Unicast and Multicast QoS Routing with Multiple Constraints

Dan Wang[1], Funda Ergun[1], and Zhan Xu[2]

[1] School of Computer Science, Simon Fraser University,
Burnaby BC V5A 1S6, Canada
{danw,funda}@cs.sfu.ca
[2] EECS Dept, Case Western Reserve University,
Cleveland OH 44106, USA
zxx10@po.cwru.edu

Abstract. We explore techniques for efficient Quality of Service Routing in the presence of multiple constraints. We first present a polynomial time approximation algorithm for the unicast case. We then explore the use of optimization techniques in devising heuristics for QoS routing both in the unicast and multicast settings using algorithmic techniques as well as techniques from optimization theory. We present test results showing that our techniques perform very well in the unicast case. For multicast with multiple constraints, we present the first results that we know of where one can quickly obtain near-optimal, feasible trees.

1 Introduction

As diverse applications such as online conferences, video broadcast, online auctions appear in the Internet, the demands by the application owners regarding the delivery of data have become more extensive and varied. Currently the delivery of such data primarily focuses on minimizing the path length, or obeying a given policy. However, the needs of the applications involve other issues such as latency, packet loss, jitter avoidance, etc. Such requirements of the applications from the network can be formally expressed in terms of *Quality of Service (QoS) constraints*. The satisfaction of these constraints comes at a cost of using valuable network resources such as buffer space, bandwidth, etc. The focus of QoS routing is to select the routes by taking into account the requirements of the applications while being efficient in terms of link costs.

Most QoS constraints are additive, such as delay, packet loss etc, i.e., they accumulate along the path. Given such constraints, in this paper, we explore QoS routing in unicast and multicast settings. Unicast QoS routing involves finding a min-cost path from a source to a destination node satisfying a set of constraints generally given as upper bounds that the path must respect. These problems are NP-complete when the number of constraints is one or higher[5]. In multicast QoS routing, given a source node and a set of multicast nodes, we seek to find a min-cost tree such that the constraints are satisfied along the path from the source to each multicast node. Even without constraints, this problem is NP-complete [10].

M. Ajmone Marsan et al. (Eds.): QoS-IP 2005, LNCS 3375, pp. 481–494, 2005.

The prohibitive hardness of these problems, and the requirements of a high-traffic network makes it necessary to develop techniques that efficiently generate near-optimal solutions. In this paper we investigate provably good, as well as practically feasible schemes. First, we present an ϵ-approximation algorithm for unicast routing with K constraints that runs in polynomial time for small K. We then develop heuristics for unicast and multicast QoS routing using algorithmic and optimization techniques with the goal of finding good solutions efficiently regardless of the number of constraints. We give out two heuristics for the unicast case, one of which is flexible with respect to optimality or feasibility. This is desirable since finding a feasible path is NP-complete if we have more than one constraint. Our algorithm explores the trade-off between cost and feasibility. As shown through our simulation results, our algorithms are very fast and obtain over 92% success rate for the unicast case. We also show with multicast multiconstraint routing, which is far more difficult than unicast routing, how to obtain a feasible solution with a high success rate.

1.1 Previous Work

Since this field is quite mature, we give a sample of the related work. Samples of abundant recent work can be found in [11, 2, 6, 3, 9], and their references. Recently several related work have appeared that use optimization techniques. In [11] a simple application of the technique is used for multiconstraint unicast routing and in [9] the technique is discussed for one constraint unicast problems. Our approximation algorithm builds on top of several work; for a full description of the algorithm see [14] and [4].

The Steiner Tree Problem is the simplest form of multicast routing and is well-studied [12, 8]. The heuristic KMB that we use as a building block is given in [12], and performs always within a factor 2 and usually within 10% of the optimal [15]. There are several results on single constraint QoS multicast routing, examples are in [13, 15], and their references.

2 The Multivariate QoS Routing Problem

We model our network as an undirected graph $G = (V, E)$, where V is the set of nodes, and E is the set of links; we assume $|V| = n$, and $|E| = m$ throughout. Each link e is associated with a cost $c(e)$ and K different QoS parameters, denoted $w_1(e) \dots w_K(e)$ representing end-to-end restrictions on the routes such as delay, packet loss, etc. Nodes s and t refer to the source and destination; for multicast the set $V_t = \{t_1, t_2, \dots, t_L\}$ refers to the set of multicast destinations. We denote by $P(s, t)$ the set of $s \to t$ paths in G and by $T(s, V_t)$ the set of multicast trees rooted at s with destinations V_t in G.

A constraint is an upper bound on the value of a QoS parameter for each destination[1]: W_{ij} refers to constraint i for destination t_j. When there is a single

[1] We use the terms "constraint" and "upper bound" interchangeably.

destination (unicast), we simplify constraint i as W_i. A path or a tree that satisfies all of the K constraints is said to be *feasible*; the least cost feasible path/tree is called *optimal*.

