Multi-postpath-based lookahead multiconstraint QoS routing

Dong-won Shin\textsuperscript{a}, Edwin K.P. Chong\textsuperscript{b,c,*}, Howard Jay Siegel\textsuperscript{b,d}

\textsuperscript{a}Samsung SDS, Seoul 135-798, Republic of Korea
\textsuperscript{b}Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523-1373, USA
\textsuperscript{c}Department of Mathematics, Colorado State University, USA
\textsuperscript{d}Department of Computer Science, Colorado State University, USA

Received 29 December 2009; received in revised form 2 January 2012; accepted 4 January 2012
Available online 12 January 2012

Abstract

Multiconstraint QoS (quality of service) routing is an essential mechanism for QoS-guaranteed services. Unfortunately, the multiconstraint QoS routing problem is NP-complete. In this paper, we propose a heuristic multiconstraint QoS routing scheme, MPLMR (multi-postpath-based lookahead multiconstraint routing). MPLMR is a routing scheme using an extended shortest-path algorithm. As in previous schemes such as TAMCRA and H_MCOP, MPLMR stores a limited number of subpaths between the source node and each intermediate node, and extends these subpaths toward the destination node. However, MPLMR uses an improved “lookahead” method to estimate the path length of the full path to which each subpath is extended. MPLMR then selects and stores the subpaths that have higher likelihood than other subpaths to be extended to feasible paths. We show via simulation that MPLMR has a smaller probability of missing a feasible path than competing schemes in the literature.

© 2012 The Franklin Institute. Published by Elsevier Ltd. All rights reserved.

1. Introduction

QoS (quality of service) routing is an essential mechanism to support computer and communication applications that have requirements on various QoS attributes, such as
delay and jitter [1–7]. The multiconstraint QoS routing problem involves finding a feasible path, i.e., a path satisfying a given set of QoS constraints between given source and destination (terminal) nodes. Unfortunately, this routing problem is NP-complete [8], and thus we need heuristic schemes to solve this problem within a limited time. Many heuristic schemes have been proposed for the problem, but we limit our interest to only schemes with polynomial complexity. In addition, we limit ourselves to unicast routing, where a single pair of source and destination nodes is given.

Typical performance measures for multiconstraint QoS routing schemes are time complexity and erroneous decision rate (EDR). The EDR is defined as the fraction of instances that a routing scheme either fails to find a feasible path that exists, or finds a path that turns out to be infeasible. Low complexity and low EDR are the main goals of multiconstraint QoS routing schemes.

We can classify QoS attributes into min/max attributes and cumulative attributes (i.e., additive or multiplicative attributes in [8]). Min/max attributes, such as bandwidth, can be dealt with easily by pruning all links (and possibly their incident nodes) that do not satisfy the constraints on the attributes before starting to search for a feasible path [9]. Multiplicative QoS attributes can be regarded as additive by taking logarithms. Therefore, it suffices to consider only additive QoS attributes for the multiconstraint QoS routing problem.

Because of the additivity of QoS attributes considered, the value of a path with respect to a QoS attribute can be regarded as the “length” of the path with respect to the QoS attribute. It is easy to see that a single-constraint QoS routing problem has a feasible solution if and only if the shortest path between the source and destination nodes is feasible. To solve the multiconstraint QoS routing problem in the same way, several schemes take a similar approach by using extended versions of a standard shortest-path algorithm (e.g., Dijkstra’s algorithm), which we call extended shortest path algorithms in this paper. However, these schemes must use a modified definition of length, because the “length” in the multiconstraint QoS routing problem cannot be defined as in the single-constraint QoS routing problem. Unfortunately, for the modified definitions of length, the multiconstraint QoS routing problem has the following property: the shortest path between the source node and an arbitrary node u may not be a subpath of the shortest path between the source and destination nodes through u. Hence, to find the shortest path between the source and destination nodes, we may have to store all the subpaths between the source node and each intermediate node during the routing procedure, extend them to the destination node, and compare all the paths between the source and destination nodes at the end. Unfortunately, this path search has exponential complexity. To achieve polynomial complexity, we should limit the number of the subpaths to be stored for each node.

Before we describe our approach to solving the multiconstraint QoS routing problem, we introduce the following terms: prepath, postpath, and full path. For an arbitrary node u, we call any path p from source node s to u a prepath of u, and any path π from u to destination node t a postpath of u. We call u the endpoint node of p or π. In addition, we call any path between a given pair of source and destination nodes a full path, as illustrated in Fig. 1.

In this paper, we propose a multiconstraint QoS routing scheme, called MPLMR (multi-postpath-based lookahead multiconstraint routing). Like previous schemes (e.g., TAMCRA [10]), MPLMR uses an extended shortest-path algorithm with the notion of the nonlinear path length, which will be explained in the following section. MPLMR
also uses a “lookahead” method, exploited in previous schemes, e.g., H_MCOP\[11\]. That is, MPLMR considers postpaths associated with prepaths for the selection of a limited number of prepaths to be stored during the routing procedure. However, MPLMR uses a more effective lookahead method than H_MCOP. In contrast to H_MCOP, which precomputes a single postpath for each node, MPLMR precomputes multiple postpaths for each node. During the routing procedure, MPLMR uses these postpaths to estimate the nonlinear path length of the shortest full path to which each prepath is extended. Using this lookahead method, MPLMR selects prepaths with higher likelihood than other prepaths to be extended to the shortest full path. We show via simulation that MPLMR performs much better than TAMCRA and H_MCOP without sacrificing execution time.

The rest of this paper is organized as follows. In Section 2, we introduce our assumptions, state the multiconstraint QoS routing problem, and discuss the notions of nonlinear path length and lookahead methods. We summarize related work on multiconstraint QoS routing in Section 3. In Section 4, we discuss the approach of MPLMR to multiconstraint QoS routing. We describe the algorithm and complexity of MPLMR in Section 5. In Section 6, we use simulation to evaluate and compare MPLMR with competing schemes in the literature. We conclude in Section 7.

2. Multiconstraint QoS routing

2.1. Multiconstraint QoS routing problem

We assume that the following are given: a connected network topology; a set of QoS attribute values associated with each link; and a connection request specifying a source node, a destination node, and the constraints that the routing path must satisfy. We also assume that the network topology does not change throughout the routing procedure, that every QoS attribute is additive, and that QoS attribute values are nonnegative, known, and fixed.

