Multi-Constrained Optimal Path Selection

Turgay Korkmaz Marwan Krunz
Department of Electrical & Computer Engineering
University of Arizona
Tucson, AZ 85721
{turgay,krunz}@ece.arizona.edu

Abstract— **Providing quality-of-service (QoS) guarantees in packet networks gives rise to several challenging issues. One of them is how to determine a *feasible* path that satisfies a set of constraints while maintaining high utilization of network resources. The latter objective implies the need to impose an additional optimality requirement on the feasibility problem. This can be done through a *primary* cost function (e.g., administrative weight, hop-count) according to which the selected feasible path is optimal. In general, multi-constrained path selection, with or without optimization, is an NP-complete problem that cannot be exactly solved in polynomial time. Heuristics and approximation algorithms with polynomial- and pseudo-polynomial-time complexities are often used to deal with this problem. However, existing solutions suffer either from excessive computational complexities that cannot be used for online network operation or from low performance. Moreover, they only deal with special cases of the problem (e.g., two constraints without optimization, one constraint with optimization, etc.). In this paper, we propose an efficient heuristic algorithm for the most general form of this problem. We first show that multiple constraints can be dealt with using a *nonlinear* cost function whose minimization provides a continuous spectrum of solutions ranging from a generalized linear approximation (GLA) to an asymptotically exact solution. We then introduce our heuristic algorithm (H_MCOP), which attempts to minimize both the nonlinear cost function (for the feasibility part) and the primary cost function (for the optimality part). We prove that H_MCOP guarantees at least the performance of GLA and often improves upon it. H_MCOP has the same order of complexity as Dijkstra's algorithm. Using extensive simulations on random graphs with correlated and uncorrelated link weights, we show that under the same level of computational complexity, H_MCOP outperforms its (less general) contenders in its success rate in finding feasible paths and in the cost of such paths.**

Keywords— **Multiple constraints, path selection, QoS routing, scalable routing, k-shortest paths.**

I. INTRODUCTION

THE current Internet has been designed to support connectivity based routing of best-effort traffic. However, the continuous growth in both commercial and public network traffic with quality-of-service (QoS) requirements is calling for new, QoS-oriented services (e.g., Diffserv, Intserv, MPLS, ATM). One of the key issues in the design of such services is how to identify a feasible route that satisfies multiple constraints (e.g., bandwidth, delay, jitter) while simultaneously achieving efficient utilization of network resources. This problem is known as QoS (or constraint-based) routing [1], [2], [3], [4], [5], [6]. Its relevance is justified for both reservation-based services (e.g., Intserv, MPLS, ATM) as well as reservationless services (e.g., Diffserv). For example, in the ATM PNNI protocol [7], constraint-based routing is performed by source nodes to determine suitable paths for connection requests. Similarly, in MPLS, which is a convergence of several efforts aimed at combining the best features of IP and ATM [8], a source router selects a path subject to QoS requirements and uses a signaling protocol (e.g., RSVP or CR-LDP) to reserve resources along that path. For QoS routing, routers need information about available network resources [8]. The mechanism to acquire such information can be part of the path establishment protocol, as in the case of PNNI, or it may be provided via existing routing protocols, as in the case of MPLS. In particular, MPLS relies on protocols such as OSPF (Open Shortest Path First) [9] to provide the link state information (e.g., delay, bandwidth, etc.). This requires slightly extending OSPF, as described in [10]. The type-of-service (TOS) field in OSPF, which has not been much used in the past, has now been redefined to advertise multiple link parameters (see [10] for details). In the case of Diffserv, the constraint-based routes can be requested, for example, by network administrators for traffic engineering purposes. Provisioning of such routes can also be used to guarantee a certain service level agreement (SLA) for aggregated flows [6].

In general, routing consists of two basic tasks: distributing the state information of the network and searching this information for a feasible, possibly optimal path. In this paper, we focus on the second task and assume that the true state of the network is available to every node (e.g., via link-state routing protocols such as OSPF) and that nodes use this information to determine end-to-end paths (see [11] for QoS routing under inaccurate information). Each link in the network is associated with multiple parameters which can be roughly classified into additive and non-additive [12], [13]. For the additive parameters (e.g., delay, jitter, administrative weight), the cost of an end-to-end path is given by the *sum* of the individual link values along that path[1]. In contrast, the cost of a path with respect to (w.r.t.) a non-additive parameter, such as bandwidth, is determined by the value of that constraint at the bottleneck link. It is known that constraints associated with non-additive parameters can be easily dealt with a preprocessing step by pruning all links that do not satisfy these constraints [14]. Hence, in this paper we will mainly focus on additive QoS parameters and assume that the optimality of a path is evaluated based on an additive parameter (e.g., administrative weight, hop-count). The underlying problem can be stated as follows.

Definition 1 *Multi-Constrained Optimal Path (MCOP) Problem*: Consider a network that is represented by a directed graph $G = (V, E)$, where V is the set of nodes and E is the set of links. Each link $(i, j) \in E$ is associated with a primary cost parameter $c(i, j)$ and K additive QoS parameters $w_k(i, j)$, $k = 1, 2, \ldots, K$; all parameters are non-negative. Given K constraints c_k, $k = 1, 2, \ldots, K$, the problem is to find a path p from a source node s to a destination node t such that:

(i) $w_k(p) \stackrel{\text{def}}{=} \sum_{(i,j)\in p} w_k(i, j) \leq c_k$ for $k = 1, 2, \ldots, K$, and

This work was supported by the National Science Foundation under grant ANI 9733143.

[1] Multiplicative constraints, such as link reliability, can be transformed into additive constraints.

(ii) $c(p) \stackrel{\text{def}}{=} \sum_{(i,j)\in p} c(i,j)$ is minimized over all feasible paths satisfying (i).

For $K = 1$ the MCOP problem is known as the restricted shortest path (RSP) problem, which is NP-complete [15]. A slightly different version of the MCOP problem is known as the multi-constrained path (MCP) problem, which aims only at finding any feasible path w.r.t. multiple constraints (no path optimization is done). For $K \geq 2$ the MCP problem is also known to be NP-complete [16], [17]. Both the RSP and MCP problems can be solved via pseudo-polynomial-time algorithms whose complexities depend on the actual values of the link weights (e.g., maximum link weight) in addition to the size of the network [18], [17]. However, these algorithms are computationally expensive if the values of the link weights are large. To cope with the NP-completeness of these problems, researchers have resorted to several heuristics and approximation algorithms.

In [18] the author presented two ϵ-optimal approximation algorithms for RSP with complexities of $\mathcal{O}(\log\log B(m(n/\epsilon) + \log\log B))$ and $\mathcal{O}(m(n^2/\epsilon)\log(n/\epsilon))$, where B is an upper bound on the solution (e.g., the longest path), m is the number of links, and ϵ is a quantity that reflects how far the solution is from the optimal one. These complexities have been slightly improved in [19], [20]. For $\epsilon \sim \mathcal{O}(1)$, these complexities, albeit polynomial, are still impractical for real-time network operation in large networks [21]. Accordingly, the author in [21] introduced a new ϵ-optimal approximation algorithm with better scalability. In [22] the authors proposed the ϵ-optimal approximation algorithm for a RSP-related problem, in which one link weight is a function of the other. In [23] the author proposed the Constrained Bellman-Ford (CBF) algorithm, which performs a breadth-first-search by discovering paths of monotonically increasing delay while maintaining lowest-cost paths to each visited node. Although this algorithm exactly solves the RSP problem, its running time grows exponentially in the worst case. Another approach to the RSP problem is to find the k-shortest paths w.r.t. a cost function defined based on the combination of link weights and the given constraint, hoping that one of these paths is feasible and near-optimal [24], [25], [26], [27]. The value of k determines the performance and overhead of this approach; if k is large, the algorithm has good performance but its computational cost is expensive. A similar approach to the k-shortest paths is to *implicitly* enumerate all feasible paths [28], but this approach is also computationally expensive. The authors in [29] proposed a distributed heuristic solution for RSP with message complexity of $\mathcal{O}(n^3)$, where n is the number of nodes. This complexity was improved in [30], [31]. The above algorithms are especially proposed for the RSP problem (i.e., they do not consider multiple constraints) and their computational complexities are often excessive in the worst case.