Unicast QoS Routing. Given source and destination nodes s, t, we aim to find the minimum $s \rightarrow t$ path p such that the sum of parameter w_i along p is upper bounded by W_i (for all $i = 1, \ldots, k$). The above can be formulated as follows.

$$\min_{p \in P(s,t)} \sum_{e \in p} c(e)$$

$$s.t. \sum_{e \in P} w_i(e) \leq W_i \ \ where \ \ i \in 1 \cdots K$$

Multicast QoS Routing. Given a source node s and a set of destinations $V_t = \{t_1, \cdots t_L\}$, we aim to find the min-cost multicast tree T rooted at s such that the sum of parameter w_i along path $s \rightarrow t_j$ is upper bounded by W_{ij} for each i, j. Formally, multicast QoS routing problem is as follows.

$$\min_{t \in T(s,V_t)} \sum_{e \in t} c(e)$$

$$s.t. \sum_{e \in P_t(s,t_j)} w_i(e) \leq W_{ij} \ \ for \ \ i \in 1 \cdots K, j \in 1 \cdots L$$

3 An ϵ-Approximation Algorithm for Unicast

In this section we give a polynomial time approximation algorithm that solves unicast QoS routing with K constraints. When $K = 1$, the problem is called Restricted Shortest Path (RSP), for which there are ϵ-approximation algorithms (see [14], [4] for the latest results).

Our algorithm has the following specifications. If a feasible path exists and the cost of the optimal feasible path is OPT, the algorithm is guaranteed to return a path p of cost $\leq (1+\epsilon)OPT$ such that (i) the total value for parameter w_1 on p is at most W_1, and (ii) for $i = 2 \ldots K$, the total value for QoS parameter w_i across p is at most $(1 + \epsilon)W_i$. Note that a slight violation of all constraints except w_1 is allowed.

We construct an approximation algorithm for $K = 2$; the technique generalizes easily to any K. We express the problem as a dynamic program. For node v, let $V(cc, d, v)$ denote the minimum value of parameter w_2 along an s-v path with total cost $\leq cc$ and total value of parameter w_1 at most d. Then, $V(cc, d, v) = min_{e=(v',v) \in E}\{V(cc - c(e), d - w_1(e), v') + w_2(e')\}$.[2] The base cases are that $\forall\ d, cc,\ V(cc, d, s) = 0$ (s-s path is trivial), $\forall\ cc, v \neq s$, $V(cc, 0, v) = \infty$ (parameters are positive), and $\forall\ d, v \neq s$, $V(0, d, v) = \infty$ (costs are positive). To solve the problem, we need to compute the smallest cost cc

[2] We assume all parameters are integers.

where $V(cc, W_1, t) \leq W_2$; the actual path is implicit in the formulation and can be obtained from the steps leading to this particular value.

Our approximation algorithm generalizes [7] by first approximately testing whether a feasible solution for a fixed budget V exists and using this test to search for the optimal solution. We proceed below by scaling down costs and w_2, and running the dynamic program on the scaled parameters.

procedure TestMult(V, ϵ)
Step 1 $\forall d, \forall cc$, $V(cc, d, s) = 0; \forall d, \forall v \neq s$, $V(0, d, v) = \infty$;
$\forall cc, \forall v \neq s$, $V(cc, 0, v) = \infty$;
Step 2 $\forall e \in E$, $c'(e) = \lfloor \frac{c(e)n}{\epsilon V} \rfloor$; $w_2'(e) = \lfloor \frac{w_2(e)n}{\epsilon W_2} \rfloor$.
Step 3 **for** $cc = 1 \ldots n/\epsilon$
$V(cc, d, v) = min_{e=(v',v)\in E}\{V(cc - c'(e), d - w_1(e), v') + w_2'(e')\}$.
Step 4 **if** $V(cc, W_1, t) \leq W_2$ **return** "SMALLER"
else return "GREATER".

We now show that TestMult is efficient and effective.

Lemma 1. *If TestMult* (V, ϵ) *returns "SMALLER" there is a path of cost* $\leq V(1+\epsilon)$*, total value of* $w_1 \leq W_1$*, and total* $w_2 \leq W_2(1+\epsilon)$*. If it returns "GREATER", then there is no path of cost at most* V *where parameter* w_1 *is at most* W_1 *and* w_2 *at most* W_2*. The running time of TestMult is* $O(mn^2/\epsilon^2)$*.*

Proof. If the procedure returns "SMALLER", then it must have found a path whose scaled total cost and scaled w_2 are $\leq n/\epsilon$ and sum of w_1 is $\leq W_1$. That W_1 is satisfied follows from the dynamic program. The scaled cost and w_2 values can be restored in the end by multiplying the values on the output path by $\epsilon V/n$ and $\epsilon W_2/n$ respectively. This way, on a path implicitly found feasible by TestMult, the cost is underestimated by at most $\epsilon V/n$ per link and the sum of the w_2 values are underestimated by at most $\epsilon W_2/n$. Since a path can have at most n links, the actual cost can be at most $(n/\epsilon)\cdot(\epsilon V/n) + n(\epsilon V/n) = V(1+\epsilon)$. A similar argument can be made for w_2. Now consider cases where TestMult returns "GREATER". Since the inaccuracy caused by the scaling only results in underestimation, clearly there cannot be a feasible solution with cost $\leq V$. The running time follows from the size of the table that one needs to keep for the dynamic program.

