

## PAPER

# An Efficient Approximate Algorithm for Finding Paths with Two Additive Constraints\*

Gang FENG<sup>†</sup> and Christos DOULIGERIS<sup>††</sup>, *Nonmembers*

**SUMMARY** The problem of finding a path with two additive constraints, in particular finding a path that satisfies both the cost and the delay constraints, is called multi-constrained path (MCP) problem in the literature. In this paper, we explore the MCP problem based on the idea of single mixed weight—a mixed weight for each link is first obtained by combining its delay and cost, and then Dijkstra's algorithm is used to find the corresponding shortest path. Given two *infeasible* paths, it can be theoretically proved that a better path can possibly be found if we choose an appropriate parameter to construct the mixed weight. An approximate algorithm is thus proposed to solve the MCP problem. Theoretical analysis demonstrates that this algorithm can make a correct judgment whether there is a feasible path or not with a very high probability even in the strict case where the delay bound is between the delays of the least delay path and the least cost path, while the cost bound is between the costs of the two paths. On the other hand, the time complexity of this algorithm is very small since it only needs to execute Dijkstra's algorithm a limited number of times. The excellent performance of the proposed algorithm is verified by a large number of experiments on networks of different sizes.

**key words:** quality of service, QoS routing, multi-constrained path, additive QoS constraints, high-speed networking

## 1. Introduction

The significance of quality of service (QoS) routing has been recognized due to the growing need for routing multimedia messages [2], [11]. QoS requirements are generally specified in terms of constraints, which can be classified into three categories: additive, multiplicative and concave [11]. It has been noted that routing with concave constraints can be easily handled [10], [11], while in the case of multiplicative or additive constraints finding solutions in an optimum and efficient way becomes very difficult [2], [11]. Also, we notice that multiplicative constraints can be converted into additive ones by simply using a logarithmic operation [9].

This paper explores the multi-constrained path (MCP) problem [1], [7], which intends to find a path that satisfies both the end-to-end delay and the end-to-end cost constraints. This problem, proved to be NP-

complete [6], has a number of applications [2]. However, by now only a few papers addressing this problem have been published. In a relatively earlier work [7], Jaffe proposed a pseudopolynomial-time algorithm to solve the MCP problem in time  $O(N^5 b \log Nb)$ , where  $N$  is the number of nodes, while  $b$  is the largest number among all costs and delays. Since this algorithm is definitely impractical, a heuristic algorithm was further presented to minimize a predefined objective function, but its solution could possibly violate the constraints. In a relatively more recent work [1], Chen & Nahrstedt proposed an algorithm that can possibly be used in real-time applications. The basic idea of their algorithm is to convert the delay (or cost) of each link to an integer, and then use an "extended Dijkstra's algorithm" (EDSP) or "extended Bellman-Ford algorithm" to solve the simplified problem. The time complexity of this algorithm is  $O(x^2 N^2)$ , where  $x$  is an integer defined as the upper bound on the path's converted delay (or converted cost). The performance of this algorithm relies heavily on the value of  $x$ . The larger the value of  $x$ , the better the solution, while the longer the execution time. Another important work regarding the MCP problem is the heuristic H-MCOP proposed by Kormaz & Krunk [14]. H-MCOP was intended to solve the generalized multi-constrained optimal-path (MCOP) problem taking into account two or more constraints. Simulations in [14] show that in the case with two constraints it can achieve a very high success ratio of finding a feasible solution if such a solution exists. In addition, this algorithm only needs to run Dijkstra's algorithm [4] (with some modifications) twice. Thus, it has been recognized as the best algorithm for the MCP problem so far.

In this paper, we present an approximate algorithm for solving the MCP problem based on an idea different from previous methods. This idea includes two basic steps. First, the delay and cost for each link is combined in terms of a parameter to form a single mixed weight. Second, Dijkstra's algorithm is employed to find the corresponding shortest path. Given two infeasible paths, one not satisfying the delay constraint while the other not satisfying the cost constraint, we can prove that using the above idea the obtained shortest path could be better than (at worst remains the same as) one of the two infeasible paths as long as the parameter is appropriately chosen. Thus, we can keep

Manuscript received April 3, 2001.

Manuscript revised December 17, 2001.

<sup>†</sup>The author is with the Department of Electrical and Computer Engineering, University of Miami, Coral Gables, FL 33124-0640, USA.

<sup>††</sup>The author is with the Department of Informatics, University of Piraeus, Piraeus, 18534, Greece.

\*This work is supported in part by NSF under grant No. 9810534 and the Greek GSRT under PENED grant.

updating one of the two infeasible paths until a feasible path is found or a conclusion that no feasible path is available has been made. A large number of computer simulations demonstrate that this algorithm can converge very fast since it only needs to execute Dijkstra's algorithm a few times even for a network of relatively large size. On the other hand this algorithm can make a correct judgment whether there is a feasible path or not with a very high probability.

There are two fundamental differences between the proposed algorithm and H\_MCOP in [14]. First, the aggregate (i.e., mixed) weight used in the proposed algorithm is a linear combination of delay and cost, while the one in H\_MCOP is a nonlinear function of delay and cost. Second, the proposed algorithm uses the standard Dijkstra's algorithm rather than a modified Dijkstra's algorithm as in H\_MCOP. From the experimental results shown in Sect. 4 we can see that the performance of the proposed algorithm is comparable with that of H\_MCOP.

The rest of the paper is organized as follows. The problem formulation is first defined in Sect. 2. Then, the basic idea of single mixed weight and the approximate algorithm are elaborated in Sect. 3. Also, an example is given to show how this algorithm works. Section 4 provides the performance analysis of the proposed algorithm, while Sect. 5 is the corresponding experimental results when it is tested on networks of different sizes. Section 6 concludes the paper.

## 2. Notation and Problem Formulation

A *network* is represented by a directed graph  $G(V, E)$ , where  $V$  is the set of nodes, and  $E$  is the set of links. Assume  $N = |V|$ , and  $M = |E|$ .

A *weight*  $w$  defines a number  $w(e) \in R_0^+$  associated with each link  $e$ , i.e.,  $w : E \rightarrow R_0^+$ . In particular, weight  $d : E \rightarrow R_0^+$  is called *delay*, while  $c : E \rightarrow R_0^+$  is called *cost*.