In [17] the author considered the MCP problem under two constraints and proposed an intuitive approximation algorithm to it based on minimizing a linear combination of the link weights. More specifically, his algorithm returns the best path w.r.t. $l(e) \stackrel{\text{def}}{=} \alpha w_1(p) + \beta w_2(p)$ by using Dijkstra's shortest path algorithm, where $\alpha, \beta \in Z^+$. The key issue here is to determine the appropriate α and β such that an optimal path w.r.t. $l(e)$ is likely to satisfy the individual constraints. Two sets of values for α and β were determined in [17] based on minimizing an objective function of the form $f(p) = \max\{w_1(p), c_1\} + \max\{w_2(p), c_2\}$. Instead of fixing α and β a priori, the authors in [32] proposed a similar approximation algorithm that dynamically adjusts the values of α and β. However, the computational complexity of this algorithm grows exponentially with the size of the network. Another heuristic for the MCP problem under two constraints was proposed in [33]. In this study, the original problem was modified by scaling down the values of one of the two link weights to bounded integers. It was shown that the modified problem can be solved by using Dijkstra's (or Bellman-Ford) shortest path algorithm and that the solution to the modified problem is also a solution to the original one. When Dijkstra's algorithm is used, the computational complexity of the algorithm is $\mathcal{O}(x^2n^2)$; when Bellman-Ford algorithm is used, the complexity is $\mathcal{O}(xnm)$, where x is an adjustable positive integer whose value determines the performance and overhead of the algorithm. To achieve a high probability of finding a feasible path, x needs to be as large as $10n$, resulting in computational complexity of $\mathcal{O}(n^4)$. In [34] this heuristic algorithm is generalized to more than two constraints with the complexity of $\mathcal{O}(x_1^2 \cdot \ldots \cdot x_{K-1}^2 n^2)$ or $\mathcal{O}(x_1 \cdot \ldots \cdot x_{K-1} nm)$, where $x_1, \ldots, x_{K-1}$ are adjustable integers for each constraint. In [35] the authors used the k-shortest path algorithm in [36] with a nonlinear cost function to solve the MCP problem. The resulting algorithm, called TAMCRA, has a complexity of $\mathcal{O}(km\log(kn) + k^3m)$, where k is the number of shortest paths. As mentioned above, the performance and overhead of this algorithm depend on k. If k is large, the algorithm gives good performance at the expense of excessive computational cost. The above algorithms are especially proposed for the MCP problem (i.e., they do not attempt to optimize the selection of the feasible path).

Other works in the literature were aimed at addressing special yet important cases of the QoS routing problem. For example, several researchers addressed the QoS routing in the context of bandwidth and delay parameters. Showing that the feasibility problem under this combination is not NP-complete, the authors in [14] presented a *bandwidth-delay based routing algorithm* which simply prunes all links that do not satisfy the bandwidth requirement and then finds the shortest path w.r.t. delay in the pruned graph. Several path selection algorithms based on different combinations of bandwidth, delay, and hop-count were discussed in [37] (e.g., widest-shortest path, shortest-widest path). In addition, new algorithms were proposed to find more than one feasible path w.r.t. bandwidth and delay (e.g., Maximally Disjoint Shortest and Widest Paths (MADSWIP)) [38]. In [39] the authors proposed bandwidth guaranteed dynamic routing algorithms. In [40] the authors considered pre-computation of paths with minimum hop-count and bandwidth guarantee. They also provided some approximation algorithms that takes into account general additive constraints during the pre-computation. In [41] the authors investigated how to set link weights based on the previous measurements so that the shortest paths can provide better load balancing and can meet the desired QoS requirements. Some researchers considered the fallback routing approach [42], [43], in which QoS parameters are ordered and the optimal path w.r.t. each single parameter in this order is found until the returned path is feasible w.r.t. all constraints. Another approach to QoS routing is to exploit the dependencies between the QoS parameters and solve the path selection problem assuming specific scheduling schemes at network routers [44], [45]. Specifically, if Weighted

Fair Queueing (WFQ) scheduling is being used and the constraints are bandwidth, queueing delay, jitter, and loss, then the problem can be reduced to standard shortest path problem by representing all the constraints in terms of bandwidth. Although queueing delay can be formulated as a function of bandwidth, this is not the case for the propagation delay, which needs to be taken into account for QoS routing in high-speed networks [46].

Contributions and Organization of the Paper

As reviewed above, previously proposed algorithms consider only special cases of the MCOP problem (e.g., RSP, MCP) and suffer from either excessive computational complexities or low performance. In this paper, we investigate an efficient heuristic algorithm to the general MCOP problem. By general, we mean that our solution is applicable to any number of constraints, irrespective of their nature and interdependence. In Section II, we first describe how to deal with multiple constraints using a nonlinear cost function, which involves a predetermined constant λ. We show that the minimization of this cost function gives new insights into finding a feasible path in the MCP problem by offering a continuous spectrum of solutions ranging from a simple, linear approximation ($\lambda = 1$) to an asymptotically optimal solution ($\lambda \rightarrow \infty$). The same nonlinear cost function was also used in [35] to develop the TAMCRA algorithm for the MCP problem (without path optimization). In Section III, we introduce a new efficient heuristic algorithm (H_MCOP) to minimize the nonlinear cost function for finding a feasible path while also incorporating the optimization of the selected feasible path. We prove that H_MCOP guarantees at least the performance that can be obtained by a generalized linear approximation algorithm and often improves upon it. The computational complexity of H_MCOP is two times that of Dijkstra's algorithm. Using extensive simulations in Section IV, we show that H_MCOP always outperforms its contenders in finding feasible paths under the same order of complexity. Our main conclusions are drawn in Section V.

II. Nonlinear Cost Function for MCP

Consider the following cost function for any path p from the source to the destination:

$$g_\lambda(p) \stackrel{\text{def}}{=} (\frac{w_1(p)}{c_1})^\lambda + (\frac{w_2(p)}{c_2})^\lambda + \cdots + (\frac{w_K(p)}{c_K})^\lambda \qquad (1)$$

where $\lambda \geq 1$. Suppose there is an algorithm $\mathcal{X}$ that returns a path p by minimizing the cost function (1) for a given $\lambda \geq 1$. Then, the following bounds on the performance of algorithm $\mathcal{X}$ can be established.

Theorem 1: Consider the MCP problem (i.e., the MCOP problem without optimizing the selection of a feasible path). Assume that there is at least one feasible path $p*$ in the network. Let p be a path that minimizes the cost function g_λ for a given $\lambda \geq 1$. Then, (i) $w_k(p) \leq c_k$ for at least one k, and (ii) $w_k(p) \leq \sqrt[\lambda]{K}c_k$ for all other k's.

Proof: If the returned path p is feasible, then from (1) the above bounds are correct. Assume that p is not feasible. Since the algorithm returns the path p (and not the feasible path $p*$), it must be true that

$$g_\lambda(p) \leq g_\lambda(p*)$$

In addition, since $w_k(p*) \leq c_k$ for all k's, we have

$$g_\lambda(p*) \leq K$$

Thus,

$$g_\lambda(p) \leq K \qquad (2)$$

If $w_k(p) > c_k$ for all k's, then $g_\lambda(p) > K$. Since this contradicts (2), we must have $w_k(p) \leq c_k$ for at least one k, and the bound in part (i) is correct. Note that if $g_\lambda(p) > K$, then it is guaranteed that there is no feasible path $p*$ in G because for at least one k, $w_k(q) > c_k$ for every path q.

To prove part (ii), assume to the contrary that for at least one constraint c_j we have $w_j(p) > \sqrt[\lambda]{K}c_j$, so that $(\frac{w_j(p)}{c_j})^\lambda > K$. It readily follows that $g_\lambda(p) \geq (\frac{w_j(p)}{c_j})^\lambda > K$, which contradicts (2). Hence, part (ii) is proved. ■

Corollary 1: As λ increases, the likelihood of finding a feasible path also increases.

Proof: Follows immediately from Theorem 1 ($w_k(p) \leq \sqrt[\lambda+\delta]{K}c_k < \sqrt[\lambda]{K}c_k$, for any $\delta > 0$). ■

Therefore, to increase the probability of finding a feasible path, it makes sense to set λ to its largest possible value, i.e., $\lambda \rightarrow \infty$. In order to provide a practical computational model for $\lambda \rightarrow \infty$, we can replace the cost function $g^*(p) \stackrel{\text{def}}{=} \lim_{\lambda\rightarrow\infty} g_\lambda(p)$ by another cost function that does not explicitly involve λ but that achieves the same ordering of candidate paths as g^*. More precisely, we consider the following cost function for a path p [35]:

$$h(p) \stackrel{\text{def}}{=} \max\{\frac{w_1(p)}{c_1}, \frac{w_2(p)}{c_2}, \cdots, \frac{w_K(p)}{c_K}\} \qquad (3)$$

The following theorem establishes the equivalence of g^* and h.

Theorem 2: Let p_1 and p_2 be any two paths. Then $g^*(p_1) \leq g^*(p_2)$ iff $h(p_1) \leq h(p_2)$.