Equipped with a test, we can now search for the optimal cost with a multiplicative binary search. For this, we establish upper and lower bounds UB and LB for the cost, and search between them for the smallest cost that will make the test return a positive ("SMALLER") answer.

procedure ApproxMult(ϵ)
Step 1 Determine UB, LB
Step 2 **while** $UB/LB > 2$
$V = \sqrt{UB \times LB}$.
if TestMult(V, ϵ)= "GREATER" $LB = V$ **else** $UB = V(1+\epsilon)$
Step 3 Run a modified TestMult(LB, ϵ) with the loop in Step going up to $\frac{2n}{\epsilon}$,
return path with smallest cc where $V(cc, \frac{n}{\epsilon}, t) \leq W_1$

Next we show that this is an ϵ-approximation algorithm[3].

Theorem 1. *ApproxMult is an ϵ-approximation algorithm for QoS routing with two parameters which returns a path P that satisfies the following. The sum of w_1 along P is at most W_1, the sum of w_2 along P is at most $W_2(1+\epsilon)$, and the total cost is at most $OPT(1+\epsilon)$, where OPT is the cost of the cheapest path. ApproxMult runs in time in time $O(mn^2 \log\log(UB/LB)/\epsilon^2)$.*

A trivial initial value for LB is 1; one for UB is nC where C is the highest link cost in the network G. If one is willing to do some extra work, one can establish tighter lower and upper bounds. In fact, a generalization of the upper-bounding technique in [14] helps reduce the overall running time. The technique proceeds as follows. Initially we run Restricted Shortest Path on the network, trying to find a path that minimizes the sum of parameter w_2 such that the parameter w_1 adds up to W_1. (Recall that Restricted Shortest Path is the one constraint QoS routing problem and can be approximated in time $O(mn/\epsilon)$[4].) If the total value of w_2 on the path returned exceeds W_2, we conclude that there is no feasible solution and stop.

Otherwise we sort all links based on their costs. After that, we start removing links from the graph in order of their cost, starting from the one with the highest cost. After each link is removed, we run RSP as above, minimizing total w_1 and bounding the sum of w_2. At some point, it will become impossible to find a path whose total w_1 is under W_1. This means that the last link removed, say e, is a *bottleneck* link. We now make the following observation regarding the upper and the lower bounds with respect to the bottleneck link.

Lemma 2. *Let e be the bottleneck link. (1) There exists an s-t path P such that the sum of w_1 across P is at most W_1, the sum of w_2 is at most $(1+\epsilon)W_2$ and the total cost of P is at most $nc(e)$. (2) There exists no path of cost less than $c(e)$ that satisfies both constraints W_1, W_2 exactly.*

Proof. (1) By definition, RSP must have returned a path of total w_1 value at most W_1 and total w_2 at most $(1+\epsilon)W_2$. In addition, since the cost of the most expensive link in the graph where P was found was $c(e)$ the overall cost of the path was at most $nc(e)$. (2) RSP could not find a solution after link e was removed, and could find one before. Thus, e must be part of the solution, and thus, the total cost of the solution must be at least $c(e)$.

One can easily see that, by setting $LB = c(e)$ and $UB = nc(e)$ as above, we guarantee that a feasible solution (where, as described in the specification of the approximation algorithm, W_1 is satisfied and W_2 is not exceeded by more than an ϵ fraction) is known to exist for all costs above UB and an exactly feasible path does not exist with a total cost under LB. This initial step involves running at most m RSP executions, which are dominated by the other terms in the running time, giving the following running time.

Corollary 1. *ApproxMult has running time $O(mn^2/\epsilon^2)\log\log n$.*

[3] The proof of the following theorem can be found in the full version of this paper[16].

One can easily generalize this result to K parameters, by scaling down the costs and $K-1$ the parameters $w_2, \ldots, w_K$, obtaining a running time of $O(mn^K/\epsilon^K) \log\log n$.

4 Optimization Techniques for QoS Routing

Since the intrinsic difficulty of QoS routing is due to the constraints, we consider heuristics whose performance are independent of constraints in this section.

4.1 Optimization Techniques for Unicast QoS Routing

QoS routing problems are instances of integer programming, which are in general NP-complete to solve, with efficient solutions only under certain restrictions. Our goal thus is to transform unicast QoS routing into one or multiple integer programming problem/s which can be solved in polynomial time, while ensuring that the answer that we obtain is good enough for the original problem.

For efficiency, we will make sure that our integer programs are free of any "difficult" constraints, by using new cost functions which incorporate "penalties" for violating the (now omitted) constraints. The choice of the set of constraints to be relaxed into the objective function is made by the assumption that after relaxing them, the problem becomes easy. First there is a hidden constraint in the integer programming described in the above section, i.e. the solution for the problem must be a path. Therefore, instead of using e, our unicast problem can also be formalized as follows:

$$P: \qquad \min c(p) \quad s.t.\ w_i(p) \le W_i, \ \ i \in 1 \ldots K$$

where $c(p)$ is the total cost, and $w_i(p)$ is the sum of constraint i along path p.

To solve this problem, we give out two heuristics.