A *path* is a finite sequence of non-repeated nodes  $\mathbf{p} = (v_0, v_1, \dots, v_k)$ , such that, for  $0 \leq i < k$ , there exists a link from  $v_i$  to  $v_{i+1}$ , i.e.,  $(v_i, v_{i+1}) \in E$ . A link  $e \in \mathbf{p}$  means that path  $\mathbf{p}$  passes through link  $e$ .

A weight  $w$ , like delay or cost, is *additive* if the weight of a path  $\mathbf{p}$  is equal to the summation of the weights of all links belonging to that path, i.e.,  $w(\mathbf{p}) = \sum_{e \in \mathbf{p}} w(e)$ . In particular, the delay and cost of a path  $\mathbf{p}$  are given by  $D_{\mathbf{p}} = d(\mathbf{p}) = \sum_{e \in \mathbf{p}} d(e)$  and  $C_{\mathbf{p}} = c(\mathbf{p}) = \sum_{e \in \mathbf{p}} c(e)$ , respectively.

In a general sense, the delay of a link is the average transmission time on that link, while the cost of a link may be the fee paid to transmit per unit of message on that link. However, the delay and cost may be redefined as other metrics such as jitter, loss, etc. as long as they are additive.

Given two additive weights  $w_1$  and  $w_2$ , a *mixed weight*  $w = w_1 + \alpha w_2$  means that for any link  $e$ ,  $w(e) =$

$w_1(e) + \alpha w_2(e)$ , where  $\alpha \in R_0^+$ . Apparently, a mixed weight of two additive weights is also additive.

Given a source node  $s$ , a destination node  $t$  and an additive weight  $w$ , we define a *function* (or *procedure*)  $\text{Dijk}(w)$  which returns the shortest path from  $s$  to  $t$  found by the Dijkstra's algorithm when the network is weighted by  $w$ . In particular, we let  $\mathbf{p}_0 = \text{Dijk}(d)$  be the least delay (LD) path,  $\mathbf{p}_1 = \text{Dijk}(c)$  be the least cost (LC) path,  $D_0 = D_{\mathbf{p}_0}$  and  $C_0 = C_{\mathbf{p}_0}$  be the delay and cost of the LD path, and  $D_1 = D_{\mathbf{p}_1}$  and  $C_1 = C_{\mathbf{p}_1}$  be the delay and cost of the LC path. Note that  $D_0 \leq D_1$  and  $C_0 \geq C_1$  always hold.

**Definition 1:** Given a network  $G(V, E)$ , two nodes  $s \in V$  and  $t \in V$ , a delay and a cost for each link, a delay upper bound  $D$ , and a cost upper bound  $C$ , the *multi-constrained path* (MCP) problem is to find a path  $\mathbf{p}$  from  $s$  to  $t$ , such that  $D_{\mathbf{p}} \leq D$  and  $C_{\mathbf{p}} \leq C$ .

For convenience, a path that satisfies both of the two constraints in the above definition will be called a *feasible path* or a *solution*, while a path that does not satisfy at least one of the two constraints will be called an *infeasible path*.

## 3. An Approximate Algorithm for the MCP Problem

In this section, we first propose two theorems, which reflect the basic idea of the mixed weight. Then, based on the second theorem, an approximate algorithm is developed.

### 3.1 Two Basic Theorems

As mentioned in Sect. 1, our basic idea to solve the MCP problem is to first combine the delay and cost to form a single mixed weight, and then use Dijkstra's algorithm to find the corresponding shortest path. To make use of this idea, it is rather important to understand the relationship between the parameter, which is used for constructing the mixed weight, and the cost and delay of the corresponding shortest path.

The following theorem shows how the parameter affects the delay and cost of the resulting shortest path.

**Theorem 1:** If  $\mathbf{p} = \text{Dijk}(d + \alpha c)$ ,  $\mathbf{q} = \text{Dijk}(d + \beta c)$ ,  $\alpha \in R_0^+$ ,  $\beta \in R_0^+$ , then

- (i) if  $\alpha \geq \beta$  then  $C_{\mathbf{p}} \leq C_{\mathbf{q}}$ ,  $D_{\mathbf{p}} \geq D_{\mathbf{q}}$ ,
- (ii)  $C_{\mathbf{p}} = C_{\mathbf{q}}$  iff  $D_{\mathbf{p}} = D_{\mathbf{q}}$ .

**Proof:** (i) Since  $\mathbf{p}$  is the shortest path obtained using Dijkstra's algorithm when the link is weighted by  $d + \alpha c$ , we have

$$D_{\mathbf{p}} + \alpha C_{\mathbf{p}} \leq D_{\mathbf{q}} + \alpha C_{\mathbf{q}}. \quad (1)$$

Similarly, the following inequality holds:

$$D_{\mathbf{q}} + \beta C_{\mathbf{q}} \leq D_{\mathbf{p}} + \beta C_{\mathbf{p}}. \quad (2)$$

By combining the two inequalities, we have

$$\begin{aligned}\alpha C_p &\leq D_q + \alpha C_q - D_p \\ &= D_q + \beta C_q - \beta C_q + \alpha C_q - D_p \\ &\leq D_p + \beta C_p - \beta C_q + \alpha C_q - D_p \\ &= \beta C_p + (\alpha - \beta) C_q.\end{aligned}$$

Therefore,  $C_p \leq C_q$  holds. Furthermore, from inequality (2), we have

$$D_p \geq D_q + \beta(C_q - C_p) \geq D_q.$$

(ii) In this case, inequalities (1) and (2) still hold. If  $C_p = C_q$ , then from inequality (1) we have  $D_p \leq D_q$ , while from inequality (2) we have  $D_p \geq D_q$ . Therefore,  $D_p = D_q$  holds. Similarly we can prove the inverse case.  $\square$

**Corollary 1:** If  $p = \text{Dijk}(d + \alpha c)$ , where  $0 < \alpha < +\infty$ , then  $C_0 \geq C_p \geq C_1$ ,  $D_0 \leq D_p \leq D_1$ .

Theorem 1 implies that the tradeoff between the cost and delay can be achieved by appropriately choosing the parameter used for computing the mixed weight. The larger the parameter chosen, the smaller the cost of the obtained shortest path, while the larger the delay of the path.