Proof: It is obvious that as $\lambda \rightarrow \infty$, $g_\lambda(p)$ is dominated by the largest term in (1), or equivalently, by $\max\{\frac{w_1(p)}{c_1}, \frac{w_2(p)}{c_2}, \cdots, \frac{w_K(p)}{c_K}\}$. A similar argument can be used to establish the proof in the other direction. ■

Figure 1 depicts a pictorial illustration of how algorithm $\mathcal{X}$ finds a feasible path with three different values of λ. The shaded area represents the feasibility region in the 2D parameter space (i.e., two weights are associated with each link). The black dots represent the normalized costs of various paths w.r.t. w_1 and w_2. Each contour line in the figure indicates paths with equal value w.r.t. the given cost function. Starting at the origin, algorithm $\mathcal{X}$ slides the fixed-cost contour line outward in the direction of the arrow until it hits a path (i.e., black dot in the figure). The returned path has the minimum cost w.r.t. g_λ. As shown in the figure, the larger the λ value, the closer the shape of the contour lines to that of the (hypercube) feasibility region. As $\lambda \rightarrow \infty$, algorithm $\mathcal{X}$ becomes exact, i.e., it is guaranteed to find a feasible path if one exists.

For $\lambda = 1$ it is easy to develop a polynomial-time algorithm that minimizes $g_1(p)$. This is done by assigning a combined weight $l(e) = \frac{w_1(e)}{c_1} + \frac{w_2(e)}{c_2} + \cdots + \frac{w_K(e)}{c_K}$ to every link e and finding the shortest path w.r.t. $l(e)$ using Dijkstra's algorithm. From Theorem 1, the returned path p satisfies the following bounds: (i) $w_k(p) \leq c_k$ for at least one k, and (ii) $w_k(p) \leq Kc_k$ for all other

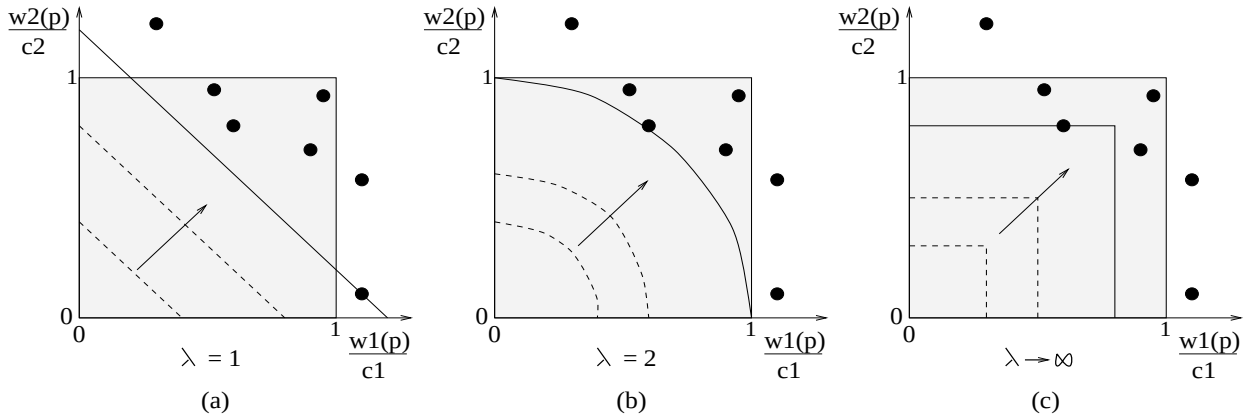

Fig. 1. Searching for a feasible path that minimizes $g_\lambda(p)$.

k's. As a matter of fact, this is a generalized linear approximation algorithm that applies to any number of constraints. It includes as special cases the approximation algorithms developed in [17] (except for the normalization factors).

For $\lambda > 1$ the nonlinearity of (1) makes it impossible to provide an exact polynomial-time minimization algorithm. Hence, one has to rely on heuristics. One such heuristics (the TAMCRA) was proposed in [35], which aims only at finding a feasible path based on the k-shortest path algorithm. Since no path optimization is performed, the selected path, albeit feasible, may be undesirable from a cost standpoint. Our goal is to provide a new heuristic algorithm that addresses both the feasibility aspect as well as the cost effectiveness of the selected path

Due to the heuristic nature of our algorithm, its performance may not always improve monotonically with λ, i.e., it is possible that the heuristic search fails to find a feasible path based on λ_2, which otherwise can be found based on $\lambda_1 < \lambda_2$). However, simulation results show that increasing λ most often results in performance improvement. Hence, it makes sense to base the design of our heuristic on the cost function g^*, or equivalently h.

III. Proposed Heuristic for MCOP

We now present our heuristic algorithm H_MCOP, which attempts to find a feasible path subject to K additive constraints and, simultaneously, minimize the cost of that path. For the feasibility part, H_MCOP tries to minimize the objective function g_λ for $\lambda > 1$. In doing so, it first exactly finds the best path w.r.t. g_1 from each node u to t. It then starts from s and discovers each node u based on the minimization of $g_\lambda(P)$, where P is a complete s-t path passing through node u. This s-t path is heuristically determined at node u by concatenating the already traveled segment from s to u and the estimated remaining segment (the above best path w.r.t. g_1) from u to t. Since the algorithm considers complete paths, it can foresee several paths before reaching the destination. For the optimality part, If some of these foreseen paths are feasible, H_MCOP selects the one that minimizes the primary cost function rather than the one that minimizes the nonlinear cost function. Using this preference rule (i.e., minimize the primary cost function if the foreseen path is feasible; otherwise, minimize the nonlinear cost function), H_MCOP can be implemented as simple as single-objective algorithms.

A pseudocode for H_MCOP is shown in Figure 2. Its inputs are a directed graph $G = (V, E)$ in which each link (i, j) is associated with a primary cost $c(i, j)$ and K weights $w_k(i, j)$, $k = 1, 2, \ldots, K$; a source node s; a destination node t; and K constraints c_k, $k = 1, 2, \ldots, K$. For each node u, the algorithm maintains the following labels: $r[u]$, $R_k[u]$, $\pi_r[u]$, $g[u]$, $G_k[u]$, $\pi_g[u]$, and $c[u]$, $k = 1, 2, \ldots, K$. Label $r[u]$ represents the cost of the shortest path from u to t w.r.t. the cost function g_1 (i.e., g_λ with $\lambda = 1$). Labels $R_k[u]$, $k = 1, 2, \ldots, K$, represent the individually accumulated link weights along that path. The predecessor of u on this optimal path is stored in label $\pi_r[u]$. Label $g[u]$ represents the cost of a foreseen complete path that goes from s to t via node u w.r.t. the cost function h (or g_λ, $\lambda > 1$). Labels $G_k[u]$, $k = 1, 2, \ldots, K$, and $c[u]$ represent the individually accumulated cost of link weights and the primary cost along the already traveled segment of this path from s to u. The predecessor of u on this path is stored in the label $\pi_g[u]$.

H_MCOP$(G = (V, E), s, t, c_k, k = 1, 2, \ldots, K)$
1 Reverse_Dijkstra($G = (V, E), t$);
2 **if** $r[s] > K$ **then**
3 return failure /* there is no feasible path */
4 **end if**
5 Look_Ahead_Dijkstra($G = (V, E), s$);
6 **if** $G_k[t] \leq c_k \; \forall \, k = 1, 2, \ldots, K$ **then**
7 return the path /* a feasible path is found */
8 **end if**
9 return failure

Fig. 2. The heuristic algorithm for the MCOP problem.

There are two directions in the algorithm: backward (from t to s) to estimate the cost of the remaining segment using $\lambda = 1$ and forward (from s to t) to find the most promising path in terms of feasibility and optimality using $\lambda > 1$. In the backward direction (lines 1-4 in H_MCOP), the algorithm finds the optimal path from every node u to t w.r.t. the cost function $g_1(\cdot)$. For that, it uses Reverse_Dijkstra [15] with some modifications to the relaxation process, as shown in Figure 3. Reverse_Dijkstra initially sets $r[u] = \infty$ and $\pi_r[u] = NIL$ for every node u. It then starts at

Reverse_Dijkstra_Relax(u, v)
1 **if** $r[u] > \sum_{k=1}^{K} \frac{R_k[v]+w_k(u,v)}{c_k}$ **then**
2 $\quad r[u] := \sum_{k=1}^{K} \frac{R_k[v]+w_k(u,v)}{c_k}$
3 $\quad R_k[u] := R_k[v] + w_k(u, v)$ for $k = 1, 2, \ldots, K$
4 $\quad \pi_r[v] := u$
5 **end if**

Fig. 3. Modified relaxation procedure for Reverse_Dijkstra based on minimizing $g_1(\cdot)$.

node t by setting $r[t]$ and $R_k[t]$, $k = 1, 2, \ldots, K$, to zeros. It explores the graph and eventually returns a path p from s to t. Before proceeding further, the algorithm checks $r[s] > K$ to determine the possibility of discovering a feasible path (based on the proof of Theorem 1, $r[s] > K$ implies necessarily the nonexistence of a feasible path).