Algorithm LRA. We relax all the constraints and construct a new problem as follows:

$$P': \quad max_\lambda\ L(\lambda) = min\ c(p) + \textstyle\sum_{i=1}^{K} \lambda_i (w_i(p) - W_i)$$

Intuitively, λ_i determines how much we penalize the violation of ith constraint. A simple observation is that $L(\lambda)$ is a lower bound for P for $\lambda \ge 0$ since $\sum_{i=1}^{K} \lambda_i (w_i(p) - W_i) \le 0$. Clearly, finding a suitable λ to maximize the above expression will bring us closer to the optimal solution.

First, to ensure that the links we choose indeed form a path, we run Dijkstra's shortest path algorithm as the basic building block of our algorithm. Second, for quick convergence on an iterative search for the best λ, we need a good starting value. For this, we would like a quick upper bound on the cost, which, unfortunately, is usually as computationally difficult to solve as the original problem. As a solution, we use a two-phase approach.

In Phase 1 we inject a feasible path q into the input such that $c(q) = \infty$. (q is an upper bound for the solution.) The algorithm then efficiently improves q to use it as the initial state in Phase 2.

In Phase 2, the artificial path is removed, and the parameters are reset. Using the output of Phase 1 as the initial state, it is now possible to process the parameters in a fine-grained manner, to converge to a good solution quickly.

procedure LRA
Step 1 Find the min-cost path p_c using **Dijkstra**. **if** p_c is feasible, **return** p_c.
Step 2 **for** each parameter w_i find the minimum weight path p_{w_i} w.r.t. w_i using **Dijkstra**. **if** total weight for $p_{w_i} > W_i$, **return** "no solution".
Step 3 Find a feasible path as a starting point.
if failed **return** "no solution".
Step 4 Set the combined cost $c_\lambda(e) = c(e) + \sum_{i=1}^{K} \lambda_i^{k+1} w_i(e)$. Find the min cost path by **Dijkstra** using c_λ.
Adjust λ^k and **repeat** Step 4.

Step 3 corresponds to Phase 1. It starts off by establishing (for > 1 constraint) an artificial feasible path. This result is then refined by running a shorter instance of Phase 2 (Only step 4 is performed), resulting in a nested repetition of Phase 2 with different parameters. In Step 4 we adjust λ iteratively as $\lambda^{k+1} = \lambda^k + \theta^k(w(p) - W)$ where the step size is $\theta^k = \frac{L(\lambda^{k+1}) - L(\lambda^k)}{\|w_i(p^*) - W_i\|^2}$.[4]

Algorithm SRA. In the above algorithm we relax all the constraints into the object function. Even though the solutions are often quite satisfactory, the problem tends to become oversimplified. For example, given a reasonable amount of running time, LRA can not guarantee to return a solution. For applications which may require a high chance of finding a solution, a trade-off between low cost and feasibility is preferred. Motivated by the above, we develop a variant of LSA which relaxes all the constraints into one single constraint. Our reasoning is that for one-constraint QoS routing, finding a feasible path is easy, whereas for two or more constraints, this task is NP-complete. Therefore, reducing the number of constraints to one (and not to any higher value) makes the feasibility aspect of the problem easier to handle. In addition, research on one-constraint QoS routing is mature enough that we are given a choice of several approximation algorithms and heuristics. Our approach involves using a solution for one-constraint QoS routing as a building block in our algorithm to solve the multi-constraint case.

Again consider our problem formulation P as shown in the previous subsection. We can derive a new problem $P'(\lambda) : \min c(p)\ s.t. \sum_{i=1}^{K} \lambda_i(w_i(p) - W_i) \leq 0$ where $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_K)$ is a vector of norm 1 with all $\lambda_i > 0$. Intuitively λ represents the weight of different constraints. In contrast to LRA, we start with an initial (normalized) weight vector λ which gives equal weight to each constraint since we do not know how each parameter behaves. Instead of using Dijkstra, we use QOSONE(λ) as a building block which returns an optimal path p for problem $P'(\lambda)$, i.e. the algorithm that solves the one-constraint, min-cost problem. Notice that in practice QOSONE can not return a real optimal solution. In our simulation, we test different QOSONE, focusing on feasibility or optimality or both.

[4] The derivation of this parameter is in the full version of this paper[16].

procedure SRA
Step 1. $\lambda \leftarrow (\frac{1}{\sqrt{K}}, \frac{1}{\sqrt{K}}, \ldots, \frac{1}{\sqrt{K}})$
Step 2. Combine the constraints by λ, find a min cost path by **QOSONE**.
if $p ==$ NULL **return** "No Solution"
if p satisfies all the constraints **then return** p
Step 3. Adjust λ and **repeat** Step 2.