If we assume that a parameter  $\alpha$  results in an infeasible path  $p$ , which satisfies the delay constraint while does not satisfy the cost constraint, then according to Theorem 1 we can possibly find a feasible path by choosing a parameter slightly greater than  $\alpha$ . Similarly, if  $p$  satisfies the cost constraint while does not satisfy the delay constraint, then we can also possibly find a feasible path by choosing a parameter slightly smaller than  $\alpha$ . Actually this is the basic idea of the algorithm to be proposed in this paper, and the following theorem provides a theoretical basis for our algorithm.

**Theorem 2:** If  $r = \text{Dijk}(d + \alpha c)$ ,  $p = \text{Dijk}(d + \beta c)$ ,  $q = \text{Dijk}(d + \gamma c)$ , where  $\beta < \gamma$ ,  $C_p \neq C_q$ ,  $\alpha = (D_q - D_p)/(C_p - C_q)$ , then

- (i)  $C_p \geq C_r \geq C_q$ ,  $D_p \leq D_r \leq D_q$ ,
- (ii)  $D_r + \alpha C_r = D_p + \alpha C_p$  iff  $D_r + \alpha C_r = D_q + \alpha C_q$ ,
- (iii)  $D_r + \alpha C_r = D_p + \alpha C_p$  iff there does not exist a path  $h$  such that  $C_h < C_p$  and  $\alpha > (D_h - D_p)/(C_p - C_h)$ .

**Proof:** (i) From Theorem 1, we only need to prove that  $\beta \leq \alpha \leq \gamma$ . Since  $p = \text{Dijk}(d + \beta c)$ , we have

$$D_p + \beta C_p \leq D_q + \beta C_q. \quad (3)$$

Moreover,  $\beta < \gamma$  implies that  $C_p \geq C_q$  and  $D_p \leq D_q$ . Notice  $C_p \neq C_q$ , therefore, inequality (3) can be rewritten as

$$\beta \leq \frac{D_q - D_p}{C_p - C_q} = \alpha.$$

Similarly, we can prove  $\alpha \leq \gamma$ .

(ii) This is obvious since we have  $D_p + \alpha C_p = D_q + \alpha C_q$ .

(iii) We need to prove that if there exists such a path  $h$ , then  $h$  should be the shortest path  $r$  whose mixed weight is less than that of  $p$  and  $q$ , i.e., the following two inequalities hold:

$$\begin{aligned}D_h + \alpha C_h &< D_p + \alpha C_p, \\ \text{and } D_h + \alpha C_h &< D_q + \alpha C_q.\end{aligned}$$

The first inequality is a direct result from the assumption

$$\alpha > \frac{D_h - D_p}{C_p - C_h}.$$

The second inequality holds due to the fact that  $D_p + \alpha C_p = D_q + \alpha C_q$ .  $\square$

### 3.2 An Approximate Algorithm

To understand the significance of Theorem 2, let us consider the following situation. Assume  $D_p \leq D$ ,  $C_p > C$ ,  $D_q > D$ , and  $C_q \leq C$ . This means that both  $p$  and  $q$  are infeasible paths. However, if we use a parameter  $\alpha$  as defined in Theorem 2 to construct a mixed weight for each link, and find the corresponding shortest path  $r$ , then  $r$  could be “better” than  $p$  or  $q$ . More precisely, if  $D_r \leq D$ , then  $r$  could be better than  $p$  in the sense that its cost is possibly lower. On the other hand, if  $C_r \leq C$ , then  $r$  could be better than  $q$  in the sense that its delay is possibly lower.

Based on the above analysis, we can propose an iterative algorithm for the MCP problem as described in Fig.1. In this algorithm, the LC path  $p_1$  and the LD path  $p_0$  are first identified. If  $C_1 > C$  or  $D_0 > D$ , then no feasible path is available, and the QoS requirements must be renegotiated. Otherwise, if  $D_1 \leq D$  (or

```

Algorithm MCP-IA( $G, s, t, c, d, C, D$ )
1   $q \leftarrow \text{Dijk}(c)$ 
2  if ( $c(q) > C$ ) then
3    return NULL
4  else if ( $d(q) \leq D$ ) then
5    return  $q$ 
6   $p \leftarrow \text{Dijk}(d)$ 
7  if ( $d(p) > D$ ) then
8    return NULL
9  else if ( $c(p) \leq C$ ) then
10   return  $p$ 
11  while TRUE do
12    $\alpha \leftarrow [d(q) - d(p)]/[c(p) - c(q)]$ 
13    $r \leftarrow \text{Dijk}(d + \alpha c)$ 
14   if ( $c(r) = c(q)$  or  $c(r) = c(p)$ ) then
15     return NULL
16   else if ( $c(r) > C$ ) then
17      $p \leftarrow r$ 
18   else if ( $d(r) > D$ ) then
19      $q \leftarrow r$ 
20   else
21     return  $r$ 

```

**Fig. 1** Algorithm MCP-IA.

$C_0 \leq C$ ), then  $\mathbf{p}_1$  (or  $\mathbf{p}_0$ ) must be a solution. If none of the above conditions happens, then the algorithm sets  $\mathbf{p}_0$  and  $\mathbf{p}_1$  to  $\mathbf{p}$  and  $\mathbf{q}$ , respectively, as their initial values. In each of successive iterations, either  $\mathbf{p}$  or  $\mathbf{q}$  is updated with a better path. From the third conclusion of Theorem 2, if the mixed weight of the newly obtained path is the same as that of  $\mathbf{p}$  (or  $\mathbf{q}$ ), then the algorithm can not find a better path with a mixed weight less than that of  $\mathbf{p}$  (or  $\mathbf{q}$ ). If this happens, the algorithm should terminate. In the algorithm described in Fig. 1, however, a stronger termination rule<sup>†</sup> is used to avoid the case that the newly obtained path is a better one, but it has the same mixed weight as  $\mathbf{p}$  (or  $\mathbf{q}$ ).

### 3.3 Illustration of the Proposed Algorithm

Now we illustrate through two examples how the algorithm MCP-IA tries to find a solution.