We denote the link between arbitrary nodes $u$ and $v$ by $uv$. To represent a path between two arbitrary nodes, we list all the nodes on the path between ‘$<$’ and ‘$>$’. With this

\[1\]H_MCOP searches for the path that not only is feasible but also minimizes the value of a primary QoS attribute. However, if there is no primary QoS attribute designated, we can use H_MCOP just for finding a feasible path, as pointed out in [11–13]. Throughout this paper, we assume that H_MCOP does not have a primary QoS attribute designated.
notation and the above assumptions, we state the multiconstraint QoS routing problem as follows.

**Definition 1 (Multiconstraint QoS routing problem).** Suppose we are given a connected graph representing a network topology, \( G = (V, E) \), where \( V \) and \( E \) represent sets of \( n \) nodes and \( m \) links, respectively. Suppose also that each link \( uv \) is characterized by nonnegative values with respect to \( q \) additive QoS attributes, \( d_i(uv) \geq 0, i = 1, \ldots, q \). Given a source node \( s \), a destination node \( t \), and a constraint value \( C_i \) with respect to the \( i \)th QoS attribute for \( i = 1, \ldots, q \), find a path \( p = /s,w_1,\ldots,w_b,t/ \), where \( w_j, j = 1, \ldots, b \) is an intermediate node on path \( p \), such that the value of path \( p \) with respect to the \( i \)th QoS attribute, i.e., \( L_i(p) = d_i(s,w_1) + d_i(w_1,w_2) + \cdots + d_i(w_b,t) \), is less than or equal to the corresponding constraint value \( C_i \) for every \( i = 1, \ldots, q \).

### 2.2. Nonlinear path length

To solve the multiconstraint QoS routing problem, Neve and Mieghem [10] propose the notion of nonlinear path length, described as follows. Let \( C_i \) and \( L_i(p) \) for \( i = 1, \ldots, q \) be as defined in Definition 1. Also define the normalized length of path \( p \) with respect to the \( i \)th QoS attribute, denoted by \( \lambda_i(p) \), as follows:

\[
\lambda_i(p) = \frac{L_i(p)}{C_i}.
\]

The nonlinear path length of \( p \), denoted by \( \Lambda(p) \), is defined to be the maximum of the normalized lengths of \( p \) with respect to each QoS attribute, as follows:

\[
\Lambda(p) = \max[\lambda_1(p), \lambda_2(p), \ldots, \lambda_q(p)].
\]

It is straightforward to see that path \( p \) is feasible if and only if \( \Lambda(p) \leq 1 \). For the single-constraint QoS routing problem, this reduces to that \( p \) is feasible if and only if \( \lambda_1(p) \leq 1 \). Therefore, the nonlinear path length provides a basis for using a standard (single-constraint) shortest-path algorithm for the multiconstraint QoS routing problem. However, standard shortest-path algorithms rely on the property that the length of a path is the sum of quantities associated only with individual links on the path, a property that fails to hold for the nonlinear path length. For this reason, multiconstraint QoS routing schemes using the nonlinear path length as the measure for the “length” of a path entails a modification of the standard approach (the modification will be described in the following sections).

### 2.3. Eligibility test and lookahead method

An arbitrary prepath \( p \) of a node \( u \) is *dominated* [14] if there is another prepath of \( u \) that has no larger value than \( p \) with respect to every QoS attribute, and has strictly smaller values than \( p \) with respect to some QoS attributes. If \( p \) is dominated by prepath \( p' \), and if \( p' \) cannot be extended to a feasible path, then prepath \( p \) also cannot. Hence, a brute-force approach is to maintain a set for each node that contains all nondominated prepaths found during the routing procedure [15]. When the algorithm terminates, we get all

---

2The definition here is a special case of the definition in [10].
nondominated full paths, and thus we can check their feasibility. However, this brute-force algorithm has exponential complexity.

To achieve polynomial complexity, extended shortest-path algorithms for multi-constraint QoS routing limit the number of prepaths to be stored for each node. Ideally, these schemes select prepaths that have higher likelihood than other prepaths to be extended to the shortest full path (in terms of the modified “length”, the nonlinear path length). Eligibility tests and lookahead methods can be used for this purpose.

*Eligibility tests* check if each prepath has the possibility to be extended to a feasible path. We call a prepath *eligible* if it has any possibility to be extended to a feasible path (eligibility depends on the specific test being used). By eliminating ineligible prepaths from consideration, we can reduce the “search space” in which we search for a feasible path.

The idea of *lookahead* methods is that the consideration of postpaths associated with prepaths is helpful for selecting prepaths with higher likelihood than other prepaths to be extended to the shortest full path. Using lookahead methods, we estimate the nonlinear path length of the shortest full path to which each prepath is extended. Due to the NP-completeness of the multiconstraint QoS routing problem, we cannot consider all the postpaths associated with a given set of prepaths. Hence, we first select some specific subset of the postpaths for applying a lookahead method, as described later.

### 3. Related work

Neve and Mieghem propose a modification to the brute-force algorithm, called TAMCRA [10]. During the course of the routing procedure, TAMCRA stores for each node at most $k$ shortest (in terms of nonlinear path length) prepaths that have been found so far, hoping that these prepaths would have higher likelihood than other prepaths to be extended to the shortest full path. However, TAMCRA uses no lookahead method. Hence, if an arbitrary node $u$ finds a new prepath $p_n$ that has nonlinear path length smaller than its $k$th shortest prepath $p_k$, then $p_n$ replaces $p_k$, even though prepath $p_n$ may be connected to a much longer postpath than that of $p_k$. Neve and Mieghem also propose an alternative scheme, called SAMCRA [16]. SAMCRA is almost the same as TAMCRA, but SAMCRA does not limit the number of prepaths to be stored for each node to guarantee finding an existing feasible path. Yuan proposes a scheme called the limited path heuristic [17], which is similar to TAMCRA. However, for each node, the limited path heuristic stores the $k$ prepaths that are not necessarily the shortest. Yuan proves that low EDR can be achieved by maintaining $O(n^2 \log n)$ prepaths for each node, where $n$ is the number of nodes.