4.2 Optimization Techniques for Multicast QoS Routing

Algorithm LRATree. For multicast QoS routing, we simplify the integer program (which is similar to that for unicast except with K constraints/multicast node) and build a cost function which encompasses penalties for QoS constraint violations. In addition, we can think that each destination is an additional constraint to the integer problem and we also associate the penalties to these "constraints". Our algorithm requires that we solve a minimum-Steiner tree problem. Since the problem is NP-complete, instead we use a heuristic KMB [12]. Our multicast algorithm also uses the two phases approach as described in the unicast case. Here we point out a few notable differences[5]. To obtain an initial feasible tree, we use our unicast algorithm to find a feasible path to each destination and then combine these paths to obtain a tree. If this results in a cycle, we insert an artificial feasible tree into the graph, and then improve this tree to obtain a feasible tree that exists in the original graph. When updating λ, we take into account how many paths share a given link. This makes sure that, if a particularly "bad" link was being shared by many paths in the tree obtained in the previous iteration, the new tree will likely not include this link, or some of the branches which do not satisfy the constraint upper bounds will avoid using this link.

5 Observations and Analysis

Efficiency of the Heuristics. As seen in Figure 3, a constant number of iterations yields high accuracy, since the improvement reduces exponentially at each iteration. As a result, we show below that our algorithms scale well in terms of both the number of constraints and the network size.

Theorem 2. *Let K be the number of constraints. The running time of LRA is $O(K+Dijkstra)$. The running time of LRATree is $O(LK + KMB)$, where L is the number of multicast nodes. The running time of SRA is O(QOSONE).*

Quality of the Heuristics. We now argue about the quality of our algorithms. First notice that for unicast and multicast routing with a single constraint, LRA and LRATree are guaranteed to return a feasible path or tree if one exists.

We next try to obtain an intuition about why our technique is expected to work well. For the next lemma, assume that all costs and weights (we will call them delays) are uniformly chosen from the same range (if the ranges are different, they can be normalized). It shows that "better" paths will be favored.

[5] Pseudocode for LRATree can be found in the full version of the paper [16].

Lemma 3. *For single constraint unicast routing, let the minimum cost path p_0 have cost C_0 and delay d_0. Let p_1 and p_2 be two paths, with costs $C_1 < C_2$ and total delay $d_1 < d_0$ and $d_2 < d_0$. Assume that $d_0 > W$, thus, the minimum cost path is not feasible. Let λ be the step size. Then, for $\lambda = 1$, if one of p_1 and p_2 is returned at the next step (if both are feasible), the probability that p_1 will be returned rather than p_2 is at least 75%. In addition, there exists a $\lambda > 0$ which will cause p_1 to be picked over p_2 with probability 1.*

Proof. After adjusting λ, the new costs for p_1 and p_2 will be $C_1' = C_1 + \lambda d_1$ and $C_2' = C_2 + \lambda d_2$ respectively. Since the delays are uniformly chosen, with probability 0.5, $d_2 > d_1$, which trivially satisfies $C_1' < C_2'$. If $d_2 \leq d_1$ (with probability 0.5), however, we must have $C_1 + \lambda d_1 < C_2' = C_2 + \lambda d_2$, i.e., we need $\frac{C_2 - C_1}{d_1 - d_2} > 1$. Since the delays and the costs were uniformly picked from the same range, by symmetry the likelihood of this is at least 0.5. Thus, the probability that $C_1' < C_2'$ is at most $0.5 + 0.25 = 0.75$. Note that $C_1' < C_2'$ can always be satisfied if $\lambda < \frac{C_2 - C_1}{d_1 - d_2}$.

This simplified analysis seems to support small λ. However, note that $C_0 < C_1 < C_2$, and $d_0 > W$. Thus, if $\lambda < \frac{C_1 - C_0}{d_0 - d_1}$, then under the new cost function, p_0 will have cost C_0', less than C_1', thus, p_0 will seem better than both p_2 and p_1, even though it is not feasible. Also note that the likelihood of this scenario depends on $C_0 - C_1$, thus the need to adjust λ depending on how close to the solution we are. This analysis generalizes to any number of paths. For example, if there are 3 paths with $C_0 < C_1 < C_2 < C_3$ one can show that the probability p_1, p_2, and p_3 will be preferred to the other paths after the adjustment is $\frac{11}{18}, \frac{5}{18}, \frac{2}{18}$ respectively. This tells us that the likelihood that one of the least costly paths will be chosen is high. Our simulations reaffirm this property.

Theorem 3. *If QOSONE returns an exact solution to P', every time a feasible solution is found by SRA, it must also be optimal.*

Proof. Assume a feasible path p is found and p is not optimal. Let *opt* denote the optimal path. Then $c(opt) \leq c(p)$, p and *opt* satisfy all the upper bounds. By design, our algorithm will next find a positive λ. Since *opt* is feasible for P'', i.e. $\forall i, w_i(opt) - W_i \leq 0$, summing up, we have $\sum_{i=1}^{K} \lambda_i (w_i(opt) - W_i) \leq 0$. Thus *opt* is also a feasible path for the one-constraint QoS routing, $P'(\lambda)$. Since p is the optimal path for $P'(\lambda)$, then $c(p) \leq c(opt)$. Thus $p = opt$.

Corollary 2. *If the output of SRA is a feasible path, the corresponding path from QOSONE is a non-optimal feasible path.*

The above theorem and corollary clearly show that our algorithm can control the performance of QoS routing by the choice of QOSONE. If QOSONE is more likely to find an optimal path, SRA is more likely to find an optimal solution. This gives us a chance of improving multi-constraint QoS routing by focusing on improving single-constraint QoS routing. On the other hand, if QOSONE is focusing on finding a feasible path, the probability of finding a feasible path in a certain amount of time for multi- constraint problem is also improved.