Figure 2 shows an MCP problem as described in [1], for which we need to find a feasible path from  $s$  to  $t$  such that the delay is bounded by 8, while the cost is bounded by 20. Using the EDSP proposed in [1], the parameter  $x$  must take a value around 10 in order to find the only feasible path  $s \rightarrow u \rightarrow v \rightarrow t$ , and the corresponding time complexity of EDSP is about 100 times the time complexity of Dijkstra's algorithm. In contrast, the proposed algorithm MCP-IA only needs to run Dijkstra's algorithm once to find the feasible path. This is because MCP-IA first finds the LC path, which is  $s \rightarrow u \rightarrow v \rightarrow t$ . Since the LC path is feasible, the algorithm will terminate at line 5 as shown in Fig. 1.

Now let us investigate a more complicated example shown in Fig. 3, for which we need to find a feasible path from node 1 to node 6 with delay upper bound  $D = 12$  and cost upper bound  $C = 8$ . MCP-IA first finds the LC path  $\mathbf{p}_1 = \{1 \rightarrow 4 \rightarrow 6\}$ . It is easy to see that  $C_1 = 2 < C$  and  $D_1 = 18 > D$ . Therefore, the algorithm continues to find the LD path  $\mathbf{p}_0 = \{1 \rightarrow 3 \rightarrow 6\}$ . Correspondingly, we have

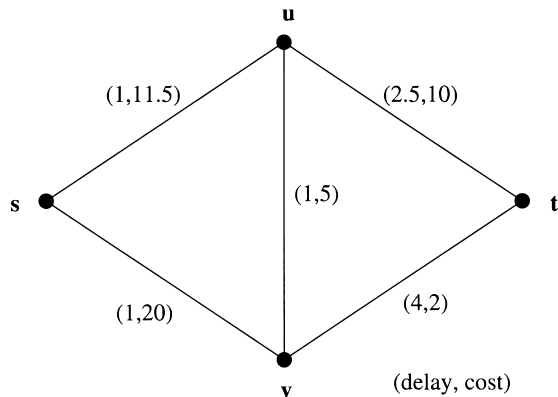


Fig. 2 A MCP problem described by Chen.

$C_0 = 28 > C$  and  $D_0 = 2 < D$ . After that, the algorithm enters an iterative procedure. In each iteration, the parameter  $\alpha$  for constructing the mixed weight is computed in terms of  $\alpha = (D_q - D_p)/(C_p - C_q)$ , and then the corresponding shortest path  $\mathbf{r}$  is found using Dijkstra's algorithm. If  $\mathbf{r}$  is feasible, then it is returned and the algorithm is terminated; otherwise either  $\mathbf{p}$  or  $\mathbf{q}$  is replaced by  $\mathbf{r}$  as long as  $\mathbf{r}$  is better.

The initial values of  $\mathbf{p}$  and  $\mathbf{q}$  are  $\mathbf{p}_0$  and  $\mathbf{p}_1$ , respectively. Thus, the parameter in the first iteration is  $\alpha = (18 - 2)/(28 - 2) = 8/13$ . As shown in Fig. 4(a), the corresponding shortest path is  $\mathbf{r} = \{1 \rightarrow 5 \rightarrow 6\}$ . Since  $\mathbf{r}$  is infeasible and better than  $\mathbf{p}$  in the sense that  $\mathbf{r}$  has smaller cost,  $\mathbf{p}$  is replaced by  $\mathbf{r}$ . In the second iteration, the parameter becomes  $\alpha = (18 - 7)/(9 - 2) = 11/7$ . As shown in Fig. 4(b), the corresponding shortest path is  $\mathbf{r} = \{1 \rightarrow 2 \rightarrow 6\}$ . Again,  $\mathbf{r}$  is infeasible, but it is better than  $\mathbf{q}$  in the sense that  $\mathbf{r}$  has smaller delay. Thus,  $\mathbf{q}$  is replaced by  $\mathbf{r}$ . In the third iteration, the parameter becomes  $\alpha = (14 - 7)/(9 - 3) = 7/6$ . As shown in Fig. 4(c), the corresponding shortest path is  $\mathbf{r} = \{1 \rightarrow 4 \rightarrow 5 \rightarrow 6\}$ , which has a delay of 10 and a cost of 6. Since  $\mathbf{r}$  is feasible, it is returned as the final solution and the algorithm stops.

It is possible that MCP-IA can not find a solution even if there exists one. For instance, in the problem shown in Fig. 3, if we assume that the delay upper bound  $D = 9$  and other parameters keep unchanged, then after finding the path  $\mathbf{r} = \{1 \rightarrow 4 \rightarrow 5 \rightarrow 6\}$  in the third iteration,  $\mathbf{q}$  is again replaced by  $\mathbf{r}$  since  $\mathbf{r}$  is still infeasible but it is better than  $\mathbf{q}$ . In the fourth iteration, the parameter becomes  $\alpha = (10 - 7)/(6 - 3) = 1$ . As shown in Fig. 4(d), the two paths  $1 \rightarrow 5 \rightarrow 6$  and  $1 \rightarrow 4 \rightarrow 5 \rightarrow 6$  have the same minimum length. The returning of any of them will terminate the algorithm. As a result, the algorithm can not find a feasible path.

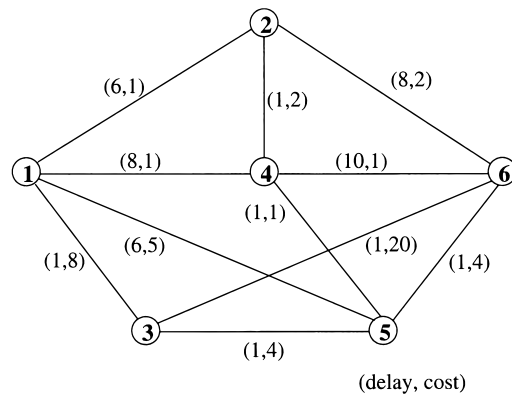


Fig. 3 Another MCP problem.

<sup>†</sup>From the second conclusion of Theorem 1, if  $C_r = C_p$  or  $C_r = C_q$  then the mixed weight of  $\mathbf{r}$  must equal to that of  $\mathbf{p}$  or  $\mathbf{q}$ .