To overcome the drawback of TAMCRA mentioned above, Korkmaz and Krunz propose an enhanced scheme, called H_MCOP [11]. Different from TAMCRA, H_MCOP uses a lookahead method as follows. H_MCOP precomputes a single postpath for each node at the first step of the routing procedure, with the hope that this single postpath would be the subpath of a feasible path through the node. Then, H_MCOP uses the postpath to update the set of at most $k$ prepaths for each node, with the goal that combining these prepaths with the postpath results in near-minimum nonlinear path lengths. However, if the precomputed postpath is not the subpath of a feasible path, then the postpath may misguide the selection of prepaths. A* Prune [18], proposed by Liu and Ramakrishnan for the problem of finding multiple feasible paths, uses a lookahead method similar to the one in H_MCOP. As in SAMCRA, A* Prune does not limit the maximum number of prepaths to be stored for each node.
There are several other approaches for solving the multiconstraint QoS routing problem. Some multiconstraint QoS routing schemes partition the QoS attribute values into a finite number of intervals and apply dynamic programming techniques or distributed routing techniques [19,20]. Some others prioritize QoS attributes to search for the path that optimizes the value of the top-priority QoS attribute under constraints on other QoS attributes [9,21,22]. To reduce the NP-complete multiconstraint QoS routing problem to one that is solvable in polynomial time, some schemes approximate the given QoS attribute values [17,23] or network topologies via topology aggregation [24]. Similarly, the dependencies between QoS attributes are also exploited for the simplification of a given multiconstraint QoS routing problem [25,26]. There are also schemes that take a sequential path search approach [27], and schemes that precompute the solutions to the expected routing problems to reduce the path-search time [28].

In our study, we do not compare our scheme with all of the above schemes, limiting our comparison only to TAMCRA and H_MCOP. The schemes in [16,18,27] have exponential complexity. Korkmaz and Krunz [11] maintain that H_MCOP is significantly superior to the schemes in [17,19,20,23]. The number of additive QoS attributes to be considered in [9,21,22,25,26] is limited to only two or less. As topology aggregation is used in [24], the imprecision in estimating aggregated values of QoS attributes also accumulates, and this has a significant negative impact on QoS routing [29]. The precomputation scheme in [28] cannot solve routing problems if connection requests have QoS constraint values for which solutions are not prepared in advance.

4. Elements of our approach

4.1. Multiple postpaths

Our approach to multiconstraint QoS routing, called MPLMR (described in detail in Section 5), uses an extended shortest-path routing algorithm based on the notion of the nonlinear path length. Like previous schemes (e.g., TAMCRA and H_MCOP), MPLMR stores at most \( k \) prepaths for each node and updates them during the routing procedure, with the intent that these prepaths have higher likelihood than other prepaths to be extended to the shortest (in terms of nonlinear path length) full path. In addition, MPLMR incorporates a lookahead method by selecting and storing \( q \) postpaths for each node \( u \) at the beginning of the routing procedure (recall that \( q \) is the number of QoS attributes), as is shown in Fig. 2. Each of these postpaths is the shortest path between \( u \) and destination node \( t \) with respect to the corresponding QoS attribute, i.e., the \( i \)th postpath has the smallest value for the \( i \)th QoS attribute among all possible postpaths. MPLMR uses these postpaths for the eligibility test and the lookahead method, as described in the following sections.

![Fig. 2. Prepaths and postpaths for each node \( u \).](image-url)
4.2. Eligibility test of MPLMR

Let $p$ be a prepath of an intermediate node $u$, $\pi_j(u)$ the shortest postpath of $u$ with respect to the $j$th QoS attribute for $j = 1, \ldots, q$, and $p + \pi_j(u)$ the full path combining $p$ with $\pi_j(u)$. For each prepath $p$, MPLMR performs the following three steps: (1) MPLMR checks if $p$ is dominated by any other prepath stored for node $u$, as TAMCRA does. If $p$ is dominated by another prepath $p'$, MPLMR eliminates $p$ from consideration (because if $p'$ cannot be extended to a feasible path, then $p$ also cannot). (ii) If $p$ is not dominated, MPLMR checks if the combined path $p + \pi_j(u)$ is feasible for $j = 1, \ldots, q$. If any of these combined paths is feasible, then the routing procedure terminates—this combined path solves the routing problem. (iii) If none of the combined paths is feasible, MPLMR investigates the eligibility of $p$ by checking if $\lambda_j(p + \pi_j(u)) > 1$ for any $j = 1, \ldots, q$, following the method introduced in [30]. If $\lambda_j(p + \pi_j(u)) > 1$, then $p$ is declared ineligible because every full path extended from $p$ violates the $j$th QoS constraint (recall that $\pi_j(u)$ is the shortest postpath of $u$ with respect to the $j$th QoS attribute)—in this case, MPLMR eliminates $p$ from consideration. Otherwise, $p$ is declared eligible.

4.3. Lookahead method of MPLMR

Let node $u$, prepath $p$, and postpaths $\pi_j(u)$ be as given in Section 4.2. If the combined path $p + \pi_j(u)$ is infeasible for $j = 1, \ldots, q$, but if $p$ is still eligible, then MPLMR uses its lookahead method to estimate the nonlinear path length of the shortest full path extended from $p$. To explain the lookahead method, let $p_j(p)$ be the shortest (in terms of nonlinear path length) full path extended from $p$, and $\pi^*$ be the corresponding postpath of $u$ (i.e., $p_j(p) = p + \pi^*$). The basic idea in the lookahead method of MPLMR is to estimate $\lambda_j(p_j(p))$ (i.e., the normalized length of $p_j(p)$ with respect to the $i$th QoS attribute) as the weighted sum of $\lambda^*_i(p + \pi_j(u))$ (i.e., the normalized length of $p + \pi_j(u)$ with respect to the $i$th QoS attribute) for $j = 1, \ldots, q$. Let $w_{ij}$ be the weight of $\lambda_j(p + \pi_j(u))$ used to estimate the value of $\lambda_j(p_j(p))$. The estimated normalized length of $p_j(p)$ with respect to the $i$th QoS attribute, denoted by $\bar{\lambda}_i(p_j(p))$, is represented as follows:

$$\bar{\lambda}_i(p_j(p)) = \sum_{j=1}^{q} w_{ij} \lambda^*_i(p + \pi_j(u)), \quad (3)$$

where

$$\sum_{j=1}^{q} w_{ij} = 1. \quad (4)$$

As in Eq. (2), the estimated nonlinear path length of $p_j(p)$, denoted by $\bar{\lambda}(p_j(p))$, is defined as follows:

$$\bar{\lambda}(p_j(p)) = \max_{i=1,\ldots,q} \bar{\lambda}_i(p_j(p)). \quad (5)$$

MPLMR uses $\bar{\lambda}(p_j(p))$ as the basis to determine if $p$ should be stored for node $u$.

