Optimal QoS Routing Based on Extended Simulated Annealing*

Yong Cui, Ke Xu, Mingwei Xu, and Jianping Wu

Department of Computer Science, Tsinghua University
Beijing, P.R.China, 100084
{cy,xuke,xmw}@csnet1.cs.tsinghua.edu.cn, jianping@cernet.edu.cn

Abstract. Quality-of-service routing (QoSR) is to find an optimal path that satisfies multiple constraints simultaneously. As an NPC problem, it is a challenge for the next-generation networks. In this paper, we propose a novel heuristic SA_MCOP to the general multi-constrained optimal path problem by extending simulated annealing into Dijkstra's algorithm. The heuristic first translates multiple QoS weights into a single metric and then seeks to find a feasible path by simulated annealing. Once a feasible path is found, it optimizes the cost without losing the feasibility. Extensive simulations demonstrate that SA_MCOP has the following advantages: (1) High performance in both success ratio and cost optimization. (2) High scalability regarding both network size and the number (k) of QoS constraints. (3) Insensitivity to the distribution of QoS constraints.

1 Introduction

Providing different quality-of-service (QoS) support for different applications in the Internet is a challenging issue [1], in which QoS Routing (QoSR) is one of the most pivotal problems [2] [3]. The main function of QoSR is to find an optimal path that satisfies multiple constraints for QoS applications. For the NP-completeness of QoSR problem [4] [5], many heuristics have been proposed. However, these algorithms have some or all of the following limitations [2]: (1) Most of the heuristics only focus on a branch of the QoSR problem. (2) High computation complexity prevents their practical applications; (3) Low performance sometimes leads to fail to find a feasible path even when one does exist. (4) Some algorithms only work for a specific network.

Simulated annealing is a meta-heuristic method for combinational optimization [6]. Based on an initial solution, it repeatedly iterates to a new solution. For the Multi-Constrained Optimal Path (MCOP) problem, we extend simulated annealing to take an end-to-end path as a solution in routing computation, and propose a novel heuristic, SA_MCOP (Simulated Annealing for MCOP), by extending simulated annealing [6] to Dijkstra's algorithm.

* Supported by: (1) the National Natural Science Foundation of China (No. 90104002; No. 69725003); (2) the National High Technology Research and Development Plan of China (No. 2002AA103067).

H.-K. Kahng (Ed.): ICOIN 2003, LNCS 2662, pp. 553–563, 2003.

When a QoS connection request arrives at a router, the router uses our SA_MCOP to compute an optimal feasible end-to-end path (or the next hop) based on the network state information it maintains. This algorithm uses a nonlinear energy function to translate multiple QoS constraints into a single metric. It first computes a complete shortest path tree (SPT) with respect to the traditional cost as the initial solution for simulated annealing by Dijkstra's algorithm. If the path along the current SPT is not feasible, it marks all of the nodes in the graph according to the current SPT. Then a new SPT is created in simulated annealing mode by our improved Dijkstra's algorithm with a nonzero probability $P(T)$ to select a non-optimal path, where T is the temperature for simulated annealing. If the path along the new SPT is not feasible yet, the algorithm then marks each node again based on the current SPT and computes a new SPT with a lower temperature T iteratively. When T decreases to $T \to 0$, we have $\lim_{T \to 0} P(T) = 0$. If a path is feasible, the algorithm optimizes the cost at last. Based on the theory about simulated annealing, SA_MCOP guarantees to find a feasible path when one exists. Extensive simulations also show that SA_MCOP performs well.

The rest of this paper is organized as follows. In Sect. 2 we analyze how to translate multiple weights to a single metric and summarize the simulated annealing. SA_MCOP is proposed in Sect. 3, and extensive simulations show the performance evaluation in Sect. 4. Finally, conclusions appear in Sect. 5.

2 Background information

2.1 Problem Formulation

A network is represented by a directed graph $G(V, E)$. V is the node set and an element $v \in V$ is called a node representing a router. E is the set of edges representing links, which connect the routers. An element $e_{ij} \in E$ represents an edge $e = v_i \to v_j$ of G. In QoSR, each link has a group of independent weights $(w_0(e), w_1(e), \cdots, w_{k-1}(e))$, which is also called QoS metric . For a path $p = v_0 \to v_1 \to \cdots \to v_n$ and $0 \leq l < k$, the weight $w_l \in R^+$ satisfies the additive property if $w_l(p) = \sum_{i=1}^{n} w_l(v_{i-1} \to v_i)$.