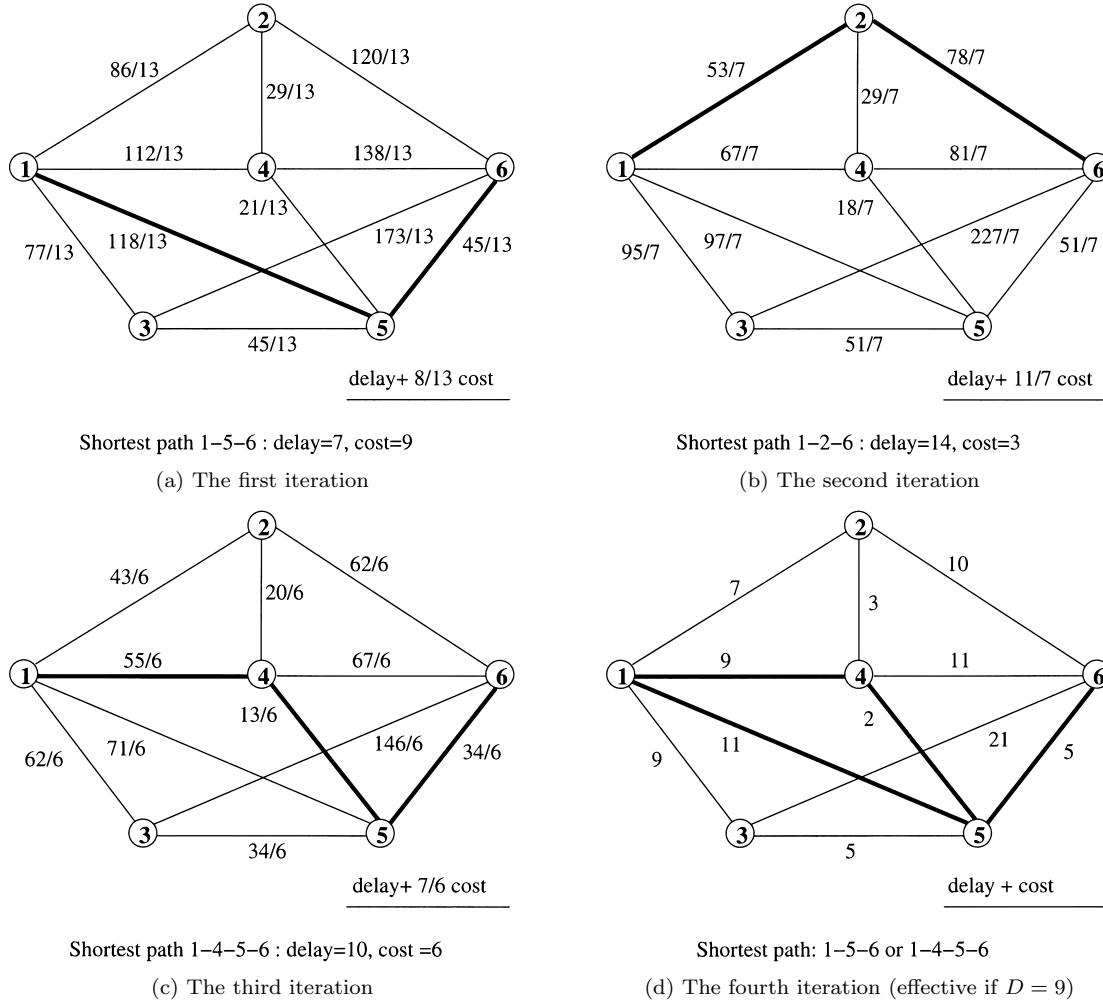


**Fig. 4** The shortest path found in each iteration for the problem shown in Fig. 3.

However, we can see that there do exist a feasible path  $1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6$  for this problem.

#### 4. Performance Analysis

##### 4.1 Quality of Solution

The MCP problem only cares whether we can find a solution. If there are multiple solutions, then for any two solutions we can not identify which one is better. This is based on the assumption that the importance of the cost and that of the delay can not be compared. If we have predetermined that one of them is more important than the other, then the problem becomes the restricted shortest path problem [8], for which we need to use different algorithms to solve<sup>†</sup>. Therefore, if a specific algorithm used to solve a MCP problem returns a solution, then we are done. However, if the algorithm makes a judgment that no feasible path is available, then we might doubt whether this judgment is correct or not. Obviously, for the proposed algorithm, only in the case when  $D_0 < D < D_1$ ,  $C_0 > C > C_1$  and a

judgment that no feasible paths is available has been made will we suspect the correctness of the judgment. Note that in our algorithm described in Fig. 1, if the algorithm stops at line 15, then it makes a judgment that there is no feasible path available.

**Theorem 3:** Consider the case when  $D_0 < D < D_1$ ,  $C_0 > C > C_1$  and algorithm MCP-IA makes a judgment that no feasible path is available. If we assume that

- (1)  $S$  is the set of paths with delays no greater than  $D_1$  and costs no greater than  $C_0$ ,
- (2) the delays and costs of the paths in  $S$  are uniformly distributed in  $[D_0, D_1]$  and  $[C_1, C_0]$ , respectively, and
- (3)  $\alpha = (D_q - D_p)/(C_p - C_q)$  is the last parameter when the algorithm terminates,

then, the following holds:

<sup>†</sup>The idea of single mixed weight can be also used to solve the restricted shortest path problem. Please refer to our recent work [5].

- (i) if  $\alpha > (D - D_p)/(C_p - C)$ , the judgment that no feasible path is available is absolutely correct,
- (ii) if  $\alpha = (D - D_p)/(C_p - C)$ , and there exists a feasible path, it must have a delay and a cost exactly equal to  $D$  and  $C$ , respectively, and
- (iii) if  $\alpha < (D - D_p)/(C_p - C)$ , the probability that there exists a feasible path is  $L^2/(2\alpha A)$ , where  $L = \alpha(C - C_q) - (D_q - D)$ ,  $A = (C - C_1) \cdot (D - D_0)$ .

**Proof:** (i) If  $\alpha > (D - D_p)/(C_p - C)$ , we can prove that there does not exist a feasible path. This is because if there exists a feasible path  $\mathbf{r}$ , which implies that  $C_r \leq C$  and  $D_r \leq D$ , then the following relation must hold:

$$\alpha > \frac{D - D_p}{C_p - C} \geq \frac{D_r - D_p}{C_p - C_r},$$

which is in contradiction with the third conclusion of Theorem 2.

(ii) In this case, if there exists a feasible path  $\mathbf{r}$  with  $D_r \neq D$  or  $C_r \neq C$  or both, then we must also have the relation that  $\alpha > (D_r - D_p)/(C_p - C_r)$ , which is also in contradiction with the third conclusion of Theorem 2.