Multicast. For multi-constraint multicast routing, the number of the destinations has a significant effect on the accuracy. This is because early in the algorithm we need to find feasible paths for each destination. As the number of destinations increases, the likelihood that LRA will succeed on all of these destinations drops exponentially, thus increasing the probability that LRATree will fail. Instead we notice from the above analysis that SRA can obtain high feasibility by choosing QOSONE as an algorithm focusing on feasibility. Therefore in the first phase of our algorithm where the goal is to find a feasible tree as a starting point for further minimization, we use SRA in our experiments.

6 Simulation Results

Simulation Environment. To test our technique, extensive simulations have been carried out. We used two different methods to generate the network topology. First we used *ANSNET* [3], shown in Figure 1. The cost of each link was set uniformly in the range [1, 1000]. The delays (constraints) of each link were set uniformly in the range [0, 100]. The delay upper bounds were set to a random number uniformly in range [100, 200], [100, 300], ... [100, 500] for different simulations. Next we used the widely adopted Waxman Model [17] to generate random networks. The network consisted of 40 to 90 nodes where the two parameters α and β were set to 0.3 and 0.2 respectively, and the grid was 30×30. The cost on each link was generated by the model itself as the Euclidean distance. The delays and delay upper bounds on each link were chosen the same as *ANSNET* model. Each data point obtained in our figures represents the average of 150 runs on different random networks.

For SRA we chose different algorithms for single constraint problem as a building block for our algorithm. The sample building blocks represent the tradeoff between accuracy and efficiency, and are as follows. 1) SRA-Dijkstra: we find the shortest path based on the single combined constraint. 2) SRA-LARAC: a relaxation heuristic algorithm for one constraint unicast QoS routing [9]. 3) SRA-Exact: we use dynamic programming to obtain an exact solution to one constraint unicast QoS routing. Here, Dijkstra and Exact are two extremes in that Dijkstra has no claim to optimality, but is efficient, whereas Exact always returns the optimal solution for QOSONE, but runs in super-polynomial time. We use Exact just to show how our algorithm perform. In practice, people can use approximation algorithm, e.g. [4] to achieve similar result instead or use LARAC which is an in-between algorithm.

We compare our algorithm with the optimal solution both for unicast and multicast routing. However, obtaining an optimal solution for multivariate multicast routing in a large network has proven to be an unconquerable task, as mentioned in [15]. For a 10-node graph, the computation for every single point in our figures took many hours. To overcome this problem, for large multicast problems we insert a random feasible tree into the network and test our algorithm based on the modified network.

Interpreting the Results. The outcome of the algorithm will fall within one of the following cases. (S) Optimal answer found, or no feasible solution exists. (F1) Feasible but not optimal answer found. (F2) Feasible answer exists, but not found. Note that by design the algorithm never returns a non-feasible path. We evaluate our algorithms with respect to the following criteria: (i)*Full success* = S/(S+F1+F2), the ratio of fully correct answers. (ii) *Partial success* = (S+F1)/(S+F1+F2), the ratio of fully correct and feasible answers. (iii) *Excess*, the percentage excess (over optimal) cost of a returned solution.

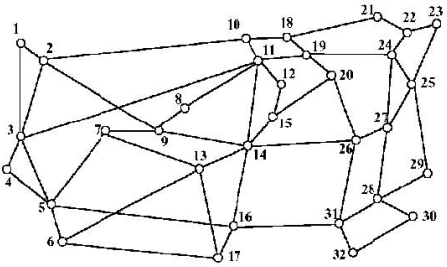

Fig. 1. ANSNET Network.

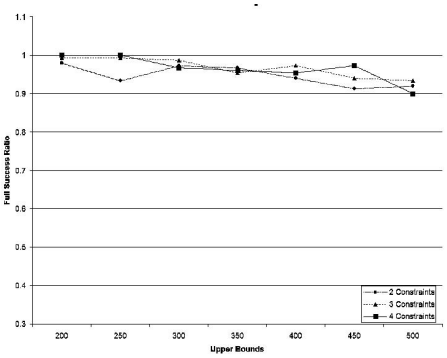

Fig. 2. ANSNET model.

6.1 Unicast Routing Simulations

LRA. In our simulations, we observed that partial or full success rates were very much independent of the number of parameters and above (usually, well above) 92% and 90% respectively. Increasing the range for the constraints led to a small (around 2-5%) drop in the full success rate, which we attribute to the increase in the number of feasible paths in the network. This can be seen in Figure 2. Figure 3 shows that increasing the number of the iterations yielded diminishing returns due to the exponentially decreasing step size with no gain beyond 16 iterations. We also explored the effects of the correlation (Figure 4) between the QoS parameters with two (as well as four, grouped into two) random, positively correlated (with a small random difference between the values), and negatively correlated (two parameters adding up to a constant plus a small random number). Even though our partial success rates was high throughout, correlation affected the total success rate (over 97%, 96%, 90% for positive, random, negative correlation). Finally, in Figure 5, we observed that the percentage value of the excess as a measure of the effectiveness of our partial successes compared to the full successes, and found average excess rates are extremely low.