As shown in Eqs. (3) and (5), the estimated nonlinear path length of $p_j(p)$ is determined by the estimated normalized lengths of $p_j(p)$, which depend on the weight values $w_{ij}$.
for $i = 1, \ldots, q$ and $j = 1, \ldots, q$. Hence, the selection of appropriate weight values is important.

To select the weight values, we take into account the following two observations. First, $p_i(p)$ is likely to have a smaller normalized length with respect to every QoS attribute than most of the full paths extended from $p$. Based on this observation, our selection of the weights has the following property.

**Property 1.** For each $i$ and $j$, the smaller the value of $\lambda_i(p + \pi_j(u))$, the larger the value of $w_{ij}$ in Eq. (3).

The second observation is the following. Because $\pi_j(u)$ is the shortest postpath of node $u$ with respect to the $j$th QoS attribute, $\lambda_j(p_i(p)) \geq \lambda_j(p + \pi_j(u))$. Thus, as the value of $\lambda_j(p + \pi_j(u))$ becomes larger, it becomes more probable that $\lambda_j(p_i(p)) > 1$, which is a violation of the $j$th QoS constraint. Hence, if the value of $\lambda_j(p + \pi_j(u))$ is larger than the value of $\lambda_i(p + \pi_i(u))$ (recall that if $p$ is an eligible prepath, $\lambda_i(p + \pi_i(u)) \leq 1$ is nonnegative for $i = 1, \ldots, q$), then the $j$th QoS constraint is more stringent than the $i$th QoS constraint. In this case, the satisfaction of the $j$th QoS constraint must be given a higher priority than the $i$th QoS constraint if we are to find a feasible path by extending prepath $p$ toward the destination node. For satisfying the $j$th QoS constraint, it is advantageous to take $\pi_j(u)$ as the postpath to be connected to $p$. Hence, as the value of $\lambda_j(p + \pi_j(u))$ becomes larger, $\pi^*$ will likely be “closer” to $\pi_i(u)$ (i.e., $\pi^*$ will likely share more links with $\pi_i(u)$). Based on this observation, our weight selection has the following property.

**Property 2.** For each $j$, the larger the value of $\lambda_j(p + \pi_j(u))$, the larger the value of $w_{ij}$ for all $i$ in Eq. (3).

Let $r$ be a nonnegative real constant. Based on Properties 1 and 2, MPLMR uses the following weight $w_{ij}$ for Eq. (3):

$$w_{ij} = \frac{a_i}{[\lambda_j(p + \pi_j(u))][1 - \lambda_j(p + \pi_j(u))]} \tag{6}$$

where

$$a_i = \left(\sum_{j=1}^{q} \frac{1}{[\lambda_j(p + \pi_j(u))][1 - \lambda_j(p + \pi_j(u))]}\right)^{-1} \tag{7}$$

In Eqs. (6) and (7), $a_i$ is the value needed to satisfy the condition of Eq. (4) for $i = 1, \ldots, q$, and $r$ is a variable to control the relative contributions of $\lambda_i(p + \pi_j(u))$ and $1 - \lambda_j(p + \pi_j(u))$ to $w_{ij}$. (The effect of $r$ to the performance of MPLMR will be described in Section 5.2.)

If $\lambda_i(p + \pi_j(u)) = 0$ or $\lambda_i(p + \pi_j(u)) = 1$, we cannot compute the value of $\lambda_i(p_i(p))$ in Eq. (3) because the weight expression in Eq. (6) is not defined. To deal with this difficulty, MPLMR sets $\lambda_i(p_i(p)) = 0$ if $\lambda_i(p + \pi_j(u)) = 0$, and simply eliminates $p$ from consideration if $\lambda_i(p + \pi_j(u)) = 1$ and $u \neq t$ (recall that if $\lambda_i(p + \pi_j(u)) = 1$ and $u = t$, then the routing procedure terminates because $p + \pi_j(u)$ is a feasible path).

5. **MPLMR: multi-postpath-based lookahead multiconstraint routing**

5.1. **MPLMR algorithm**

The basic principle of MPLMR is to select and update at most $k$ prepaths for each node $u$ during the routing procedure, as in previous schemes (e.g., TAMCRA and H_MCOP).
However, in contrast to previous schemes, MPLMR selects the prepaths of \( u \) to minimize the estimated nonlinear path lengths of the full paths containing these prepaths (i.e., the values of Eq. (5)). To compute the estimated nonlinear path lengths of Eq. (5), MPLMR uses Eqs. (1), (3), (6), and (7). The pseudocode of MPLMR is shown in Fig. 3.

When a connection request occurs, MPLMR starts the routing procedure. Lines 01–06 represent the initialization procedure of the modified Dijkstra’s algorithm, described in lines 07–25. We can use any standard shortest-path algorithm to find the \( q \) postpaths for every node in line 02. In lines 04–05, if the normalized length of the \( i \)th postpath of source node \( s \) with respect to the \( i \)th QoS attribute exceeds one, the routing procedure stops because no full path can satisfy the \( i \)th QoS constraint. MPLMR maintains a set \( P(v) \) for each node \( v \), and another set \( Q \). \( P(v) \) contains at most \( k \) prepaths of \( v \), and \( Q \) contains prepaths (of any node) to be extended toward \( t \). In line 06, \( Q \) initially contains only a zero-length path from \( s \) to itself.

While \( Q \) is not empty, MPLMR extracts from \( Q \) a prepath, denoted by \( x \), such that the estimated nonlinear path length of the shortest full path including \( x \) is the smallest (lines 08–10). If the endpoint node of prepath \( x \) (denoted by \( w \)) has outgoing nodes, then MPLMR extends \( x \) to an outgoing node \( u \) that is not on \( x \) (the extended prepath is denoted by \( y \) in line 12). If \( y \) is not dominated by any other prepaths of \( u \) that are stored in \( P(u) \), MPLMR performs the following tasks: (i) MPLMR checks the feasibility of the full paths that consist of \( y \) and each of the postpaths of \( u \). If any of these full paths turns out to be feasible,
then MPLMR outputs the path as a solution to the multiconstraint QoS routing problem, and terminates the routing procedure (lines 14–16). (ii) If none of the full paths is feasible, but if \( y \) is eligible (i.e., \( \lambda_i(y + \pi_i(u)) \leq 1 \) for \( i = 1, \ldots, q \)), then MPLMR inserts \( y \) into \( Q \) and \( P(u) \). However, in this case, if the number of the prepaths that are stored for \( u \) in \( P(u) \) exceeds \( k \) (by one), then MPLMR removes a prepath from \( Q \) and \( P(u) \), such that the shortest full path extended from this prepath has a larger estimated nonlinear path length than the shortest full paths extended from any other prepaths of \( u \) in \( P(u) \). If there is no prepath in \( Q \), then the routing procedure terminates in line 25 with no feasible path found.

