Highly Responsive and Efficient QoS Routing Using Pre- and On-demand Computations along with a New Normal Measure

Yanxing Zheng[1], Turgay Korkmaz[2] and Wenhua Dou[1]

[1] School of Computer Science, National University of Defense Technology, ChangSha, HuNan, 410073, P.R. China. Email: yxzheng@nudt.edu.cn
[2] Department of Computer Science, University of Texas, San Antonio, USA. E-mail: korkmaz@cs.utsa.edu

Abstract. Multi-constrained path (MCP) selection is one of the great challenges that QoS routing (QoSR) faces. To address it in an efficient and highly responsive manner, we propose a new QoSR algorithm, namely NM_MCP. Using the Dijkstra's algorithm with respect to each link metric, NM_MCP pre-computes k primary paths, where k is the number of link weights. When a routing request arrives, it executes a modified version of the Dijkstra's algorithm using a newly proposed, normal-measure-based nonlinear cost function.

keywords: QoS routing, Pareto optimal, multi-objective optimization.

1 Introduction

One of the key issues in the next-generation networks is how to identify feasible paths that can satisfy the quality-of-service (QoS) requirements of different applications. This problem is commonly known as QoS routing (QoSR). The potential benefits of QoSR and the need for it have been recognized and acknowledged by the research community and industry [1][2]. Much work has been done on various aspects of QoSR [3][4][5]. For better responsiveness and higher success rate in identifying feasible paths, we developed a new normal measure based MCP algorithm (NM_MCP).

The rest of the paper is organized as follow. In Section 2, we formally define some related concepts. In Section 3, we review the most related studies. In Section 4, we present the proposed NM_MCP algorithm. We report simulation results in Section 5. Finally, we conclude the paper in Section 6.

2 Problem Formulation and Preliminaries

Let P_{sd} denote the set of paths between source node s and destination node d in network $G(N, E)$, where N is the set of nodes and E is the set of links. For path $p = n_1 \rightarrow n_2 \rightarrow, \ldots, \rightarrow n_m \in P_{sd}$, each link is associated with k

additive metrics. The sum of the j^{th} metric of path p can be represented as: $w_j(p) = \sum_{i=2}^{m} w_j(n_{i-1} \rightarrow n_i), j \in \{1, 2, \ldots, k\}$. Thus path p can be written as $p(w_1(p), w_2(p), \ldots, w_k(p))$.

Definition 1. *$W^k = W_1 \times W_2 \times \ldots \times W_k$ is called QoS Metric Space (QoSMS), where $w_j(p) \in W_j, j \in \{1, 2, \ldots, k\}$ for any $p \in G(N, E)$.*

Definition 2. *Mapping F is a function that maps path $p(w_1(p), w_2(p), \ldots, w_k(p))$ to a point in W^k, i.e., $F(p) = (f_1(p), f_2(p), \ldots, f_k(p)) = (w_1(p), w_2(p), \ldots, w_k(p))$.*

Definition 3. *Multi-constrained path (MCP) problem: Consider a network $G(N, E)$. Each link (u, v) is associated with a k-dimensional metric vector. Each element of w is an additive QoS metric: $w_i(u, v) \geq 0, i = 1, 2, \ldots, k$. Given routing request $c = (c_1, c_2, \ldots, c_k)$, the problem is to find a path $p \in P_{sd}$ such that $f_i(p) = w_i(p) = \sum_{(u,v)\in p} w_i(u, v) \leq c_i$, for $i = 1, 2, \ldots, k$.*

Definition 4. *Multi-constrained and Multi-optimization Problem (MCMOP): For the above MCP problem, in addition to finding a path subject to the given constraints, the objective is to*

$$\min F(p) = \min\{w_1(p), w_2(p), \ldots, w_k(p)\} \tag{1}$$

Clearly, the solutions to MCMOP will also be the solutions to MCP in Definition 3. Therefore, given a MCP problem, we can solve the derived MCMOP problem and take its solutions as that of original MCP problem. MCMOP is actually a special case of discrete MOOP that can simply be stated as follows.