If there is a possibility that the network contains a feasible path, a heuristic search procedure called Look_Ahead_Dijkstra is executed in the forward direction (line 5 in H_MCOP). This procedure uses the information provided by the above Reverse_Dijkstra to identify whether there is another path q which *provably* improves the performance over the above returned path p. To implement Look_Ahead_Dijkstra, we need to slightly modify the relaxation process of Dijkstra's algorithm [47], as shown in Figure 4.

Look_Ahead_Dijkstra_Relax(u, v)
1 Let tmp be a temporary node
2 $c[tmp] := c[u] + c(u, v)$
3a **if** $\lambda < \infty$ **then**
$\quad g[tmp] := \sum_{k=1}^{K} (\frac{G_k[u]+w_k(u,v)+R_k[v]}{c_k})^{\lambda}$
3b **if** $\lambda = \infty$ **then**
$\quad g[tmp] := \max\{\frac{G_k[u]+w_k(u,v)+R_k[v]}{c_k} \mid 1 \le k \le K\}$
4 $G_k[tmp] := G_k[u] + w_k(u, v)$ for $k = 1, 2, \ldots, K$
5 $R_k[tmp] := R_k[v]$ for $k = 1, 2, \ldots, K$
6 **if** Prefer_the_best$(tmp, v) = tmp$) **then**
7 $\quad c[v] := c[tmp]$
8 $\quad g[v] := g[tmp]$
9 $\quad G_k[v] := G_k[tmp]$ for $k = 1, 2, \ldots, K$
10 $\quad \pi_g[v] := u$
11 **end if**

Fig. 4. Modified relaxation procedure for Look_Ahead_Dijkstra of H_MCOP.

Look_Ahead_Dijkstra initially sets $g[u] = \infty$ and $\pi_g[u] = NIL$ for every node u. It then starts from node s, setting $g[s]$, $c[s]$, and $G_k[s]$, $k = 1, 2, \ldots, K$, to zeros. It explores the graph by choosing the next node based on the preference rule in Figure 5. This rule takes as input two nodes and their labels. It then selects one of these nodes such that the selected one minimizes the primary cost function if foreseen s-t paths passing through these nodes are feasible; otherwise, it selects the one that minimizes the

Prefer_the_best(a, b)
1 **if** $c[a] < c[b]$ **and** $\forall k \;\; G_k[a] + R_k[a] \le c_k$ **then return**(a)
2 **if** $c[a] > c[b]$ **and** $\forall k \;\; G_k[b] + R_k[b] \le c_k$ **then return**(b)
3 **if** $g[a] < g[b]$ **then return**(a)
4 **return**(b)

Fig. 5. Preference rule used in H_MCOP to choose between two nodes a and b.

objective function g_λ.

Eventually, H_MCOP returns a path q from s to t using $\lambda > 1$. The following theorem guarantees that q cannot be worse than the path p found using $\lambda = 1$, i.e., q has either less primary cost than p if p is feasible, or it has more chance of being feasible than p if p is not feasible. This theorem also states that H_MCOP guarantees at least the performance of the linear approximation algorithm with $\lambda = 1$ and often improves upon it.

Theorem 3: Suppose that H_MCOP algorithm returns the path p by searching backward from t to s (using $\lambda_1 = 1$) and, subsequently, returns the path q by searching forward from s to t (using $g_{\lambda_2}()$ with $\lambda_2 > 1$). Then, (i) if p is feasible, q is feasible and $c(q) \le c(p)$; (ii) if p is not feasible, $g_{\lambda_2}(q) \le g_{\lambda_2}(p)$.

Proof: Assume that p consists of l nodes $(v_0, v_1, v_2, \ldots, v_l)$ where $v_0 = s$ and $v_l = t$. In the forward direction, the algorithm discovers the neighboring nodes to s and explores the graph from one of these nodes for which either the foreseen path is feasible and $c[\cdot]$ is minimum or $g[\cdot]$ is minimum when there is no foreseen feasible path. As in the relaxation procedure, H_MCOP selects the next node based on the preference rule given by the procedure Prefer_the_best. Since v_1 is a neighbor of s, the algorithm will consider v_1 at the first time. If the foreseen path at v_1 is feasible (i.e., p is feasible) and $c[v_1]$ is minimum, then the algorithm will continue to explore the graph from v_1, visiting nodes $v_2, v_3, \ldots, v_l$, as long as they have the minimum $c[\cdot]$. Otherwise, the algorithm will explore another node, say u, from which a foreseen path is also feasible but $c[u] \le c[v_1]$; in this case, the algorithm will return a path whose cost is lower than the cost of p. Thus, part (i) is correct. If no foreseen path is feasible, then the algorithm explores the graph based on the minimum $g[.]$. Again the algorithm considers v_1 at the first time. If $g[v_1]$ is minimum, then the algorithm will explore the graph from v_1 and continue to explore from the other nodes $v_2, v_3, \ldots, v_l$ as long as they have the minimum $g[\cdot]$. Otherwise, the algorithm will explore another node whose d value is less than $g[v_1]$ and finds a better path than p in terms of feasibility. Thus, part (ii) is also correct. As a result, the returned path q will be either better than p or at least as good as p in terms of both feasibility and optimality. ■

The computational and space complexities of the resulting H_MCOP algorithm are equal to that of Dijkstra's, since *at most* two modified versions of Dijkstra's algorithm are executed with the complexity of $\mathcal{O}(n \log(n)+m)$. To improve the performance, the forward direction of H_MCOP can also be used with the k-shortest path implementation of Dijkstra's algorithm presented in [36]. Note that k-shortest paths are considered with respect to the minimization of the nonlinear cost function. The complexity of this k-shortest path algorithm is $\mathcal{O}(km \log(kn)+k^2 m)$. Hence,

the complexity of H_MCOP with k-shortest paths is $\mathcal{O}(n\log(n)+km\log(kn)+(k^2+1)m)$. Note that TAMCRA uses the same k-shortest path algorithm with the nonlinear cost function h. However, its complexity is $\mathcal{O}(km\log(kn)+k^3m)$ because of the extra computation to determine "dominated" paths. Although both algorithms have comparable worst-case complexities, our simulation results (shown in the next section) indicate that H_MCOP has a much lower average complexity than TAMCRA, i.e., to achieve the same performance, TAMCRA requires a much higher value of k than H_MCOP. Furthermore, due to the additional path optimization feature of H_MCOP, its returned paths are much more resource efficient than their TAMCRA counterparts.

Example

The following example illustrates the operation of H_MCOP under the nonlinear cost function h. For simplicity, we exclude path optimization from this example. Consider the network in Figure 6(a). For simplicity, assume that each link has two weights w_1 and w_2 and that links are symmetric (note that H_MCOP can run on asymmetric links with multiple real-valued weights). Suppose that a path is to be found from s to t which satisfies the constraints $c_1 = 10$ and $c_2 = 10$. Figure 6 describes the steps taken by H_MCOP to discover such a path. In the backward direction (Figures 6(a)-(c)), Reverse_Dijkstra finds a path from every node to the destination node t. Since the returned path (s,u,v,t) is not feasible and the value of the cost function ($r[s] = 1.6$) is less than $K = 2$, the algorithm proceeds to search for a feasible path in the forward direction using Look_Ahead_Dijkstra. Although Reverse_Dijkstra cannot find a feasible path in this example, it provides useful information (labels $R_k[u]$) for Look_Ahead_Dijkstra, enabling it to find a feasible path. Figures 6(d)-(e) show the state of the algorithm during the execution of Look_Ahead_Dijkstra based on the cost function h. The algorithm starts from s and discovers its neighbors u and v by relaxing links (s,u) and (s,v). The process of relaxing a link (i,j) consists of testing whether the cost of the foreseen path that goes through j can be improved by going through i to j and, if so, of updating the new values of the cost function and the predecessor of node j [47]. The algorithm then selects node v at which the value of the cost function is minimum and tries to discover its neighbors. Since the value of the cost function at node t decreases if link (v,t) is used, the algorithm relaxes this link. However, it cannot relax link (v,u) since the value of the cost function at node u does not decrease if this link is used. Now there are two nodes u and t to explore the graph. Since the value of the cost function at t is minimum, the algorithm selects it but cannot relax any more links. Finally, the algorithm selects u and relaxes only the link (u,t) since the value of the cost function at t is decreased through this relaxation. Finally, the algorithm returns the feasible path (s,u,t).