(iii) This case is shown in Fig. 5. Any feasible path  $\mathbf{r}$  must have its coordinates  $(C_r, D_r)$  located in the rectangular area. However, when the algorithm terminates with a judgment that there is no feasible path, the only possible solution  $\mathbf{r}$  must have its coordinates  $(C_r, D_r)$  located in the shaded area. Since the delays and costs of the feasible paths are assumed to be uniformly distributed, the probability that there is a feasible path is actually the ratio of the area of the shaded region to the area of the rectangular region, which is given by

$$\frac{L^2/(2\alpha)}{(C - C_1)(D - D_0)} = \frac{L^2}{2\alpha A},$$

where  $L$  can be calculated by the relation  $\alpha = (L + D_q - D)/(C - C_q)$ .  $\square$

**Corollary 2:** If algorithm MCP-IA can not find a solution, then the maximum probability that there exists a feasible path is less than  $1/2$ .

**Proof:** From Fig. 5, we can find out that the following two relations must hold:  $L < D - D_0$  and  $L/\alpha < C - C_1$ . Therefore, we have  $L^2/(2\alpha A) < 1/2$ .  $\square$

#### 4.2 Time Complexity

Algorithm MCP-IA only includes two major operations in each iteration. The first one is the update of the mixed weight, which can be finished with a negligible time complexity  $O(M)$ , while the second one is the use of Dijkstra's algorithm to find the shortest path in time  $O(N^2)$ <sup>†</sup>. Therefore, the time complexity of algorithm MCP-IA, as another important performance measure, can be given by  $O(kN^2)$ , where  $k$  is the number of

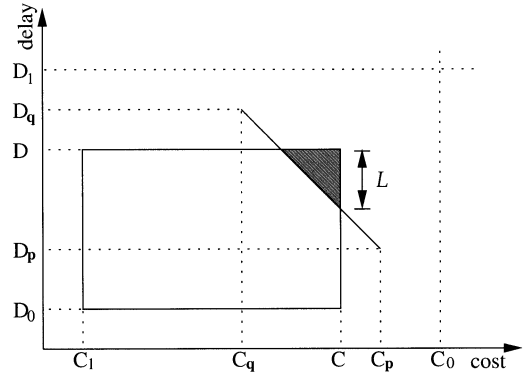


Fig. 5 Illustration of possible solutions.

the executions of Dijkstra's algorithm. Although  $k$  is not deterministic, the computer simulations in the next section show that the maximum number of executions is small even for relatively large networks.

#### 5. Experimental Results

For a routing request between two nodes on a specific network, both the proposed algorithm and an exact algorithm [13] are used to search for a solution. A value of 1 is returned if a feasible path is found, otherwise a value of 0 is returned. Experiments on a specific network model are divided into groups, each of which includes 10,000 experiments. When we start a group of experiments, the delays and costs of links are randomly generated and uniformly distributed in  $(0, 50]$  and  $(0, 200]$ , respectively, like the method used in [1]. For all the experiments of a specific group, the delays and costs of links keep unchanged. For a specific experiment, a source node and a destination node are randomly generated. Three performance measures are investigated after a group of experiments are finished:

- *Execution time*, which is the time consumed by the experiments in this group when the experiments are done on a HP Celeron 450 MHz PC,
- *Average number of the executions* of Dijkstra's algorithm in one experiment,
- *Maximum number of the executions* of Dijkstra's algorithm among all experiments in this group,
- *Optimality*, which is the percentage of experiments in which the value returned by the proposed algorithm equals to the one returned by the exact algorithm.

##### 5.1 Performance Comparison between MCP-IA and EDSP

In order to make a comparison between algorithm

<sup>†</sup>If a Fibonacci heap [3] is used in the priority queue, the time complexity of Dijkstra's algorithm becomes  $O(N \log N + M)$ .

MCP-IA and the EDSP proposed by Chen & Nahrstedt in [1], we repeated their experiments. The network model as shown in Fig. 6 includes 32 nodes and 54 duplex links.

Table 1 shows the performance measures of both MCP-IA and EDSP when they are used to test this network model. The value of parameter  $x$  in EDSP is set to 10. Let us take the third column in Table 1 as an example to explain its meaning. For this column, a group of experiments, which includes 10,000 routing requests, has been done. The values of  $D$  and  $C$  are uniformly distributed in  $[50, 65]$  and  $[200, 260]$ , respectively. The performance measures of MCP-IA are as follows: the execution time is 1.54 seconds, the average number of executions of Dijkstra's algorithm is 1.4754, the maximum number of executions is 5, and the optimality is 99.92%. In contrast, the execution time and the optimality of EDSP are 14.99 seconds and 27.51%, respectively.

By comparing the performance measures of MCP-IA and those of EDSP, we notice that in certain cases the optimality of EDSP is unsatisfactorily low. It can achieve a high optimality only when the cost and delay upper bounds are relatively large, in which case it is

easy to find a feasible path. In contrast, MCP-IA can always achieve a high optimality. On the other hand, the execution time of EDSP is much longer than the time MCP-IA takes. If one wants to increase the optimality of EDSP by picking a larger value for  $x$ , its time complexity will become even higher.

## 5.2 Performance Comparison between MCP-IA and H\_MCOPI

We have also done experiments to compare the performance of MCP-IA and H\_MCOPI. In this case the same method as in [14] is adopted to generate the delay and cost upper bounds, i.e.,  $D$  and  $C$  are uniformly distributed in  $[0.8 * D_1, 1.2 * D_1]$  and  $[0.8 * C_0, 1.2 * C_0]$ , respectively. Four types of topologies have been tested. The first one is the 32-node network shown in Fig. 6. The other three networks, i.e., 50-node, 100-node and 200-node, which include 313 links, 683 links, and 1762 links, respectively, are generated using Waxman's method [12].