5.2. Control variable \( r \)

MPLMR uses the estimated nonlinear path length in Eq. (5) as the basis for selecting the prepaths to be stored for each node. Because MPLMR computes the estimated nonlinear path length using the weight values given in Eq. (6), the value of control variable \( r \) of the weight affects the selection of prepaths. In Eq. (6), we can see that the value of the weight becomes more dependent on Property 1 (Property 2) as the value of \( r \) increases (decreases). Property 1 makes \( \tilde{\lambda}_i(p_i(p)) \) close to \( \lambda_i(p + \pi_i(u)) \) for \( i = 1, \ldots, q \), which is the minimum value that \( \lambda_i(p_i(p)) \) could take from the viewpoint of the endpoint node of \( p \). Hence, the following proposition holds.

**Proposition 1.** Let \( p \) be a prepath of node \( u \), \( p_i(p) \) the shortest (in terms of nonlinear path length) full path extended from \( p \), and \( \pi_i(u) \) the shortest postpath of \( u \) with respect to the \( j \)th QoS attribute for \( j = 1, \ldots, q \). If \( \lambda_i(p + \pi_i(u)) \neq 0 \) and \( \lambda_i(p + \pi_i(u)) \neq 1 \) for \( i = 1, \ldots, q \) and \( j = 1, \ldots, q \), then, \( \lim_{r \to \infty} \tilde{\Lambda}(p_i(p)) \leq \Lambda(p_i(p)) \).

**Proof.** Because \( \pi_i(u) \) is the shortest postpath of \( u \) with respect to the \( i \)th QoS attribute, \( \lambda_i(p + \pi_i(u)) \leq \lambda_i(p + \pi_j(u)) \). If \( \lambda_i(p + \pi_i(u)) < \lambda_i(p + \pi_j(u)) \), then \( w_{ij} \to 0 \) as \( r \to \infty \) (see Eqs. (6) and (7)). Hence, if \( w_{ij} \to 0 \) as \( r \to \infty \), then \( \lambda_i(p + \pi_i(u)) = \lambda_i(p + \pi_j(u)) \). Thus, \( \tilde{\lambda}_i(p_i(p)) \to \lambda_i(p + \pi_i(u)) \) as \( r \to \infty \) for \( i = 1, \ldots, q \) by Eqs. (3) and (4). Because of the same reason (i.e., \( \pi_i(u) \) is the shortest postpath of \( u \) with respect to the \( i \)th QoS attribute), \( \lambda_i(p + \pi_i(u)) \leq \lambda_i(p_i(p)) \) for \( i = 1, \ldots, q \). Hence, \( \lim_{r \to \infty} \tilde{\lambda}_i(p_i(p)) \leq \lambda_i(p_i(p)) \) for \( i = 1, \ldots, q \), and therefore \( \lim_{r \to \infty} \tilde{\Lambda}(p_i(p)) \leq \Lambda(p_i(p)) \) by Eqs. (2) and (5). \( \Box \)

In contrast, Property 2 makes \( \tilde{\lambda}_i(p_i(p)) \) close to \( \lambda_i(p + \pi_j(u)) \) for \( i = 1, \ldots, q \) if the \( j \)th QoS constraint is the most stringent for finding a feasible path by extending prepath \( p \) toward the destination node. Thus, in this case, \( \tilde{\Lambda}(p_i(p)) \) becomes closer to \( \Lambda(p + \pi_j(u)) \) as the value of \( r \) decreases. Note that \( \Lambda(p + \pi_j(u)) \) must be larger than or equal to \( \Lambda(p_i(p)) \), and that \( \tilde{\Lambda}(p_i(p)) \) does not necessarily converge to \( \Lambda(p + \pi_j(u)) \) as \( r \to 0 \).

Consider the example shown in Fig. 4. Let the number of QoS attributes be two (i.e., \( q = 2 \)). As illustrated, for prepath \( p \) and postpaths \( \pi_j(u) \), \( j = 1, 2 \), of node \( u \), suppose \( \lambda_1(p + \pi_1(u)) = 0.5 \), \( \lambda_2(p + \pi_1(u)) = 1.1 \), \( \lambda_1(p + \pi_2(u)) = 2.0 \), and \( \lambda_2(p + \pi_2(u)) = 0.9 \). There may exist other postpaths, e.g., \( x \), such that \( \lambda_1(p + \pi_1(u)) \leq \lambda_1(p + x) \leq 1 \) and \( \lambda_2(p + \pi_2(u)) \leq \lambda_2(p + x) \leq 1 \). Because \( \lambda_1(p + \pi_1(u)) \leq 1 \) and \( \lambda_2(p + \pi_2(u)) \leq 1 \), \( p \) is eligible. As the value of \( r \) increases, Property 1 makes the values of \( \tilde{\lambda}_1(p_i(p)) \) and \( \tilde{\lambda}_2(p_i(p)) \) become closer to \( \lambda_1(p + \pi_1(u)) \) (i.e., 0.5) and \( \lambda_2(p + \pi_2(u)) \) (i.e., 0.9), respectively. However, \( \lambda_2(p + \pi_2(u)) = 0.9 > 0.5 = \lambda_1(p + \pi_1(u)) \), and thus the second QoS constraint is more stringent than the first QoS constraint. Hence, the effect of Property 2 due to the second QoS constraint is larger than the
The best value of \( r \) (e.g., to minimize the EDR of MPLMR) depends on the given routing problem. Our simulation in Section 6 shows that the value of \( r \) to minimize EDR is typically between 3 and 10. If the value of \( r \) is much larger than the typical values, Property 1 dominates Property 2. In this case, each estimated length may be larger than the actual length, and thus the EDR may also increase.

5.3. Comparison with competing schemes using an example

Fig. 5 shows an example to compare how TAMCRA, H_MCOP, and MPLMR work. Let the pair of values on each link represent two QoS attribute values associated with the link. Suppose that the constraint values with respect to the QoS attributes are given as \( C_1 = C_2 = 10 \). Suppose also that we are in the middle of the routing procedure to determine the prepaths of node \( u \) to be stored among three prepaths, \( p_1, p_2, \) and \( p_3 \). By an exhaustive search, we can see that path \( p_3 + p_4 = \langle s, v_3, u, v_4, t \rangle \) is the only feasible path.