Definition 5. *Multi-objective optimization problem (MOOP):*

$$\min_{x} F(x) = \min[f_1(x), f_2(x), \ldots, f_k(x)] \tag{2}$$

where k is the number of objectives.

Definition 6. *Cover: Vector $u = (u_1, u_2, \ldots, u_k)$ is said to cover $v = (v_1, v_2, \ldots, v_k)$, denoted by $u \preceq v$, iff $\forall i \in \{1, 2, \ldots, k\}, u_i \leq v_i$.*

Definition 7. *Dominance: Vector $u = (u_1, u_2, \ldots, u_k)$ is said to dominate $v = (v_1, v_2, \ldots, v_k)$, denoted by $u \prec v$, iff u is partially less than v, namely $\forall i \in \{1, 2, \ldots, k\}, u_i \leq v_i \wedge \exists i \in \{1, 2, \ldots, k\} : u_i < v_i$.*

Definition 8. *Pareto Optimal solution: Path $p(w_1(p), w_2(p), \ldots, w_k(p)) \in P_{sd}$ is a Pareto Optimal solution iff there is no path $p'(w'_1, w'_2, \ldots, w'_k) \in P_{sd}$ such that $(w'_1, w'_2, \ldots, w'_k) \prec (w_1, w_2, \ldots, w_k)$*

Definition 9. *Pareto set P^* and Pareto front PF^*: For a given MOOP, $P^* = \{p(w_1, w_2, \ldots, w_k,) \in P_{sd} | p(w_1, w_2, \ldots, w_k)$ is a Pareto optimal solution of MOOP $\}$, and $PF^* = \{(w_1, w_2, \ldots, w_k) | p(w_1, w_2, \ldots, w_k) \in P^*\}$*

Elements in PF^* are also called as Pareto optimal points. If a routing request is satisfied by path $p(w_1, w_2, \ldots, w_k)$, then the request can also be satisfied by an element in PF^*. So, when we deal with MCMOP problems, we can only consider the elements in PF^*. If any Pareto optimal point cannot satisfy the routing request, the request should be refused. This property allows us to efficiently reduce the search space without compromising solutions.

3 Related Work

In general, researchers used two kinds of path length functions: linear path length function (LPLF) and nonlinear path length function (NLPLF). LPLF first takes linear combination of multiple link weights and make a single link weight for each link as $l(u \rightarrow v) = \sum_{i=1}^{k} \alpha_i w_i(u \rightarrow v)$. Combination coefficient vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ is usually called as a *search direction* of Dijkstra's algorithm. Later we use Dijkstra(α) to denote the Dijkstra's algorithm that searches in direction α. Typical LPLF-based QoSR algorithms include Jaffe's algorithm [6], LARAC [7], MEFPA [8] and DWCBLA [9].

The main disadvantage of LPLF-based algorithms is that Pareto optimal points lying in the nonconvex part of Pareto front can never be found. To deal with such cases, researchers naturally considered NLPLF-based algorithms [10][11]. One main drawback of the existing QoSR algorithms using NLPLF is that they do not take the practical spread of Pareto front into consideration and thus introduce unnecessary computations in various cases [12].

In contrast to the above QoSR algorithms, we consider a new NLPLF, which is inspired by the work of Das and Dennis [13]. Das and Dennis developed a new method called *normal boundary intersection* (NBI) to obtain Pareto optimal points for an MOOP in case of *continuous* objective space. This method provides a means for obtaining an even distribution of Pareto optimal points based on the user-supplied parameters vector β, even with a nonconvex Pareto front. NBI method transforms an MOOP into the following sub-problem:

$$\text{Minimize} \quad \lambda \tag{3}$$

such that

$$\phi\beta + \lambda\hat{n} = F(x) - F^* \tag{4}$$

In this sub-problem, ϕ is a pay-off $k \times k$ matrix in which the i^{th} column is composed of the vector $F(x_i^*) - F^*$, where $F(x_i^*)$ is the vector of objective functions evaluated at the minimum of the i^{th} objective function; β is a vector of scalars such that $\sum_{i=1}^{k} \beta_i = 1$ and $\beta_i \geq 0; \hat{n} = \phi e$, where $e \in R^k$ is a column vector of ones in objective space. $\phi\beta$ is referred as the *Convex Hull of Individual Minima* (CHIM). The set of attainable objective vectors $\{F(x)\}$ is denoted by $\hbar$. The boundary of $\hbar$ is denoted by $\partial\hbar$. NBI is in fact a technique intended to find the portion of $\hbar$ which contains the Pareto optimal points.

For a specific β, $\phi\beta$ represents a point in the CHIM. $\phi\beta + t\hat{n}, t \in R$ represents the points on normal $\hat{n}$. The solution of (3) is the point of intersection of the normal and $\partial\hbar$ closest to the origin. The constraint in (4) ensures that the point x is actually mapped by F to a point on the normal. The sub-problem given in (3) is referred as the NBI sub-problem and written as NBI_β. Different solutions would be found by varying β.

4 Proposed Algorithm

The NBI method proposed for an MOOP in case of continuous objective space cannot directly be used for MCMOP because (as clearly stated by Das and Dennis [13]) NBI may fail if the objective space is discrete as in MCMOP. Therefore, we reconsider the ideas behind NBI in the context of MCMOP problems.

4.1 Definition of path length

For MCMOP problems, we modify the constraint in (4) such that the points in $\partial\hbar$ can still be measured by the normal when $\partial\hbar$ is not connected. We normalize the objective function as in [14] so that the scaling deficiencies will be avoided.

Let p^{i^*} denote the path that has the minimum w_i for all paths between s and d. We can then define a *Utopia point* as

$$F^* = [f_1(p^{1^*}), f_2(p^{2^*}), \ldots, f_k(p^{k^*})]^T = [f_1^*, f_2^*, \ldots, f_k^*]^T \tag{5}$$

Let us also define a normalizing matrix as:

$$L = [l_1, l_2, \ldots, l_k]^T = F^N - F^* \tag{6}$$

where $F^N \stackrel{def}{=} [f_1^N, f_2^N, \ldots, f_k^N]^T$ and $f_i^N = \max[f_i(p^{1^*}), f_i(p^{2^*}), \ldots, f_i(p^{k^*})]$.

We can now define the normalized $F(p)$ as follows

$$\bar{F}(p) = [\bar{f}_1(p), \bar{f}_2(p), \ldots, \bar{f}_k(p)],$$

where $\bar{f}_i(p) = (f_i(p) - f_i^*)/l_i$.

Based on the above definitions, we define the path length as follows

$$\text{len}(p) = -\min(\lambda_i) \tag{7}$$

$$s.t. \;\; \bar{\phi}\beta + N = \bar{F}(p) \tag{8}$$

where $N = (\lambda_1 n_1, \lambda_2 n_2, \ldots, \lambda_k n_k)^T$ and $\bar{F}(p) = [\bar{f}_1(p), \bar{f}_2(p), \ldots, \bar{f}_k(p)]^T$.

The meaning of $\hat{n} = (n_1, n_2, \ldots, n_k)^T$ is the same as the one in (4) except that it is now pointing away from the origin. By denoting $\hat{\phi}\beta$ by a vector $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_k)^T$, we can rewrite the constraint in (8) as follow.

$$\lambda_i n_i = \gamma_i - \bar{f}_i(p), \text{i=1,2,...,k} \tag{9}$$

The path length defined in (7) is called Normal measure length, or NM_length. Note that NM_length is nonlinear and the length of path p in this definition is not necessarily positive. (Due to page limitations, detailed discussions related to the intuitive meaning of λ_i and how different Pareto optimal points can be found by varying β had to be skipped here but can be found in [12].)

We now give some important properties of NM_length. Again due to page limitations, we omit their proofs but they can be found in [12].