IV. Performance Evaluation

From Corollary 1, it is expected that the performance of H_MCOP improves with λ, although the trend may not be monotone due to the approximate nature of the algorithm. Our first goal is to verify this intuition. We then continue our investigations, using h (which is equivalent to g^*) for path selection in the H_MCOP algorithm. We contrast H_MCOP with Jaffe's two approximation algorithms (one with $\alpha = \beta = 1$ while the other with $\alpha = 1$ and $\beta = \sqrt{c_1/c_2}$) [17]), Chen's heuristic algorithm [33], ϵ-optimal algorithm [18], and TAMCRA [35]. After observing that H_MCOP and TAMCRA give significantly better performance than the other algorithms, we continue to contrast the two algorithms in more detail. Recall that TAMCRA only addresses the MCP problem, without attempting to optimize the selection of a feasible path. Several random network topologies has been studied in our simulations and similar trends have been observed across these topologies. For brevity, we report the results obtained under two constraints for random topologies with 50, 100, and 200 nodes and with correlated and uncorrelated link weights.

A. Simulation Model and Performance Measures

In our simulations, using the random graph generator package in [48], 50-, 100-, and 200-node topologies are generated based on Waxman's model [49]. We then associate two randomly generated weights with each link (i,j). As shown in Table I, these weights are selected from uniform distributions under several types of correlation between them. If there is positive correlation, we assume that both weights are selected from uniform distributions with either small mean or large mean. If there is negative correlation, we assume that one of weights is selected from a uniform distribution with small mean while the other is selected from another uniform distribution with large mean. If there is no specific correlation, we assume both weights are independently selected from uniform distributions. The primary cost of a link (i,j) is taken as $c(i,j) \sim uniform[1,200]$. To test the algorithms under the critical cases, the source and destination of a request are randomly generated such that the minimum hop-count between them is at least three. The constraints are also randomly generated, but their ranges are determined based on the best paths w.r.t. w_1 and w_2 as follows. Let p_1 and p_2 be two shortest paths from s to t w.r.t. w_1 and w_2, respectively. We take $c_1 \sim uniform[0.8*w_1(p_2), 1.2*w_1(p_2)]$ and $c_2 \sim uniform[0.8*w_1(p_1), 1.2*w_2(p_1)]$. The shaded box in Figure 7 represents the region where the constraints are selected. We extended this region in several directions (up, down, left, right). Although the absolute values of performance measures can change, the relative difference and the ranking among the compared algorithms do not change.

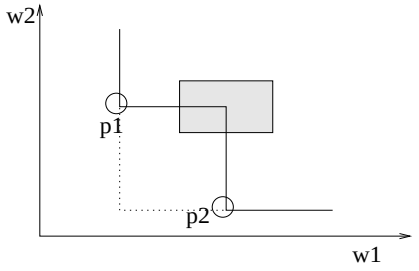

Fig. 7. The selection of constrains.

We contrast the performance of various path selection algorithms using the *success ratio* (SR), which refers to the fraction of connection requests for which feasible paths are found by the given heuristic or approximation algorithm, and the *average value of the primary cost function per routed connection* (AvgCost), where a *routed connection request* is one for which the given path selection algorithm returns a feasible path. AvgCost shows

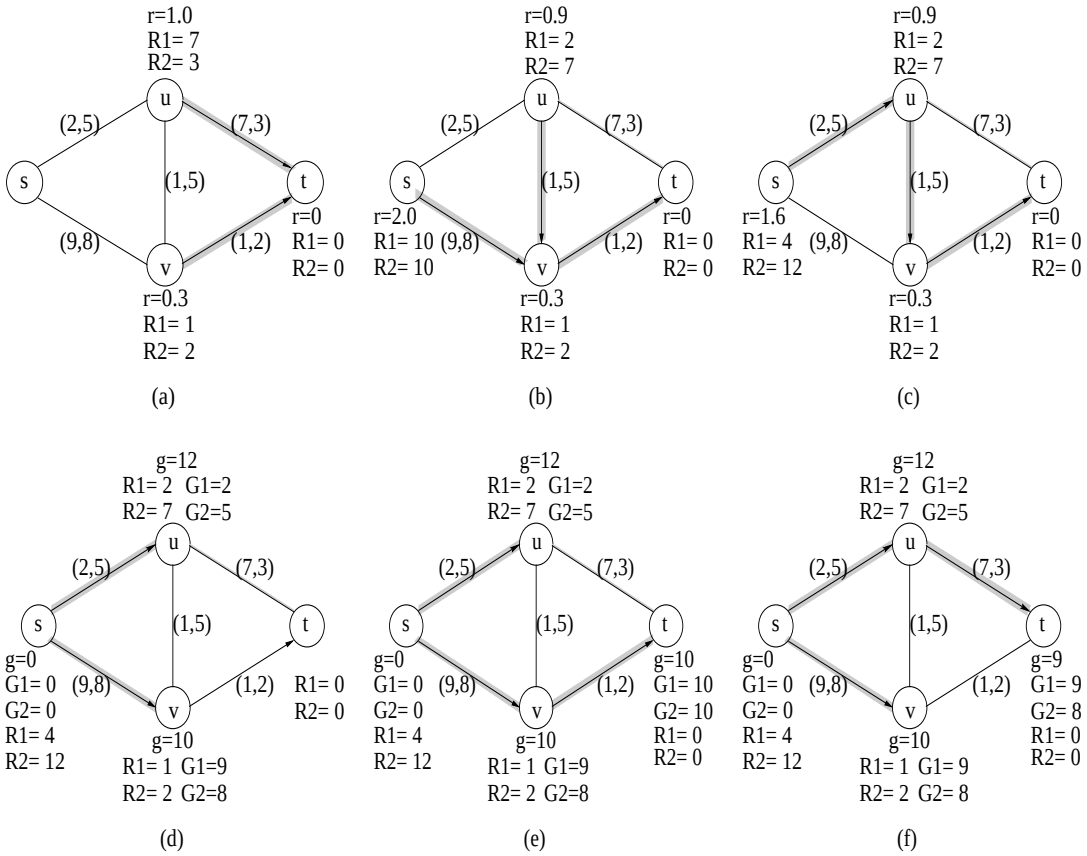

Fig. 6. Example that illustrates the operation of H_MCP. Shaded edges indicate the selected path. In (a)-(c) Reverse_Dijkstra returns an infeasible path. In (d)-(e) a feasible path is discovered using Look_Ahead_Dijkstra.

TABLE I

RANGES OF LINK WEIGHTS AND THE CORRELATION BETWEEN THEM.

Positive correlation	No correlation	Negative correlation
$w_1(i,j) \sim uniform[1,50]$	$w_1(i,j) \sim uniform[1,100]$	$w_1(i,j) \sim uniform[1,50]$
$w_2(i,j) \sim uniform[1,100]$	$w_2(i,j) \sim uniform[1,200]$	$w_2(i,j) \sim uniform[100,200]$
OR		OR
$w_1(i,j) \sim uniform[50,100]$		$w_1(i,j) \sim uniform[50,100]$
$w_2(i,j) \sim uniform[100,200]$		$w_2(i,j) \sim uniform[1,100]$

how costly a feasible path is, on average. The results reported in the subsequent sections are averaged over several runs and the 95% confidence intervals are computed. In each run, ten random graphs are generated. For each instance of a random graph, ten independent realizations of link weights are generated using different random seeds. Finally, for each instance of a random graph with given link weights (there are a total of 100 of such instances per experiment), about 2000, 2500, and 3000 connection requests are generated for graphs with 50, 100, and 200 nodes, respectively.

B. Performance of H_MCOP with Different Values of λ

Because H_MCOP is only an approximation of an exact (but nonexistent) algorithm $\mathcal{X}$ that minimizes g_λ, it is possible that the performance of H_MCOP does not always improve with λ. However, the following simulations show that almost always, increasing λ will improve the SR of H_MCOP, which gives the justification for using g^* (equivalently, h) in the design of H_MCOP. Figure 8 shows the average SR with the 95% confidence interval versus λ for random graphs with 50, 100, and 200 nodes and with uncorrelated, positively correlated, and negatively correlated link weights. In particular, the difference in performance between the case of $\lambda = 1$ and $\lambda = \infty$ is quite significant. Due to the heuristic nature of the algorithm, one can expect few anomalies in the general trend. However, these deviations are observed to be negligible in magnitude and frequency.