When TAMCRA selects the prepaths to be stored for \( u \), the scheme considers the nonlinear path lengths of the prepaths only. The nonlinear path length of \( p_1 \) (i.e., \( A(p_1) \)) is 0.8 (i.e., \( \max[(0 + 2)/10, (3 + 5)/10] \)). Similarly, \( A(p_2) \) and \( A(p_3) \) are 0.8 and 0.9, respectively.
Because $A(p_3) > A(p_1) = A(p_2)$, TAMCRA selects $p_3$ to store for $u$ only if the maximum number of prepaths to be stored for each node (i.e., $k$) is at least three. Thus, TAMCRA finds a feasible path only if $k \geq 3$.

Recall that H_MCOP selects a single postpath for each node. Among all possible postpaths for each node, the selected postpath should have the minimum value of the sum of normalized lengths. Hence, H_MCOP selects path $p_5$ as this single postpath of node $u$, because the corresponding value for path $p_5$ (i.e., $\lambda_1(p_5) + \lambda_2(p_5) = 0.8$) is smaller than that for path $p_4$ (i.e., $\lambda_1(p_4) + \lambda_2(p_4) = 0.9$). When path $p_5$ is connected with the prepaths of $u$, $p_1 + p_5$, $p_2 + p_5$, and $p_3 + p_5$ have nonlinear path lengths of 1.2, 1.2, and 1.3, respectively. Because $A(p_3 + p_5) > A(p_1 + p_5) = A(p_2 + p_5)$, H_MCOP stores $p_3$ for $u$ only if $k \geq 3$. Thus, H_MCOP also finds a feasible path only if $k \geq 3$.

The paths $p_4$ and $p_5$ are the shortest ones between $u$ and $t$ for the first and the second QoS attributes, respectively. Hence, MPLMR selects paths $p_4$ and $p_5$ as the postpaths of node $u$ with respect to the first and the second QoS attributes, respectively (i.e., $\pi_1(u) = p_4$ and $\pi_2(u) = p_5$). Suppose that $r = 5$. Using Eq. (1), MPLMR computes the normalized lengths of all the full paths in Fig. 5, as shown in Table 1. Because $\lambda_2(p_1 + \pi_2(u)) = 1.2 > 1.0$, $p_1$ is ineligible. Hence, MPLMR eliminates $p_1$ from consideration. Using Eqs. (6) and (7) and the values in Table 1, MPLMR also computes the weight values in Table 2. From Eq. (3) and the values in Tables 1 and 2, $\lambda_1(p_3(p_2)) = w_{11}\lambda_1(p_2 + \pi_1(u)) + w_{12}\lambda_1(p_2 + \pi_2(u)) = 0.884$. Similarly, $\lambda_2(p_3(p_2)) = 0.926$, $\lambda_1(p_3(p_3)) = 0.910$, and $\lambda_2(p_3(p_3)) = 0.447$. Hence, the estimated nonlinear path lengths of the shortest full paths extended from $p_2$ and $p_3$ (i.e., $\lambda(p_3(p_2))$ and $\lambda(p_3(p_3))$) are 0.926 and 0.910, respectively. Because $\lambda(p_3(p_3)) < \lambda(p_3(p_2))$, MPLMR selects $p_3$ first, and thus finds a feasible path even for $k = 1$.

5.4. Complexity of MPLMR

Recall that $n$, $m$, and $q$ are the numbers of nodes, links, and QoS attributes, respectively, and that $k$ is the maximum number of prepaths to be stored for each node. Assume that we use a

<table>
<thead>
<tr>
<th>Table 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>The normalized lengths of all the full paths in Fig. 5.</td>
</tr>
<tr>
<td>$\lambda_1(p_1 + \pi_1(u)) = 0.2$</td>
</tr>
<tr>
<td>$\lambda_1(p_1 + \pi_2(u)) = 0.6$</td>
</tr>
<tr>
<td>$\lambda_1(p_2 + \pi_1(u)) = 0.8$</td>
</tr>
<tr>
<td>$\lambda_1(p_2 + \pi_2(u)) = 1.2$</td>
</tr>
<tr>
<td>$\lambda_1(p_3 + \pi_1(u)) = 0.9$</td>
</tr>
<tr>
<td>$\lambda_1(p_3 + \pi_2(u)) = 1.3$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>The values used to compute the estimated nonlinear path lengths of the shortest full paths extended from $p_2$ and $p_3$.</td>
</tr>
</tbody>
</table>

For $p_2$,

- $w_{11} = 0.792$, $w_{12} = 0.208$, $w_{21} = 0.052$, and $w_{22} = 0.948$
- $(a_1 = 0.0519$ and $a_2 = 0.0560)$

For $p_3$,

- $w_{11} = 0.974$, $w_{12} = 0.026$, $w_{21} = 0.094$, and $w_{22} = 0.906$
- $(a_1 = 0.0575$ and $a_2 = 0.0056)$
heap [31] for the data structure to store paths. Lines 01–02 in the pseudocode of MPLMR (Fig. 3) require \( q \) executions of a standard shortest-path algorithm to find the shortest postpaths of every node. If we use Dijkstra’s algorithm, then the run-time of lines 01–02 is \( O(mq + nq \log n) \) [31]. Because each node has at most \( k \) prepaths, the set \( Q \) contains at most \( kn \) prepaths. The computation of the estimated nonlinear path length in Eq. (5) takes \( O(nq^2) \) time for each prepath. Hence, the total computation time of the values in lines 04, 08, 15, 17, 19, 20, and 21 for \( kn \) prepaths is \( O(kn^2 q^2) \). Because we use a heap structure, the run-time for selecting a prepath among at most \( kn \) prepaths and for removing/inserting the prepath in lines 10, 18, 22, and 24 is \( O(kn \log(kn)) \) for the entire course of the MPLMR algorithm [31]. The for-loop between lines 11 and 24 should run at most \( k \) times to examine each link \( wu \) in the adjacency lists of \( w \) and \( u \), respectively. Hence, the total number of iterations of the for-loop is \( O(km) \).