Theorem 1. *If the length of a path p is larger than the length of routing constraint c (which is taken as a point in objective space), then at least one weight of the path p violates the corresponding constraint.*

Theorem 2. *Suppose two paths $p(w_1, w_2, \ldots, w_k)$ and $q(w'_1, w'_2, \ldots, w'_k)$ are given. If $F(p) \preceq F(q)$, then the length of path p is no longer than that of q.*

Theorem 3. *Using LPLF in a given search direction $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k)$, Dijkstra($\alpha$) generates a Pareto optimal solution*

Theorem 4. *Let $p(w_1, w_2, \ldots, w_k) \in P_{sd}$ denote the solution generated by Dijkstra(α). If routing request $c = c(c_1, c_2, \ldots, c_k)$ dominates $p(w_1, w_2, \ldots, w_k)$, then c cannot be satisfied by any paths in P_{sd}.*

4.2 NM_MCP algorithm

As outlined in Fig 1, the proposed algorithm NM_MCP mainly consists of two phases: precomputation and on-demand computation. In the precomputation

NM_MCP$(G, s, d,\ c)$
G: Network topology; s: source node;
d: destination node; c: routing request $(c_1, c_2, \ldots, c_k)$
Boundlen: The length of the routing request
(1): p^{i^*}=Pre-computation(G,s), $i = 1, 2, \ldots, k$
(2): on-demand computation
(2.1) Check_feasibility(p^{i^*}, c)
(2.2) if feasibility is not decided
(2.2.1) Preparation($p^{i^*}, c = (c_1, c_2, \ldots, c_k)$)
(2.2.2) NM_Dijkstra(G, s, d, c,*Boundlen*)

Fig. 1. Proposed NM_MCP algorithm.

phase, NM_MCP executes the procedure called Pre_computation given in Fig 2. This procedure simply computes the shortest paths individually with respect to each weight. We call those paths that minimize the individual weights as *primary paths*. Note that these paths depict the 'state' of the network from the viewpoint of the source node s and will be used in on-demand computation phase. Based on the primary paths, we also obtain $f_i^*, i = 1, 2, \ldots, k$, and compute normalizing vector L using Eq. (6).

In the on-demand computation phase, which is entered when a routing request c arrives, NM_MCP first calls Check_feasibility that is also given in Fig 2.

This procedure checks the feasibility of primary paths. If there is no primary paths that can satisfy c, NM_MCP tries to find a feasible path by executing a

(I)Pre_computation(G, s)
G: Network topology; s: Source node
(1): Computing k primary paths p^{i^*},for $i = 1, 2, \ldots, k$
(2): Determining Utopia point
$F^* = [f_1^*, f_2^*, \ldots, f_k^*]^T = [f_1(p^{1^*}), f_2(p^{2^*}), \ldots, f_k(p^{k^*})]^T$
(3):Computing normalizing vector $L = [l_1, l_2, \ldots, l_k]^T$

(II)Check_feasibility(p^{i^*}, c)
(1)For $i = 1, 2, \ldots, k$
(a) If $p^{i^*} \preceq c$ return p^{i^*}; success
(b) If $c_i < w_i(p^{i^*})$return fail;
(2)End of for
(3)Otherwise, return undecided

Fig. 2. Procedure Pre_computation and Check_feasibility.

modified version of Dijkstra's algorithm (called NM_Dijkstra) with respect to the new link weight NM_length. Recall that different values for β will lead to different shortest paths with respect to NM_lengh. So, one key issue here is, for a specific routing request c, how to select a proper β that can generate a better path. Actually, we need to determine $\bar{\phi}\beta$ rather than the specific values of $\bar{\phi}$ and β. For this, we take the intersection of the normal which is across the routing request $c = (c_1, c_2, \ldots, c_k)$ and the Utopia hyperplane.