C. Performance Comparison with Other Path Selection Algorithms

We now contrast the performance of H_MCOP (based on the cost function h) against other path selection algorithms. Figure 9 shows the SRs of various algorithms. When the link weights are positively correlated, the path weights also become positively correlated, and thus a linear approximation algorithm has good chance of finding a feasible path by minimizing the linear com-

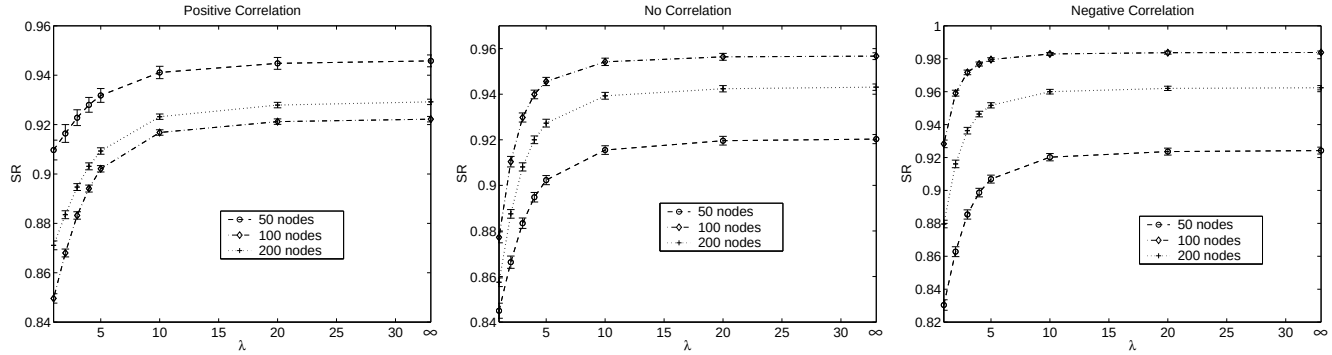

Fig. 8. The SR performance of H_MCOP with λ and the 95% confidence intervals.

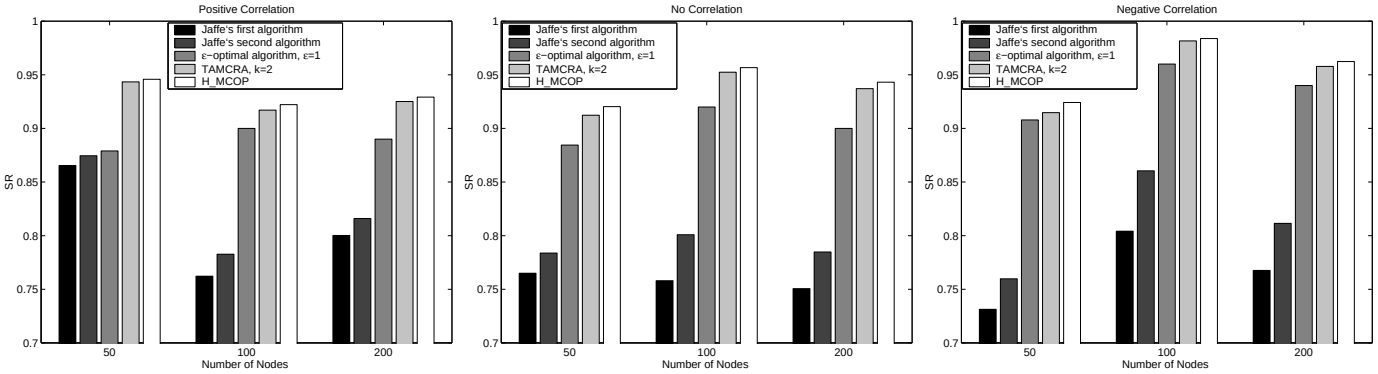

Fig. 9. SR of various algorithms used on random graphs with 50, 100, and 200 nodes.

bination of link weights. However, if the link weights are negatively correlated, there will be more paths in the network for which $w_1(p) \gg w_2(p)$, or vice versa. This degrades the performance of a linear approximation algorithm, which works best when the two link weights are comparable in value. Such a problem is absent in H_MCOP and TAMCRA due to their use of a nonlinear cost function. Hence, both algorithms provide significantly superior performance to Jaffe's linear approximations, especially when the link weights are not positively correlated. Note that H_MCOP requires at most two iterations of Dijkstra's algorithm while Jaffe's approximations require one. To compare H_MCOP with Chen's heuristic, we need to properly set the value of x of the latter algorithm. As x goes to infinity, the SR of Chen's heuristic approaches that of the optimal exponential-time algorithm. But given its $\mathcal{O}(x^2n^2)$ complexity, a large x clearly makes the algorithm impractical. To get as close as possible to achieving the same computational complexity of H_MCOP, x must be set to two. However, since in our simulations we consider paths with a minimum of three hops and Chen's algorithm finds paths with at most x hops, we set x to three. Even with $x = 3$, the SR of this algorithm still lags significantly behind others (its SR, which is not shown in the figure, is around 0.2). Note that even with such a small value of x, this algorithm requires nine iterations of Dijkstra, i.e., at least four times that of H_MCOP.

We also include in our comparisons the performance for one of the two ϵ-optimal algorithms proposed in [18]. These algorithms provide approximate solutions to the RSP problem, but are readily applicable to the underlying MCP problem. Their respective complexities are $\mathcal{O}(\log\log B(m(n/\epsilon)+\log\log B))$ and $\mathcal{O}(m(n^2/\epsilon)\log(n/\epsilon))$. We set ϵ to one, making the complexities of these algorithms at least $\mathcal{O}(m)$ times that of H_MCOP. In spite of the extra complexity, the examined ϵ-optimal algorithm is shown to lag behind H_MCOP in terms of the SR. To compare H_MCOP with TAMCRA, we need to properly set the value of k in the latter algorithm. As k goes to infinity, the performance of TAMCRA approaches that of the exact exponential-time algorithm. But given its $\mathcal{O}(km\log(kn)+k^3m)$ complexity, a large k clearly makes the algorithm impractical. To get as close as possible to achieving the same computational complexity of H_MCOP, k must be set to one or two. Figure 9 shows that with $k = 2$, TAMCRA performs worse than H_MCOP in terms of computational complexity and SR. However, since both algorithms (which use the same nonlinear cost function) perform better than the other algorithms under comparable computational complexities, we single them out and contrast their performances in more detail in the next section.

D. Detailed Performance Comparison Between H_MCOP and TAMCRA

In this section, we compare H_MCOP and TAMCRA in more detail using k-shortest paths for both. Figure 10 shows the SRs of both algorithms versus the number of considered shortest paths. The figure shows that H_MCOP gives better SR using smaller values of k than TAMCRA regardless of the number of nodes or the correlation between the link weights. In other words, the SR of TAMCRA increases with k and finally converges to the SR that is initially provided by H_MCOP with smaller k. For instance, the SR performance provided by H_MCOP with $k = 1$ can