SRA. We then investigated the effects of using different one-constraint QoS routing algorithms. In Figure 6, we observe the *full success ratio*. This indicates how well the algorithm performs in achieving optimality. As expected, SRA-Exact outperformed all other QOSONEs in terms of full success rates, followed

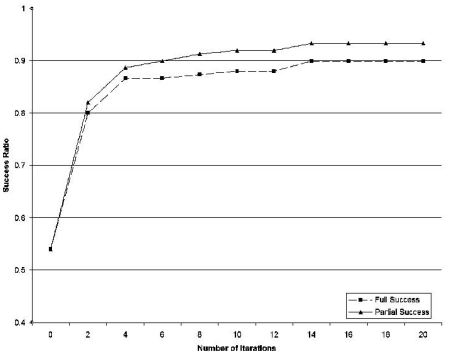

Fig. 3. Waxman model, 90 Nodes, 3 constraints.

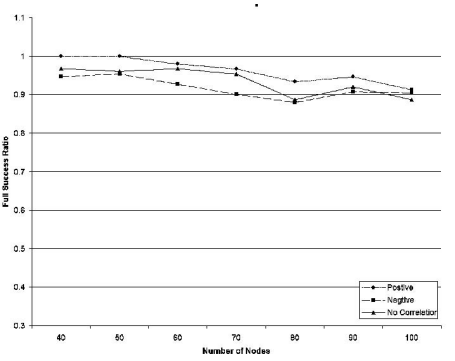

Fig. 4. Waxman model, 2 constraints.

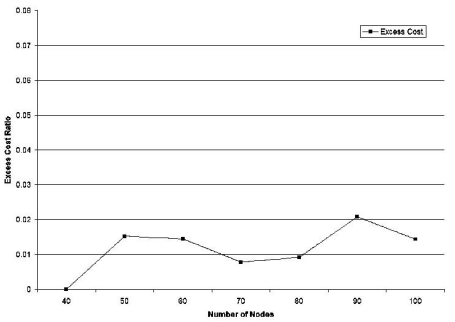

Fig. 5. Waxman model, 3 constraints.

by SRA-LARAC. LRA (Lagrange) was a very close third, and SRA-Dijkstra performed the worst. With the partial success rate (Figure 7), SRA-Dijkstra was the best with a near-perfect performance in finding a feasible path. SRA-LARAC was the second and LRA the third. As we expected, SRA-Exact performed the worst. We can conclude that SRA maintains the properties of the one-constraint algorithm; therefore one can choose different one constraint unicast algorithm based on their requirements.

6.2 Multicast Routing Simulations

Our experiments for multivariate multicast routing were similar to those with unicast routing. Due to the prohibitive running time of finding the optimal solution we tested our algorithms on small graphs. The number of iterations was set to be 8 for the main program, and 8 for finding an upper bound.

In general, while the running times remained fast, the accuracy dropped when we wanted to satisfy all of the constraints for all of the nodes. Part of this was due to cases where the KMB algorithm failed. In addition, as argued before, with more destinations a fully feasible multicast tree is difficult to find. We thus used SRA in some experiments. Test results are given in Figures 8-11.

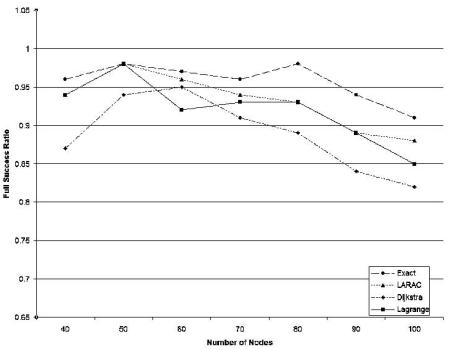

Fig. 6. Upper bounds [100, 400], 2 constraints, full success ratio.

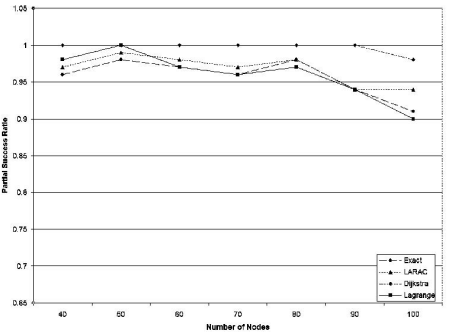

Fig. 7. Upper bounds [100, 400], 2 constraints, partial success ratio.

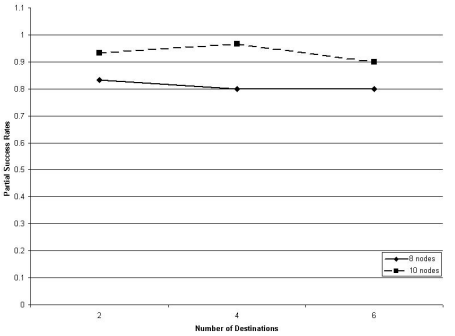

Fig. 8. Waxman, 2 constraints.