To compute $\bar{\phi}\beta$ and other required parameters (e.g., the length *Boundlen* of the routing request which will be used as the upper bound for paths explored in NM_Dijkstra), NM_MCP calls Procedure Preparation given in Fig 3. It then ex-

Preparation$(p^{i^*}, c = (c_1, c_2, \ldots, c_k))$
(1): Calculating $\bar{\phi\beta}$
(1.1)Determine the normal$\hat{n} = (n_1, n_2, \ldots, n_k)$ that across the routing request
(1.2)Get the intersection point of $\hat{n}$ and the hyperplane U.
This point is taken as $\bar{\phi}\beta$, and denoted by vector $(\gamma_1, \gamma_2, \ldots, \gamma_k)$
(2): Computing the length of routing request
(2.1)Normalizing the routing request: $\bar{c}_i = \frac{c_i - f_i^*}{l_i}$
(2.2)$\lambda_i = \frac{(\gamma_i - \bar{c}_i)}{n_i}$
(2.3)Boundlen=-min(λ_i)

Fig. 3. Preparation procedure.

ecutes NM_Dijkstra. Similar to the standard Dijkstra's algorithm, NM_Dijkstra maintains only three labels: cost(u), parent(u), and len(u) for each node u. cost(u) is a k-dimensional vector, each entry of which represents the individually accumulated link weights along the current path from s to u. parent(u) represents the predecessor of node u. len(u) is the length of the path from s to u. NM_Dijkstra initially sets len(u)=∞ and parent(u)=NIL for every node u. It then starts from node s, setting each entry of cost(s) to 0 and len(s) to $-\infty$. It then updates the labels of each adjacent node of s using the relaxation process shown in Fig 4.

NM_Dijkstra_Relax(u,v)
(1) Increase=$(w_1(u,v), w_2(u,v), \ldots, w_k(u,v))$
(2) Tempcost=cost(u)+Increase
(3) For $i = 1, 2, \ldots, k$
    newcost(i)=$\frac{\text{Tempcost}_{(i)} - f_i^*}{l_i}$
(4) For $i = 1, 2, \ldots, k$
    $\lambda_i = \frac{(\gamma_i - \text{newcost}_{(i)})}{n_i}$
(5) templen=-min(λ_i)
(6) If templen < len(v)
    len(v)=templen; parent(v)=u; cost(v)=Tempcost
(7) EndIf

Fig. 4. Relaxation procedure of NM_Dijkstra.

This relaxation procedure is the fist modification to the Dijkstra's algorithm. NM_Dijkstra then continues to explore the graph by choosing the next node that has the least length. The second modification here is to consider only those nodes whose lengths are smaller than the Boundlen when deciding whether to extend a node or not.

The computational complexity of NM_MCP algorithm is (k+1) times that of the Dijkstra's algorithm, k times for precomputation and (if needed) one time for on-demand computation. Therefore, the worst-case response time of NM_MCP is at most one iteration of the Dijkstra's algorithm while the best of existing on-demand algorithms require at least a few iterations of it.

4.3 Improvements to NM_MCP

Since NM_MCP adopts nonlinear path length, it also suffers from the inherent drawback of nonlinear path lengths, namely the sub-section of a shortest path is not necessary the shortest one. To further improve the performance, we enhanced NM_MCP by using a look-ahead method similar to the one used in [15]. In the

look-ahead method, we first compute the shortest path tree rooted at the destination to each node v in the graph. For this, Dijkstra's shortest path algorithm is executed k times for each link weight separately. Accordingly, the least length between the destination node and every node u is determined and maintained with respect to each link weight individually. We then use this information in the relaxation process of NM_Dijkstra algorithm to have a better estimation of the path length based on NM_length. The computation complexity of NM_MCP with look-ahead ability is ($2k$+1) times that of Dijkstra's algorithm. To distinguish from Pareto look-ahead, we call this look-ahead as nonlinear look-ahead.