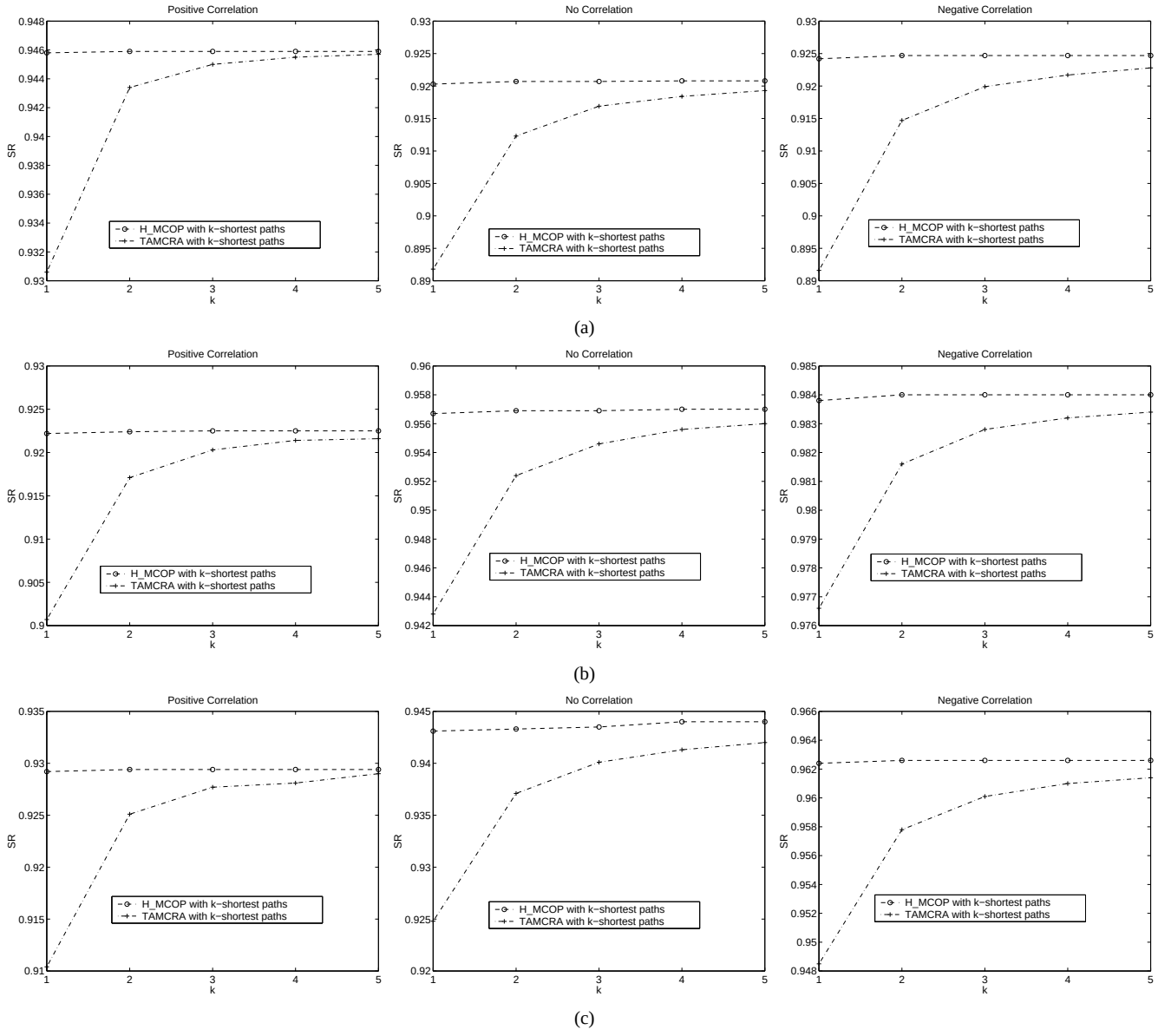

Fig. 10. SRs of H_MCOP and TAMCRA with k-shortest paths: (a) 50 nodes; (b) 100 nodes; (c) 200 nodes.

only be achieved by TAMCRA with k no less than 5. Given that the complexities of H_MCOP and TAMCRA are $\mathcal{O}(n\log(n) + km\log(kn) + (k^2+1)m)$ and $\mathcal{O}(km\log(kn) + k^3m)$, respectively, we can claim that H_MCOP outperforms TAMCRA in performance for the same amount of computational complexity. Note that the SR of H_MCOP barely changes with k, suggesting that this SR is probably very close to the SR of the exact, exponential-time algorithm (which, of course, cannot be obtained in practice as it requires enumerating all paths in the network).

While the above comparison demonstrates the performance advantage of H_MCOP over TAMCRA (for comparable computational complexities), the real advantage of H_MCOP is in terms of minimizing the cost of the path (i.e., selecting a resource-efficient feasible path). This is demonstrated in Table II, which shows the percentage reduction in the AvgCost when using H_MCOP instead of TAMCRA.

The AvgCost reduction is significant when there are several feasible paths to consider. In our simulation, the constraints are selected based on the best path w.r.t. each individual link weight. Thus, if the path weights are positively correlated, the constraints are selected from among a small set of feasible paths. As path weights become less (or negatively) correlated more paths are added to this set, making it more likely that a path selected by TAMCRA will not be the most cost effective; hence, increasing the cost reduction achieved through H_MCOP. The results also show that using the k-shortest path algorithm in conjunction with H_MCOP does not significantly change the the AvgCost reduction. The reason of this is most of the feasible paths are already found when $k = 1$, and thus increasing k makes algorithm to consider a few more feasible paths but minimizing their cost is not enough to significantly change the AvgCost reduction.

TABLE II
PERCENTAGE REDUCTION IN AVGCOST WHEN GOING FROM TAMCRA TO H_MCOP. THE LABELS '+', '−', AND '0' INDICATE POSITIVE, NEGATIVE, AND NO CORRELATION, RESPECTIVELY.

	50 nodes			100 nodes			200 nodes		
k	'+'	'0'	'−'	'+'	'−'	'0'	'+'	'−'	'0'
1	6.8%	12.1%	14.9%	18.3%	34.9%	38.5%	13.9%	25.8%	28.2%
2	7.3%	12.8%	15.5%	19.1%	35.4%	38.7%	14.5%	26.4%	28.5%
3	7.4%	13.1%	15.8%	19.4%	35.6%	38.9%	14.8%	26.6%	28.7%
4	7.5%	13.2%	15.9%	19.5%	35.7%	39.0%	15.2%	27.2%	28.8%
5	7.5%	13.3%	16.1%	19.7%	35.8%	39.0%	15.4%	27.3%	28.9%

V. CONCLUSIONS

Optimal path selection subject to multiple constraints is an NP-complete problem, which can only be addressed through heuristics and approximation algorithms. Previously proposed algorithms suffer from excessive computational complexities and/or low performance. Moreover, most of them are only applicable to special cases of the problem. In this paper, we investigated the general multi-constrained optimal path (MCOP) problem with the goal of developing highly efficient heuristics in terms of computational time, performance, and resource utilization. First, we investigated the theoretical properties of a nonlinear cost function, g_λ, that can be used as the basis for efficient heuristic solutions to the MCOP problem. We showed that as the nonlinearity parameter λ increases to infinity, the minimization of g_λ provides a better approximation to the MCP problem (the MCOP problem without path optimization). We demonstrated that a generalized linear approximation algorithm can be easily developed for $\lambda = 1$. For $\lambda > 1$, the nonlinearity of g_λ does not allow for an exactly polynomial path selection algorithm. Although finding a solution to the new minimization problem is left open, we provided an efficient heuristic algorithm (H_MCOP) that tries to approximate the minimization of $g^*_\lambda \stackrel{\text{def}}{=} \lim_{\lambda\to\infty} g_\lambda$. To optimize the use of resources while searching for a feasible path, H_MCOP also attempts to minimize a primary cost function. We proved that H_MCOP guarantees at least the same performance provided by a linear approximation algorithm, and most often provides significant improvements upon it. H_MCOP has the same order of complexity as Dijkstra's shortest path algorithm. For further performance improvement, H_MCOP can also be used in conjunction with the k-shortest path algorithm.

Using extensive simulations, we first verified that the performance of H_MCOP generally improves with λ, with few anomalies that are negligible in magnitude and frequency. We then contrasted H_MCOP with other contending algorithms. Our results show that at a fixed computational complexity, H_MCOP outperforms existing algorithms in terms of the SR, followed by TAMCRA (which also uses g^*_λ as a basis for path selection). Consequently, we compared H_MCOP with TAMCRA in more detail using the k-shortest paths in both. The results indicate that a much larger value of k (i.e., more computational time) is needed in TAMCRA to produce the same SR obtained under H_MCOP. Furthermore, H_MCOP reduces the AvgCost of the returned feasible paths over TAMCRA, thus providing more efficient use of network resources. The AvgCost reduction is significant when the network contains a large number of feasible paths. We also investigated the impact of correlated and uncorrelated link weights on the path selection algorithms. We observed that when the path weights are negatively correlated, i.e., $w_1(p) \gg w_2(p)$, or vice versa, linear approximation algorithms often return such paths that satisfy one constraint but violets the other. When link weights are positively correlated, linear approximation algorithms are more likely to succeed in finding feasible paths. The simulation results verified that when the link weights are positively correlated, the path weights also become positively correlated and thus the linear approximation algorithms often succeeds in finding feasible paths. However, when link weights are negatively correlated, the path weights tend to also be negatively correlated, degrading the performance of linear approximation algorithms. In all cases, H_MCOP was shown to provide better performance than linear approximation algorithms. When negative or no correlations are present between link weights, H_MCOP provides significant performance improvement upon linear approximation algorithms than when positive correlation exists.

H_MCOP performs well when the true state of the network is given. However, the true state may not be available to every node at all times due to network dynamics, aggregation of state information, and latencies in state dissemination. As a future work, we will investigate how H_MCOP performs in the presence of inaccurate state information and what modifications need to be done.

REFERENCES

[1] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi, "Quality of service based routing: A performance perspective," in *Proceedings of the ACM SIGCOMM '98 Conference*, Vancouver, British Columbia, Canada, August-September 1998, pp. 15–26, http://www.acm.org/sigcomm/sigcomm98/tp/abs_02.html.