Each of these iterations takes \( O(kq + n) \) time (without considering lines 18, 22, and 24) because of the tests for looping and dominancy on lines 11 and 13. Thus, the run time of the for-loop between lines 11 and 24 for the entire course of the MPLMR algorithm is \( O(km(kq + n)) \) without considering lines 18, 22, and 24. Therefore, by adding all these contributions, we obtain the time complexity for MPLMR of \( O(mq + nq \log n) + O(kn^2 q^2) + O(kn \log(kn)) + O(km(kq + n)) = O(nq \log n + kn \log(kn) + k^2 mq + kmn + kn^2 q^2) \). Note that if the maximum number of prepaths per node (i.e., \( k \)) is fixed, this time complexity is polynomial. TAMCRA and H_MCOP have the time complexities of \( O(kn \log(kn) + k^3 mq) \) [10] and \( O(n \log n + km \log(kn) + m(k^2 + 1)) \) [11], respectively. Hence, the time complexity of MPLMR is comparable to those of TAMCRA and H_MCOP.

MPLMR has to store at most \( kn \) prepaths and \( qn \) postpaths. Because each path has at most \( n \) nodes, MPLMR needs \( O(n^2 (k + q)) \) memory space. It is clear that the EDR of MPLMR decreases as the value of \( k \) increases. Hence, MPLMR achieves low EDR at the expense of the increased run time and the memory space for an increased number of prepaths.

6. Performance evaluation

6.1. Simulation setup

To evaluate the performance of MPLMR, we simulate the QoS routing problem according to the following steps. First, we generate a random network topology. Next, we randomly generate QoS attribute values to assign to every link in the generated network topology, such that the values have a given distribution with a given correlation coefficient between each pair of QoS attributes. We also assign a constraint value to each QoS attribute. Then, we apply MPLMR (for several values of \( r \)), TAMCRA, H_MCOP, and MPMP\(^3\) [32] to compare their performance.

To check if there exists any feasible path, we also apply an exhaustive-search scheme, which is MPLMR without any limitation on the number of prepaths to be stored per node (i.e., \( k = \infty \)). To compute the EDR, we ascertain whether or not there is an erroneous decision for each scheme. Note that an erroneous decision in this case corresponds to

\(^3\)MPMP is a predecessor of MPLMR, which does not have Property 2 of MPLMR (i.e., \( r = \infty \)). Like MPLMR, MPMP searches for the shortest postpath for each prepath with respect to each QoS attribute. However, MPMP supposes that a virtual postpath exists with the smallest path length with respect to each QoS attribute, and combines the prepath and the virtual postpath to estimate the nonlinear path length of the shortest full path extended from the prepath. Thus, by Proposition 1, an estimated nonlinear path length may be less than the corresponding actual nonlinear path length.
a failure to find a feasible path when one exists, because we assume that all the information on a given network topology and QoS attribute values is known and fixed. We perform 10,000 simulation runs of the above procedure for each combination of the following four items: (a) the number of QoS attributes (i.e., $q$)—two or three, (b) the distribution of link values—two distributions that will be described, (c) the correlation coefficient between each pair of QoS attributes—the five values $-0.8$, $-0.4$, $0$, $0.4$, and $0.8$, and (d) the maximum number of prepaths per node (i.e., $k$)—one or two.

We generate the random network topologies as follows. Source and destination nodes are located at diagonally opposite corners of a square area of unit dimension, and then 198 nodes are spread randomly in the square area. Using the Waxman model [33], we introduce a link between arbitrary nodes $u$ and $v$ with the following probability, which depends on the distance between them, $d(u, v)$:

$$
Pr(uv) = \alpha \exp\left(-\frac{\delta(u,v)}{\beta \sqrt{2}}\right).
$$

For the values of $\alpha$ and $\beta$ in the above equation, we use 0.8 and 0.06, respectively. The above approach results in 200 nodes and approximately 567 links per network topology. Hence, the average node degree is 5.67.

To assign each link QoS attribute values, we generate correlated random values using Randgen [34]. We simulate for two kinds of distributions. For the first distribution, the link values with respect to every QoS attribute are distributed uniformly in $[1,3]$, and thus the mean and standard deviation are 2 and 0.577, respectively. We set the correlation between QoS attributes by the method explained in Section 7.1 of [34]. The second distribution is a jointly normal distribution. The mean and standard deviation of the link values with respect to every QoS attribute are the same as in the first distribution (i.e., 2 and 0.577, respectively). Whenever we generate a negative value, we replace it by zero. Because very few values are replaced by zeros, QoS attribute values are still approximately normally distributed.

After assigning QoS attribute values, we assign the constraint values. We assign the same value to all the constraints to keep every constraint equally difficult.

### 6.2. Simulation results

Table 3 shows the numbers of erroneous decisions versus correlation coefficients for the case of two QoS attributes. We can obtain the EDR of each scheme for a given correlation coefficient by dividing the number of erroneous decisions by the number of simulation runs (i.e., 10,000). We can see that MPLMR has lower EDR than TAMCRA, H_MCOP, and MPMP for all the correlation coefficients. This observation applies to both the distributions in (a) and (b). Table 4 shows simulation results for the case of three QoS attributes: the first two are the same as in Table 3, and the third QoS attribute is uncorrelated with each of the first two. Again we observe that MPLMR has a lower EDR than the other schemes. We show results in Table 4 only for the first distribution (uniform), because the results for the second distribution (jointly normal) are very similar.

For the above simulation, the constraint value is 18 for every QoS attribute; we chose this value because the EDR is maximized around 18. Fig. 6 shows the impact of different constraint values for MPLMR with $k=1$. It is evident that the EDR decreases as the constraint value decreases: one reason for this is that the number of instances without
solutions increases. The EDR also decreases as the constraint value increases: one reason for this is that the constraints become increasingly loose.

In our simulation of 200 nodes with the first (uniform) distribution of two uncorrelated QoS attributes, the average execution times for the routing schemes are MPLMR \(k = 2, r = 5\) \(-0.295\) s, TAMCRA \(k = 2\) \(-1.36\) s, H_MCOP \(k = 2\) \(-0.682\) s, MPMP \(k = 2\) \(-0.316\) s, and the exhaustive scheme \(-58.2\) s (these execution times are only for a specific

Table 3
The number of erroneous decisions among 10,000 simulation runs for the routing problem with two QoS attributes, where MPLMR, TAMCRA, H_MCOP, and MPMP are applied to the randomly generated network topologies with QoS attribute values of (a) the first distribution (uniform) and (b) the second distribution (normal) described in Section 6.1. The constraint value is 18 for every QoS attribute.