5 Performance Evaluation

Performance evaluation includes three parts. The first part is the comparison of NM_MCP with H_MCOP [16][11], which is an efficient on-demand QoSR algorithm using nonlinear path length, and MEFPA [8], which is an efficient pre-computation algorithm using linear path length. The second part evaluates the performance increase introduced by Pareto look-ahead. The last part evaluates the response time of NM_MCP.

5.1 Simulation model and performance measures

Topologies used for simulations are based on Waxman's model with 50, 100 and 200 nodes. Each link is associated with k weights using $w_i \sim$ uniform[1,300]. For each node number, 200 graphs are generated per experiment. For each instance of a random graph, one source node and 25 destination nodes, which are at least two hops away from the source node, are generated randomly. For each source-destination pair, one routing request is generated.

As the key performance measure, we use success rate (SR), the ratio of the routing requests satisfied by heuristic to the total routing requests generated.

5.2 Performance comparison of QoSR algorithms

In this part, we compare the SR of NM_MCP with that of two QoSR algorithms: H_MCOP and MEFPA. The computational complexity of H_MCOP for every routing request is twice that of the Dijkstra's algorithm. MEFPA, on the other hand, requires pre-iterations of the Dijkstra's algorithm, where b is a user specified parameter that controls the complexity and the performance. MEFPA with parameter b is denoted by MEFPA(b). We use NM_MCP2 to denote NM_MCP with nonlinear look-ahead ability.

Routing requests are generated with respect to the k primary paths denoted by $p_i(w_{i1}, w_{i2}, \ldots, w_{ik}), i = 1, 2, \ldots, k$. Let $f_j^{max} = \max(w_{ij})$ and $f_j^{min} = \min(w_{ij}), i = 1, 2, \ldots, k$.

We first consider the same method in [16] to generate routing requests. Specifically, we generate:

$$c_i \sim \text{uniform}[0.8 * f_i^{max},\ 1.2 * f_i^{max}] \tag{10}$$

The case of generating constraints according to (10) in bi-objective QoSMS is shown in Fig 5.(a). As seen from the figure, most routing requests generated

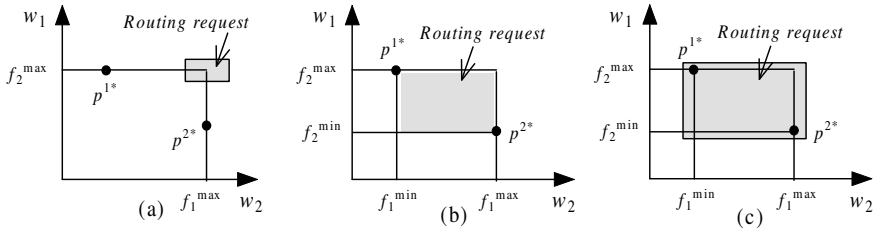

Fig. 5. The distribution of critical routing requests.

in this manner are likely to be feasible. We call such routing requests as *loose routing requests* (LRR).

In fact, the most critical routing requests lie in the area shown in Fig 5.(b) [9]. This critical area is the so called "NP-complete" range [17]. Loose routing requests only cover a small part of this area. So we use another method to generate routing requests:

$$c_i \sim \text{uniform}[0.8 * f_i^{min},\ 1.2 * f_i^{max}] \tag{11}$$

Fig 5.(c) shows the case of generating constraints according to (11) in bi-objective QoSMS. The whole NP-complete range is covered by the distribution of routing requests. We call such routing requests as critical routing requests (CRR).

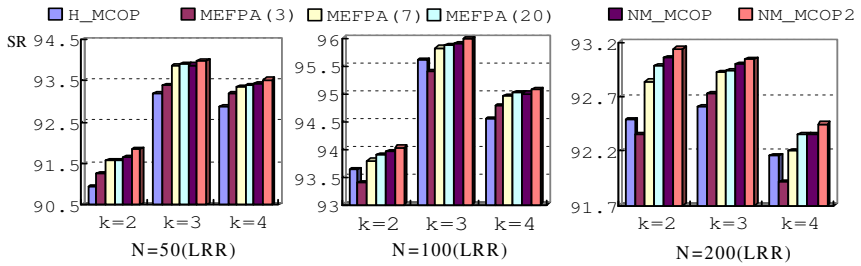

Fig. 6. Performance comparison of different algorithms under loose routing requests.