[2] S. Chen and K. Nahrstedt, "An overview of quality-of-service routing for the next generation high-speed networks: Problems and solutions," *IEEE Network*, vol. 12, no. 6, pp. 64–79, Nov-Dec 1998.

[3] E. Crawley et al., "A framework for QoS-based routing in the Internet," Internet draft, IETF, July 10, 1998, (draft-ietf-qosr-framework-06.txt).

[4] R.A. Guerin and A. Orda, "Networks with advance reservations: the routing perspective," in *Proceedings of the INFOCOM '00 Conference*. IEEE, 2000, vol. 1, pp. 118–127.

[5] R. Vogel et al., "QoS-based routing of multimedia streams in computer networks," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1235–1244, September 1996.

[6] X. Xiao and L. M. Ni, "Internet QoS: A big picture," *IEEE Network*, vol. 13, no. 2, pp. 8–18, March-April 1999.

[7] The ATM Forum, "Private network-to-network interface specification version 1.0 (PNNI 1.0)," March 1996, af-pnni-0055.000.

[8] T.M. Chen and T.H. Oh, "Reliable services in MPLS," *IEEE Communication Magazine*, vol. 37, no. 12, pp. 58–62, Dec. 1999.

[9] J. Moy, "OSPF version 2," Standards Track RFC 2328, Internet Engineering Task Force, April 1998.

[10] G. Apostolopoulos et al., "QoS routing mechanisms and OSPF extensions," Tech. Rep. draft-guerin-qos-routing-ospf-05.txt, Internet Engineering Task Force, April 1998.

[11] R. Guerin and A. Orda, "QoS-based routing in networks with inaccurate information: Theory and algorithms," in *Proceedings of the INFOCOM '97 Conference*. IEEE, 1997, pp. 75–83.

[12] A. Alles, "ATM internetworking," White Paper, Cisco Systems, Inc., May 1995.

[13] Z. Wang, "On the complexity of quality of service routing," *Information Processing Letters*, vol. 69, no. 3, pp. 111–114, 1999.

[14] Z. Wang and J. Crowcroft, "Bandwidth-delay based routing algorithms," in *Proceedings of the GLOBECOM '95 Conference*. IEEE, Nov. 1995, vol. 3, pp. 2129–2133.

[15] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Inc., 1993.

[16] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[17] J. M. Jaffe, "Algorithms for finding paths with multiple constraints," *Networks*, vol. 14, pp. 95–116, 1984.

[18] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Mathematics of Operations Research*, vol. 17, no. 1, pp. 36–42, 1992.

[19] D.H. Lorenz, A. Orda, D. Raz, and Y. Shavitt, "Efficient QoS partition and routing of unicast and multicast," in *IWQoS 2000*, June 2000, pp. 75–83.

[20] C.A. Phillips, "The network inhibition problem," in *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing (STOC)*, May 1993, pp. 776–785.

[21] A. Orda, "Routing with end-to-end QoS guarantees in broadband networks," *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 365–374, 1999.

[22] F. Ergun, R. Sinha, and L. Zhang, "QoS routing with performance-dependent costs," in *Proceedings of the INFOCOM '00 Conference*. IEEE, 2000, vol. 1, pp. 137–146.

[23] R. Widyono, "The design and evaluation of routing algorithms for real-time channels," Tech. Rep. TR-94-024, University of California at Berkeley & International Computer Science Institute, June 1994.

[24] D. Eppstein, "Finding the k shortest paths," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. IEEE, Nov. 1994, pp. 154 – 165.

[25] L. Guo and I. Matta, "Search space reduction in QoS routing," in *Proceedings of the 19th IEEE International Conference on Distributed Computing Systems*. IEEE, May 1999, pp. 142 – 149.

[26] G. Y. Handler and I. Zang, "A dual algorithm for the constrained shortest path problem," *Networks*, vol. 10, pp. 293–310, 1980.

[27] Christopher C. Skiscim and Bruce L. Golden, "Solving k-shortest and constrained shortest path problems efficiently," *Ann. Oper. Res.*, vol. 20, no. 1-4, pp. 249–282, 1989.

[28] Y. P. Aneja, V. Aggarwal, and K. P. K. Nair, "Shortest chain subject to side constraints," *Networks*, vol. 13, pp. 295–302, 1983.

[29] H. F. Salama, D. S. Reeves, and Y. Viniotis, "A distributed algorithm for delay-constrained unicast routing," in *Proceedings of the INFOCOM '97 Conference*. IEEE, 7-11 April 1997, vol. 1, pp. 84–91.

[30] K. Ishida, K. Amano, and N. Kannari, "A delay-constrained least-cost path routing protocol and the synthesis method," in *Proceedings of the Fifth International Conference on Real-Time Computing Systems and Applications*. IEEE, Oct. 1998, pp. 58 – 65.

[31] J. Zhou, "A new distributed routing algorithm for supporting delay-sensitive applications," in *Proceedings of ICCT '98*. IEEE, 22-24 Oct. 1998, pp. S37–06(1–7).

[32] D. Blokh and G. Gutin, "An approximation algorithm for combinatorial optimization problems with two parameters," IMADA preprint PP-1995-14, May 1995, http://www.imada.ou.dk/Research/Preprints/Abstracts/1995/14.html.

[33] S. Chen and K. Nahrstedt, "On finding multi-constrained paths," in *Proceedings of the ICC '98 Conference*. IEEE, 1998, pp. 874 –879.

[34] S. Chen and K. Nahrstedt, "On finding multi-constrained paths," Tech. Report UIUCDCS-R-97-2026, Dept. of Computer Science, Uni. of Illinois at Urbana-Champaign, August 1997.

[35] H. De Neve and P. Van Mieghem, "A multiple quality of service routing algorithm for PNNI," in *Proceedings of the ATM Workshop*. IEEE, May 1998, pp. 324 – 328.

[36] E. I. Chong, S. R. Sanjeev Rao Maddila, and S. T. Morley, "On finding single-source single-destination k shortest paths," in *the Seventh International Conference on Computing and Information (ICCI '95)*, July 5-8, 1995, pp. 40–47, http://styx.trentu.ca/jci/icci/stream95.html.

[37] Q. Ma and P. Steenkiste, "Routing traffic with quality-of-service guarantees in integrated services networks," in *Proceedings of NOSSDAV '98*, http://www.cs.cmu.edu/∼qma/Publications.html, July 1998.

[38] N. Taft-Plotkin, B. Bellur, and R. Ogier, "Quality-of-service routing using maximally disjoint paths," in *the Seventh International Workshop on Quality of Service (IWQoS '99)*, London, England, May/June 1999, IEEE, pp. 119 – 128.

[39] M. Kodialam and T.V. Lakshman, "Dynamic routing of bandwidth guaranteed tunnels with restoration," in *Proceedings of the INFOCOM '00 Conference*. IEEE, 2000, vol. 2, pp. 902–911.

[40] A. Orda and A. Sprintson, "QoS routing: the precomputation perspective," in *Proceedings of the INFOCOM '00 Conference*. IEEE, 2000, vol. 1, pp. 128–136.

[41] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proceedings of the INFOCOM '00 Conference*. IEEE, 2000, vol. 2.

[42] W. C. Lee, M. G. Hluchyi, and P. A. Humblet, "Routing subject to quality of service constraints in integrated communication networks," *IEEE Network*, pp. 46–55, July/August 1995.

[43] A. Iwata et al., "ATM routing algorithms with multiple QOS requirements for multimedia internetworking," *IEICE Trans. Commun.*, vol. E79-B, no. 8, pp. 999–1006, August 1996.

[44] Q. Ma and P. Steenkiste, "On path selection for traffic with bandwidth guarantees," in *Proceedings of the IEEE International Conference on Network Protocols (ICNP '97)*, 1997, pp. 191 –202.

[45] C. Pornavalai, G. Chakraborty, and N. Shiratori, "QoS based routing algorithm in integrated services packet networks," in *Proceedings of ICNP '97*. IEEE, 1997, pp. 167–174.

[46] D. Clark et al., "Strategic directions in networks and telecommunications," *ACM Computing Surveys*, vol. 28, no. 4, pp. 579–690, 1996.

[47] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, The MIT press and McGraw-Hill book company, sixteenth edition, 1996.

[48] K.I. Calvert, M.B. Doar, and E.W. Zegura, "Modeling internet topology," *IEEE Communications Magazine*, vol. 35, no. 6, pp. 160 – 163, June 1997.

[49] B.M. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, vol. 69, pp. 1617–1622, Dec. 1988.