Similarly, both MCP-IA and H\_MCOPI are employed to process 10,000 routing requests for each network topology. The statistical results are shown in Table 2. Comparing these results, we can conclude that the performance of MCP-IA is at least comparable with the performance of H\_MCOPI (when both of them are used to solve the MCP problem with two constraints). First of all, even though there is a slight difference between the optimality of MCP-IA and that of H\_MCOPI, the difference is small enough to be negligible. Second, the execution time of H\_MCOPI is slightly higher than that of MCP-IA. There are several reasons that contribute to the higher time complexity of H\_MCOPI. First, its nonlinear formulation of the aggregate weight requires more computation time than a linear one as adopted by MCP-IA. Second, the modified Dijkstra's algorithm described in H\_MCOPI needs to maintain

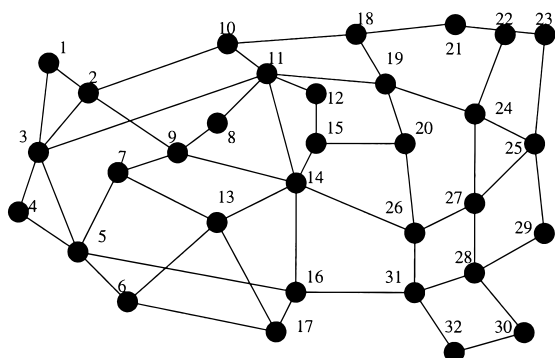


Fig. 6 A 32-node network.

Table 1 Performance comparison between MCP-IA and EDSP.

| Algorithm | Delay bound<br>Cost bound | $D \in [50, 65]$<br>$C \in [200, 260]$ | $D \in [75, 90]$<br>$C \in [300, 360]$ | $D \in [100, 115]$<br>$C \in [400, 460]$ | $D \in [125, 140]$<br>$C \in [500, 560]$ | $D \in [150, 165]$<br>$C \in [600, 660]$ |
|-----------|---------------------------|----------------------------------------|----------------------------------------|------------------------------------------|------------------------------------------|------------------------------------------|
| MCP-IA    | Execution time (seconds)  | 1.54                                   | 1.65                                   | 1.49                                     | 1.43                                     | 1.32                                     |
|           | Average execution number  | 1.4754                                 | 1.6119                                 | 1.4616                                   | 1.3381                                   | 1.2216                                   |
|           | Maximum execution number  | 5                                      | 6                                      | 5                                        | 5                                        | 6                                        |
|           | Optimality                | 99.92%                                 | 99.63%                                 | 99.34%                                   | 99.68%                                   | 99.74%                                   |
| EDSP      | Execution time (seconds)  | 14.99                                  | 24.55                                  | 29.6                                     | 32.79                                    | 36.2                                     |
|           | Optimality                | 27.51%                                 | 60.15%                                 | 72.25%                                   | 93.7%                                    | 99.86%                                   |

Table 2 Performance comparison between MCP-IA and H\_MCOPI.

| Algorithm | Network                  | 32-node | 50-node | 100-node | 200-node |
|-----------|--------------------------|---------|---------|----------|----------|
| MCP-IA    | Execution time (seconds) | 1.59    | 3.79    | 19.01    | 101.23   |
|           | Average execution number | 1.4629  | 1.6413  | 1.7064   | 1.7113   |
|           | Maximum execution number | 5       | 5       | 5        | 5        |
|           | Optimality               | 99.53%  | 98.04%  | 98.24%   | 98.19%   |
| H_MCOPI   | Execution time (seconds) | 3.19    | 8.84    | 33.06    | 150.72   |
|           | Optimality               | 99.14%  | 98.96%  | 98.97%   | 99.27%   |

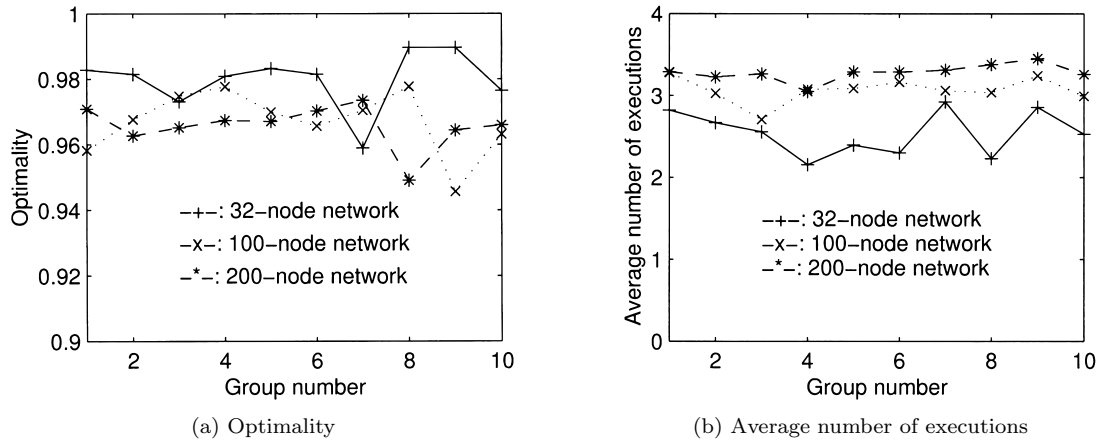


Fig. 7 Performance measures for different networks when  $D \in [D_0, D_1]$  and  $C \in [C_1, C_0]$ .

more state variables than the standard Dijkstra's algorithm. Third, H\_MCOPI always runs the modified Dijkstra's algorithm twice; In contrast, MCP-IA may only run the standard Dijkstra's algorithm once if the LC path is feasible.

### 5.3 Performance of MCP-IA in the Strictest Case

In the experiments conducted in the previous two subsections, the values of  $D$  and  $C$  are generated in arbitrarily chosen areas. If the values of  $D$  and  $C$  are too small, then in most cases algorithm MCP-IA will execute Dijkstra's algorithm only once or twice because in such cases there is no feasible path available. For instance, in the third column of Table 1  $D$  and  $C$  are relatively small, the average number of executions is only 1.4754. Similarly, If the values of  $D$  and  $C$  are too big, then in most cases the LD or LC path will be a feasible path, and algorithm MCP-IA will also run Dijkstra's algorithm only once or twice. For instance, in the last column of Table 1  $D$  and  $C$  are relatively large, the average number of executions is only 1.2216.

To test the algorithm in the strictest case, in subsequent experiments we let the values of  $D$  and  $C$  be uniformly distributed in  $[D_0, D_1]$  and  $[C_1, C_0]$ , respectively. This method can ensure that the case where the LD path or LC path is a solution will not happen often.