Fig 6 show the performance comparison of various algorithms under multiple constraints using LRR. As clearly seen from the figures, NM_MCP gives high SR and MEFPA becomes comparable to it when b is increased. When b takes a larger

number, MEFPA(b) will introduce not only a high computation overhead but also a larger routing table (after each iteration of Dijkstra's algorithm, a path will be stored in routing table) [9]. On the other hand, NM_MCP only performs k iterations of Dijkstra's algorithm and give comparable SR to that of MEFPA(20). This implies that NM_MCP can re-execute the Pre_computation procedure more frequently than MEFPA within the same period. As a result, NM_MCP will suffer less from stale routing information and give better performance in practice.

Due to their ability to cope with stale routing information, on-demand computation is also essential for the success of QoSR solutions. But H_MCOP requires two iterations of Dijkstra's algorithms. However, NM_MCP executes Dijkstra's algorithm at most once and give better performace with the help of the precomputed paths. Hence, NM_MCP has a quicker response speed than H_MCOP. So NM_MCP makes a good tradeoff between precomputation and on-demand computation. Furthermore, NM_MCP uses LPLF in precomputation phase and NLPLF in on-demand computation phase, making a good tradeoff between LPLF and NLPLF as well.

5.3 Evaluation of Pareto look-ahead

In this part, we evaluate the performance increase brought by Pareto look-ahead mechanism. We use the unfeasible rate (UR), the ratio of the times that on-demand computation is avoided to the number of total routing requests generated. Simulation results for UR versus k (which denote the number of link wights) are shown in Fig 7. We can see that performance gains are less as k

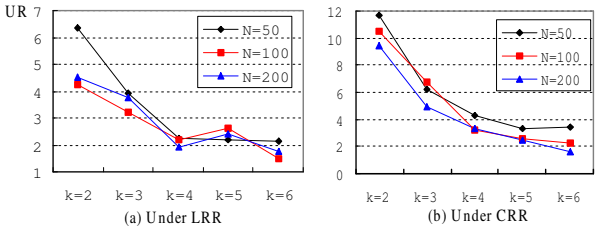

Fig. 7. Evaluation of Performance increase brought by Pareto look-ahead.

increases. This is because when k is large, the probability that routing requests dominate primary paths is small. Moreover, when routing requests are critical, considerable parts of them are inherently unfeasible. So the probability that these routing requests dominate primary paths is large and hence more performance increase is gained.

5.4 Evaluation of response time

A practical QoSR algorithm should have not only a high SR, but also a quick speed of response. In this part, we look at how often routing requests can be responded immediately by NM_MCP. Routing requests that can be responded immediately includes those satisfied by primary paths and those determined to be unfeasible by Pareto look-ahead. Hence we count individually the precomputation success rate (PSR), the ratio of the number of routing requests satisfied by primary paths to the number of routing requests generated, and UR. Routing requests that cannot be responded immediately need further on-demand computation. We count the ratio of the number of these routing requests to the number of total routing requests as on-demand computation rate (OCR).

Fig 8 shows the simulation results when routing request are loose and critical

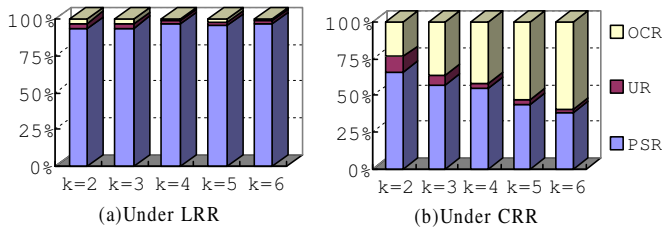

Fig. 8. Evaluation of response time and on-demand computation overhead.

with node number N=200. We can see that when routing requests are loose, most routing requests can be responded immediately and only very small parts of routing requests need further on-demand computation. When routing requests are critical, again a considerable part of routing requests does not require further on-demand computation. Note that only one iteration of Dijkstra's algorithm is executed for each routing request that needs on-demand computation. On average, on-demand computation overhead of NM_MCP is significantly lower than one iteration of Dijkstra's algorithm, particularly under LRR.

