, 90ms]$ and $C \in [300, 360]$, $D \in [125, 140ms]$ and $C \in [500, 560]$, $D \in [150, 165ms]$ and $C \in [600, 660]$, $D \in [150, 165ms]$ and $C \in [600, 660]$, respectively. Larger values for D and C result in more relaxed delay and cost constraints and thus higher *success ratios* as seen from the figures.

3 Related Work

Much work has been done in QoS routing recently [6, 10, 11, 12, 13]. Some routing algorithms consider a single constraint. Plotkin discussed the competitive routing strategy in [10], which considers only the bandwidth requirement. The Salama's algorithm [11] and the Sun's algorithm [12] consider the delay constraint. Though both algorithms use heuristic approaches trying to minimize the cost of the found route, the cost is not required to be bounded. The multi-constrained routing was studied in [6, 13]. Wang and Crowcroft [13] used the Dijkstra's algorithm in their bandwidth-delay-constrained routing. Ma and Steenkiste [6] showed that, when the WFQ-like scheduling algorithms are used, the metrics of delay, delay-jitter and buffer space are no longer independent from each other and all of them become functions of bandwidth, which simplifies the problem and makes it solvable in polynomial time.

All the above algorithms can not solve MCP whose weight functions are assumed to be additive and independent. The work closest to ours was done by Jaffe [5]. Jaffe proposed a distributed algorithm solving MCP with a time complexity of $O(V^5 b \log Vb)$, [5] where b is the largest weight of all edges in the network. The complexity is *pseudo-polynomial* because the run time is polynomial in b, the largest number in the input. See [4] for the definition of the pseudo-polynomial time complexity. Heuristic algorithms are also proposed in [5] to approximate the MCP problem. Instead of finding a multi-constrained path, the algorithms find a path with minimized $w_1(p) + d \cdot w_2(p)$. [6] However, minimizing $w_1(p) + d \cdot w_2(p)$ may not always lead to a solution of MCP. More detailed explanation and comparison between the Jaffe's algorithms and our heuristic algorithm can be found in [1].

4 Conclusion

Any multi-constrained routing problem which involves two additive weight functions such as delay and cost is NP-complete. We formalized it as an MCP problem (Definition 1) and proposed a heuristic algorithm with a polynomial time complexity. The algorithm first reduces the problem MCP(G, s, t, w_1, w_2, c_1, c_2) to a simpler one MCP$(G, s, t, w_1, w_2, c_1, x)$, and then uses an extended Dijkstra's (or Bellman-Ford) algorithm to find a solution for the new problem in polynomial time. We showed the correctness of the algorithm by proving that any solution found for the simpler problem must also be a solution for the original problem. We showed the effectiveness of the algorithm by proving that the simpler problem must have a solution if the original problem has a solution p and $w_2(p) \leq (1 - \frac{l-1}{x})c_2$, where l is the length of p and x is an integer given by the algorithm. With an increasing x, the condition $w_2(p) \leq (1 - \frac{l-1}{x})c_2$ is gradually relaxed and approaching the original constraint, $w_2(p) \leq c_2$. The statistical performance of the heuristic algorithm was studied by experiments, which showed that higher performance of the algorithm can be achieved at the expense of higher overhead.

[5] The Jaffe's algorithm finds a solution for *every* pair of nodes in the network whereas our heuristic algorithm does that for a single pair.

[6] The paper [5] used different terminologies and notations from ours. $w_1(p) + d \cdot w_2(p)$ was written as $L(p) + d \cdot W(p)$, where L and W correspond to w_1 and w_2, respectively.

References

[1] S. Chen and K. Nahrstedt. On finding multi-constrained paths. *Tech. Report UIUCDCS-R-97-2026, Dept. of Com. Sci., UIUC*, August 1997.

[2] D. E. Comer. *Internetworking with TCP/IP, Volume I.* Prentice Hall, 1995.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press and McGraw-Hill Book Company, 1990.

[4] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: W.H. Freeman and Co., 1979.

[5] J. M. Jaffe. Algorithms for finding paths with multiple constraints. *Networks*, 14:95–116, 1984.

[6] Q. Ma and P. Steenkiste. Quality-of-service routing with performance guarantees. *IWQoS'97*, May 1997.

[7] Q. Ma, P. Steenkiste, and H. Zhang. Routing high-bandwidth traffic in max-min fair share networks. *Sigcom'96*, August 1996.

[8] J. Moy. Ospf version 2, internet rfc 1583. March 1994.

[9] C. Parris, H. Zhang, and D. Ferrari. Dynamic management of guaranteed performance multimedia connections. *Multimedia Systems Journal*, 1:267–283, 1994.

[10] S. Plotkin. Competitive routing of virtual circuits in atm networks. *IEEE JSAC*, 13:1128–1136, August 1995.

[11] H. F. Salama, D. S. Reeves, and Y. Viniotis. A distributed algorithm for delay-constrained unicast routing. *Infocom'97, Japan*, March 1997.

[12] Quan Sun and Horst Langendorfer. A new distributed routing algorithm with end-to-end delay guarantee. *IWQoS'97*, May 1997.

[13] Z. Wang and Jon Crowcroft. Qos routing for supporting resource reservation. *IEEE JSAC*, September 1996.