The 32-, 100-, and 200-node networks described in previous subsections are tested. For each network model we have done 10 groups of experiments.

The corresponding performance measures are in shown Fig. 7 and Table 3. The execution time is not given since it is proportional to the average number of executions of Dijkstra's algorithm. In Fig. 7, each point corresponds to a group of experiments. From Fig. 7(a) we can see that the optimality is satisfactorily high either for a network of relatively small size or for a relatively large scale network. From Fig. 7(b), we can see that the average number of executions is between 2 and 4 for the three networks, while from Table 3 we

Table 3 Maximum number of executions for different networks when  $D \in [D_0, D_1]$  and  $C \in [C_1, C_0]$ .

| Group No. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|-----------|---|---|---|---|---|---|---|---|---|----|
| 32-node   | 6 | 5 | 4 | 4 | 4 | 6 | 6 | 5 | 5 | 5  |
| 100-node  | 7 | 6 | 6 | 6 | 7 | 6 | 6 | 6 | 6 | 6  |
| 200-node  | 7 | 6 | 7 | 7 | 6 | 6 | 7 | 7 | 7 | 6  |

may see that the maximum number of executions for each group is between 4 and 7, which is very small. Moreover, although the average and maximum numbers of executions are increasing with the increase of the network size, the rate of increase is also very small.

## 6. Conclusion

We have demonstrated through theoretical analysis and computer simulations that the approximate algorithm proposed in this paper can very well solve the MCP problem, which is very important in the QoS unicast routing area. This algorithm tries to find a solution by performing Dijkstra's algorithm a limited number of times. A large number of computer simulations show that the maximum number of the executions of Dijkstra's algorithm is satisfactorily small even for relatively large networks, e.g., 7 for a 200-node network. On the other hand, the probability that the algorithm makes a correct judgment whether there is a solution or not is very high, e.g., over 95% for a 200-node network. The good performance of the proposed algorithm has also been verified by comparing it with previous works.

The MCP problem discussed in this paper only takes into account two weights, i.e., delay and cost. It is very likely to develop similar heuristics for the MCP problem with three or more weights. In such cases, we need to use more parameters to construct the mixed weight  $w$ , e.g.,  $w = w_0 + \alpha_1 w_1 + \dots + \alpha_n w_n$ . If the shortest path w.r.t.  $w$  violates the constraint corresponding to weight  $w_k$ , the corresponding parameter  $\alpha_k$  can be adjusted in a way similar to the one described in this paper.

## Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments.

## References

- [1] S. Chen and K. Nahrstedt, "On finding multi-constrained paths," ICC'98, pp.874–879, Atlanta, Georgia, June 1998.
- [2] S. Chen and K. Nahrstedt, "An overview of quality of service routing for next-generation high-speed networks: Problems and solutions," IEEE Network, pp.64–79, Nov./Dec. 1998.
- [3] T. Cormen, C.E. Leiserson, and R.L. Rivest, Introduction to Algorithms, The MIT Press, 1985.
- [4] E. Dijkstra, "A note on two problems in connexion with graphs," Numerische Mathematik, vol.1, pp.269–271, 1959.
- [5] G. Feng and C. Douligeris, "Fast algorithms for delay-constrained least-cost unicast routing," shortened version presented in INFORM'2001, Miami Beach, Nov. 2001, available at <http://www.students.miami.edu/~gfeng>.
- [6] M.R. Garey and D.S. Johnson, Computers and Intractability, A Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979.
- [7] J.M. Jaffe, "Algorithms for finding paths with multiple constraints," Networks, vol.14, pp.95–116, 1984.
- [8] R. Hassin, "Approximation scheme for the restricted shortest path problem," Mathematics of Operation Research, vol.17, no.1, pp.36–42, Feb. 1992.
- [9] D.H. Lorenz and A. Orda, "Optimal partition of QoS requirements on unicast paths and multicast trees," INFOCOM'99, vol.1, pp.246–253, New York, March 1999.
- [10] A. Orda and A. Sprintson, "QoS routing: The precomputation perspective," INFOCOM'2000, vol.1, pp.128–136, Israel, March 2000.
- [11] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," IEEE J. Sel. Areas Commun., vol.14, no.7, pp.1288–1234, Sept. 1996.
- [12] B.M. Waxman, "Routing of multipoint connections," IEEE J. Sel. Areas Commun., vol.6, no.9, pp.1617–1622, Dec. 1988.
- [13] R. Widryono, "The design and evaluation of routing algorithms for real-time channels," Technol Report ICSI TR-94-024, University of California at Berkley, International Computer Science Institute, June 1994.
- [14] T. Korkmaz and M. Krunz, "Multi-constrained optimal path selection," INFOCOM'2001, vol.2, pp.834–843, Alaska, 2001.



**Gang Feng** received his B.E. and M.E. from University of Electronic Science and Technology of China in 1992 and 1995 respectively, and Dr.Eng. from Beijing University of Posts and Telecommunications in 1998, all in electrical engineering. Between 1998 and 2001, he was studying in University of Miami, where he was granted Ph.D. in December 2001. His major research interests include routing in high-speed networks, combinatorial optimization, feedback neural network and evolutionary computation.



**Christos Douligeris** was born in Melissopetra Arcadia Greece. He received his Diploma in Electrical Engineering from the National Technical University of Athens in 1984 and the M.S., M.Phil. and Ph.D. degrees from Columbia University in 1985, 1987, 1990 respectively. He has held positions with the Department of Electrical and Computer Engineering at the University of Miami, where he reached the rank of associate professor and was the associate director for engineering of the Ocean Pollution Research Center. He has also held visiting positions at the National Technical Univ. of Athens and the Univ. of Piraeus, Greece. He has served in technical program committees of several conferences. His main technical interests lie in the areas of performance evaluation of high speed networks, neurocomputing in networking, resource allocation in wireless networks and information management, risk assessment and evaluation for emergency response operations. He is an editor of the IEEE Communications Letters and the IEEE Network, a technical editor of the IEEE Communications Magazine, a technical editor of the IEEE Communications Magazine Interactive, an associate editor of Computer Networks, a senior member of IEEE, and a member of INFORMS (formerly ORSA) and the Technical Chamber of Greece. During the academic year 1999–2000 he worked in the National Greek Agricultural Foundation as a Senior Fulbright Scholar.