6 Conclusions

To solve a given MCP problem, we derived from it a new MCMOP problem, which can be viewed as a special case of discrete version of MOOP. We considered this problem to develop a new nonlinear path length function, namely NM_length. Based on this length function, we designed an efficient algorithm (namely, NM_MCP). Using extensive simulations, we showed that NM_MCP algorithm is very efficient when both SR and response time are taken into account.

References

1. Korkmaz, T., Krunz, M.: Bandwidth-delay constrained path selection under inaccurate state information. IEEE/ACM Transactions on Networking (ToN) **11** (2003) 384–398
2. Korkmaz, T.: State-path decoupled QoS-based routing framework. In: Proceedings of the IEEE GLOBECOM '04 Conference. (2004) (to appear).
3. Chen, S., Nahrstedt, K.: An overview of quality-of-service routing for the next generation high-speed networks: Problems and solutions. IEEE Network **12** (1998) 64–79
4. Zheng, Y., Dou, W., Tian, J., Xiao, M.: An overview of research on qos routing. In: Advanced Parallel Processing Technologies(APPT03), Springer LNCS(2834) (2003) 387–397
5. Kuipers, F., Korkmaz, T., Krunz, M., Van Mieghem, P.: Performance evaluation of constraint-based path selection algorithms. IEEE Network (2004) (to appear).
6. Jaffe, J.M.: Algorithms for finding paths with multiple constraints. Networks **14** (1984) 95–116
7. Juttner, A., Szviatovszki, B., Mecs, I., Rajko, Z.: Lagrange relaxation based method for the QoS routing problem. In: Proceedings of the INFOCOM 2001 Conference. Volume 2., IEEE (2001) 859–868
8. Cui, Y., Xu, K., Wu, J.: Precomputation for multi-constrained qos routing in high-speed networks. In: Proceedings of the INFOCOM 2003 Conference, San Francisco (2003)
9. Zheng, Y., Tian, J., Liu, Z., Dou, W.: An efficient dynamic weight coefficient qos routing algorithm. In: International Network Conference(INC04), UK (2004)
10. De Neve, H., Van Mieghem, P.: TAMCRA: a tunable accuracy multiple constraints routing algorithm. Computer Communications **23** (2000) 667–679
11. Korkmaz, T., Krunz, M.: Routing multimedia traffic with QoS guarantees. IEEE Transactions on Multimedia **5** (2003) 429–443
12. Zheng, Y., Korkmaz, T., Zhang, H., Dou, W.: Pre- and on-demand computations along with a new normal measure. Technical report (2004) http://www.cs.utsa.edu/∼korkmaz/yanxing/TR-NM-MCP.pdf.
13. Das, I., Dennis, J.: Normal-boundary intersection: a new method for generating the pareto surface in nonlinear multicriteria optimization problems. SIAM J. Optim **8** (1998) 631–657
14. Messac, A., Ismail-Yahaya, A., Mattson, C.: The normalized normal constraint method for generating the pareto frontier. Struct. Multidisc. Optim **25** (2003) 86–98
15. Korkmaz, T., Krunz, M.: A randomized algorithm for finding a path subject to multiple QoS constraints. Computer Networks Journal **36** (2001) 251–268
16. Korkmaz, T., Krunz, M.: Multi-constrained optimal path selection. In: Proceedings of the INFOCOM 2001 Conference. Volume 2., Anchorage, Alaska, IEEE (2001) 834–843
17. Kuipers, F., Mieghem, P.: The impact of correlated link weights on qos routing. In: Proceedings of the INFOCOM 2003 Conference, IEEE (2003)