<table>
<thead>
<tr>
<th>Correlation coefficient</th>
<th>-0.8</th>
<th>-0.4</th>
<th>0.0</th>
<th>0.4</th>
<th>0.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPLMR (k = 1, r = 5)</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TAMCRA (k = 1)</td>
<td>221</td>
<td>153</td>
<td>101</td>
<td>46</td>
<td>17</td>
</tr>
<tr>
<td>H_MCOP (k = 1)</td>
<td>127</td>
<td>74</td>
<td>43</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>MPMP (k = 1)</td>
<td>31</td>
<td>12</td>
<td>5</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>MPLMR (k = 2, r = 5)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TAMCRA (k = 2)</td>
<td>84</td>
<td>46</td>
<td>33</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>H_MCOP (k = 2)</td>
<td>88</td>
<td>48</td>
<td>25</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>MPMP (k = 2)</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPLMR (k = 1, r = 5)</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>TAMCRA (k = 1)</td>
<td>246</td>
<td>140</td>
<td>73</td>
<td>42</td>
<td>7</td>
</tr>
<tr>
<td>H_MCOP (k = 1)</td>
<td>128</td>
<td>69</td>
<td>40</td>
<td>22</td>
<td>8</td>
</tr>
<tr>
<td>MPMP (k = 1)</td>
<td>34</td>
<td>19</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>MPLMR (k = 2, r = 5)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TAMCRA (k = 2)</td>
<td>86</td>
<td>45</td>
<td>28</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>H_MCOP (k = 2)</td>
<td>95</td>
<td>44</td>
<td>26</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>MPMP (k = 2)</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4
The number of erroneous decisions among 10,000 simulation runs for the routing problem with three QoS attributes, where MPLMR, TAMCRA, H_MCOP, and MPMP are applied to the randomly generated network topologies with QoS attribute values of the first distribution (uniform) in Section 6.1. The constraint value is 18 for every QoS attribute. The first line shows the correlation coefficients between the first and second QoS attributes. The third QoS attribute has zero correlation with the first and the second QoS attributes.

<table>
<thead>
<tr>
<th>Correlation coefficient</th>
<th>-0.8</th>
<th>-0.4</th>
<th>0.0</th>
<th>0.4</th>
<th>0.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPLMR (k = 1, r = 5)</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>TAMCRA (k = 1)</td>
<td>241</td>
<td>183</td>
<td>151</td>
<td>142</td>
<td>104</td>
</tr>
<tr>
<td>H_MCOP (k = 1)</td>
<td>140</td>
<td>99</td>
<td>78</td>
<td>54</td>
<td>59</td>
</tr>
<tr>
<td>MPMP (k = 1)</td>
<td>26</td>
<td>8</td>
<td>15</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>MPLMR (k = 2, r = 5)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TAMCRA (k = 2)</td>
<td>85</td>
<td>53</td>
<td>39</td>
<td>39</td>
<td>31</td>
</tr>
<tr>
<td>H_MCOP (k = 2)</td>
<td>105</td>
<td>66</td>
<td>42</td>
<td>28</td>
<td>40</td>
</tr>
<tr>
<td>MPMP (k = 2)</td>
<td>7</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
machine, and can be different for different machines). Hence, MPLMR has a shorter execution time than the other schemes. (Recall that the asymptotic time complexity of MPLMR is comparable to those of TAMCRA, H_MCOP, and MPMP.) Two possible reasons are as follows. First, in contrast to TAMCRA and MPMP, MPLMR stores multiple postpaths (not just the values of postpaths). Hence, if the full path consisting of a prepath and a postpath is feasible, MPLMR terminates the routing without extending the prepath toward the destination node. Second, in contrast to H_MCOP, MPLMR performs an eligibility test to discard ineligible prepaths. Thus, MPLMR reduces the search space for a feasible path.

Fig. 7 shows plots of the EDR for MPLMR versus \( r \) (the control variable in Eq. (6)) for the routing problem with two QoS attributes. We can see that the value \( r \) to minimize the EDR of MPLMR is approximately between 3 and 10, which is why we used \( r = 5 \) in Tables 3 and 4.

7. Conclusions

As demand for QoS-guaranteed services (such as multimedia contents delivery services) keep growing, multiconstraint QoS routing becomes more important. To solve routing problems with constraints on two or more cumulative QoS attributes (e.g., delay, jitter, and packet loss), we developed MPLMR, a multiconstraint QoS routing scheme using an extended shortest-path algorithm. MPLMR involves efficient features of previous schemes: the notion of nonlinear path length and the dominancy test as in TAMCRA [10], the eligibility test as in the randomized algorithm of [30], and a lookahead feature as in H_MCOP [11]. MPLMR uses not only these existing features but also an improved lookahead method. Using this lookahead method, MPLMR estimates the nonlinear path
length of the shortest (in terms of nonlinear path length) full path to which each prepath is extended. Based on the estimated nonlinear path lengths, MPLMR selects and stores a limited number of prepaths that have higher likelihood than other prepaths to be extended to the shortest full path. The asymptotic worst-case complexity of MPLMR is comparable with those of TAMCRA and H_MCOP. However, we show via simulation that MPLMR achieves much lower EDR than the schemes. Furthermore, MPLMR achieves the low EDR with even smaller execution time than the competing schemes. Hence, MPLMR provides a promising solution for multiconstraint QoS routing, which will become an essential tool for high-quality communication/computer services in the near future.

In this paper, we have limited ourselves to unicast routing, where a single pair of source and destination nodes is given. Some routing problems involve multiple source–destination pairs, such as in multicast. However, the problem of jointly optimizing the routes of multiple source–destination pairs is a different problem and involves different techniques that go beyond the scope of this paper. Moreover, even in situations with multiple source–destination pairs, algorithms to optimize individual routes are still relevant, in the following sense. In many practical scenarios, it is impossible to jointly optimize the routes of multiple source–destination pairs, because of the well-known computational burden associated with the full-blown problem. In such scenarios, a reasonable practical compromise is to optimize individual routes independently, where the network constraints come from the residual resources available after the other unicast routes have been established. This is the approach in many practical networks today. In this approach, our unicast routing method is applicable.

**Acknowledgments**

This research was supported in part by the Purdue Center for Education and Research in Information Assurance and Security (CERIAS), by the Colorado State University.

![Graph](image-url)
George T. Abell Endowment, and by NSF under Grants 0905399-CNS, 0700559-ECCS, 0098089-ECS, and 0099137-ANI.

References