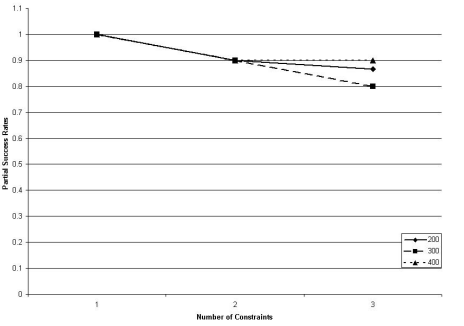

Fig. 9. Waxman, 10 nodes 6 destinations.

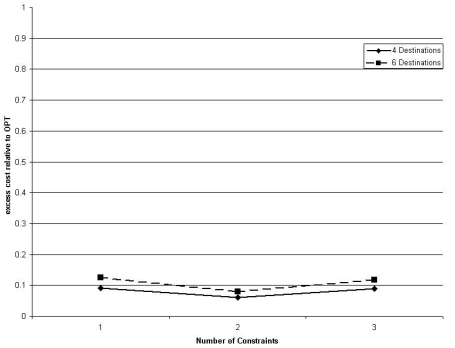

Fig. 10. Waxman model, 10 nodes.

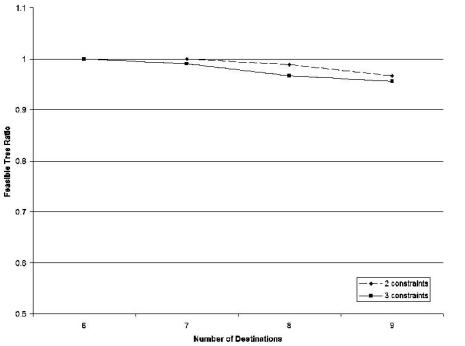

Fig. 11. ANSNET, SRA with Dijkstra.

Even though our success rates were lower than those for unicast, our results remained desirable: one must consider the enormous difficulty of solving this problem exactly, our extremely low running times, as well as the absence of known efficient algorithms for this problem. The figures show that network size, the number of multicast nodes, and the number of constrains all lead to a fairly small drop in accuracy. The excess cost is upper bounded by 17%, higher for increased number of multicast nodes.

We also tested our algorithm on *ANSNET*. Since we could not obtain the optimal solution, we focused on feasibility. To maximize feasibility we chose SRA for each destination and for SRA we chose Dijkstra for QOSONE which had shown a nearly perfect feasibility ratio previously, i.e. 100% ratio for up to 90 nodes. We see that in Figure 11, we had a high success to reach a feasible tree.

References

1. R. Ahuja, T. Magnanti, J. Orlin *Network flows: theory, algorithms, and applications* NJ Prentice Hall, 1993.
2. S. Chen and K. Nahrstedt, *An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions*, IEEE Network, pp. 64-79, November/December, 1998.
3. S. Chen and K. Nahrstedt, *On Finding Multi-Constrained paths*, IEEE ICC, 1998.
4. F. Ergun, R. Sinha and L. Zhang *An Improved FPTAS for Restricted Shortest Path* Information Processing Letters 83(5): 287-291, 2002.
5. M. Garey and D. Johnson, *Computers and Intractability: Guide to the Theory of NP-Completeness*, W. H. Rreeman, New York, 1979.
6. R. Guerin and A. Orda, *QoS-based Routing in Networks with Inaccurate Information: Theory and Algorithms*, IEEE INFOCOM, 1997.
7. R. Hassin *Approximation scehems forthe restricted shortest path problems* Math. Op. Res. 17(1):36-42,1992.
8. F. Hwang and D. Richards, *Steiner Tree Problem*, Networks, vol. 22. no. 1, pp. 55-89, January 1992.
9. A.Juttner, B. Szviatovszki, I. Mecs and Z. Rajko, *Lagrange Relaxation Based Method for the QoS Routing Problem*, IEEE INFOCOM, 2001.
10. R. Karp, *Reducibility among Combinatorial Problems*, in Complexity of Computer Computatioins (R. Miller and J. Thatcher, eds.), pp. 85-103, Plem Press, 1972.
11. T. Korkmaz, M. Krunz and S. Tragoudas *Multiconstrained Optimal Path Selection* IEEE INFOCOM, 2001.
12. L. Kou, G. Markowsky and L. Berman, *A Fast Algorithm for Steiner Trees*, Acta Infomatica, vol. 15, no. 2, pp. 141-145, 1981.
13. M. Parsa, Q. Zhu and J. J. Garcia-Luna-Aceves, *An Iterative Algorithm for Delay-Constrained Minimum-Cost Multicasting* IEEE/ACM Transactions on Networking, vol. 6, No. 4, August 1998.
14. D. Raz and D. Lorenz *Simple Efficient Approximation Scheme for the Restricted Shortest Path Problem* Operational Research Letters (ORL), 28(5), pp. 213-219, June 2001.
15. H. F. Salama, D. S. Reeves, Y. Viniotis, *Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks*, IEEE Journal on Selected Areas in Communication, vol 15, no. 3, pp 332-345, April 1997.
16. http://vorlon.cwru.edu/∼dxw49
17. B. M. Waxman, *Routing of multipoint connections*, IEEE Journal on Selected Areas in Cummunications, 6(9): 1617-1622, 1988