Definition 1. *Multi-Constrained Optimal Path (MCOP): For a given graph $G(V, E)$ with source node s, destination node t, and a constraint vector $c = (c_0, c_1, \cdots, c_{k-1})$, when $k \geq 2$, the path p from s to t is called a multi-constrained optimal path, if (1) $w_l(p) \leq c_l$ for each $0 \leq l \leq k-1$ (we write $w(p) \leq c$ in brief), and (2) $cost(p) \leq cost(p')$ for any p' satisfying $w(p') \leq c$.*

Note that both $w(e)$ and c are k-dimensional vectors. For a given QoS request and its constraint c, QoSR seeks to find the path p with optimal cost based on the network state information, where p satisfies $w(p) \leq c$. Dijkstra proposed the shortest path tree (SPT) algorithm, which has a low computation complexity [8]. However, QoSR problem is related to multiple weights simultaneously. Thus the

problem is changed to an NPC one that the original Dijkstra's algorithm cannot solve in polynomial time. Therefore, one feasible method is to translate the multiple weights into a single metric, as follows:

Definition 2. *Energy Function:* $g(p) = \max_{l=0}^{k-1}\{w_l(p)/c_l\}$ *is called the energy function of path p, where* $c = (c_0, c_1, \cdots, c_{k-1})$ *is the constraint vector of a QoS application.*

2.2 Simulated Annealing

Research on statistical mechanics shows that in temperature T, the probability for a molecule of substance to stay in the state r satisfies Boltzmann's distribution:

$$Pr\{\hat{E} = E(r)\} = \frac{1}{Z(T)} exp(-\frac{E(r)}{k_B T}) \tag{1}$$

$\hat{E}$ is the stochastic variable representing the energy of a molecule. $E(r)$ is the energy of a molecule that stays in the state r. T is the temperature. k_B is Boltzmann's constant and $Z(T)$ is the normalized factor.

Annealing is a physical process. After a metal body is heated, when it cools down slowly, the molecules of the body stay in different states with different probabilities, which satisfy Boltzmann's distribution. Annealing usually requires the following two conditions:

(1) The initial temperature is high enough so that the probabilities for a molecule to stay in arbitrary states are approximately equal. If we have

$$T_0 \gg E(r)/k_B \tag{2}$$

then $E(r)/k_B T_0 \approx 0$. As a result, $Pr\{\hat{E} = E(r)\} \approx 1/Z(T_0)$, i.e. the probabilities are approximately equal.

(2) When it cools down to $T = 0$, all of the molecules will stay in the least-energy state with the probability being one. If r^* presents the least energy state, when $T \to 0$, we have

$$Pr\{\hat{E} = E(r)\} = \begin{cases} 1, & r = r^*; \\ 0, & others; \end{cases} \tag{3}$$

The idea of simulated annealing was first proposed by Metropolis [9] and was applied to combinational optimization successfully in 1983 [6]. In its basic form, it first generates an initial solution as the current solution. It then selects another solution in the neighborhood of the current solution and replaces the current solution with the new one. The same process continues iteratively for many times. Although the goal is to find an optimal solution, it selects a non-optimal solution with a non-zero probability $P(T)$ to avoid being stuck in a local optimization. When the temperature decreases, $P(T)$ also decreases. When $T \to 0$, it is guaranteed to find an optimal solution since the probability $P(T)$ is zero to select a non-optimal solution.

3 MCOP Based on Simulated Annealing

3.1 The Idea of SA_MCOP

The key issues to use metaheuristics in QoSR include (1) how to express a solution, and (2) how to iterate. We extend simulated annealing to take an end-to-end path as a solution and use Dijkstra's algorithm to guarantee that a new solution is still an end-to-end path in iterations. When we compute the SPT by Dijkstra's algorithm, we select with probability $P(T)$ a node that is not the current optimal node. Therefore, our SA_MCOP can overcome the local optimization problem that all heuristics face.

For a given QoS request from s to t, node s first uses Dijkstra's algorithm to calculate the least-cost SPT rooted by s and marks each node in the network. Then it uses an improved Dijkstra's algorithm to compute new labels for each node iteratively based on the old labels computed last time. With different probabilities $P(T)$, it selects different links including non-optimal links, where $\lim_{T\to 0} P(T)$ satisfies Eq. 3. After each iteration, the temperature T decreases. When the algorithm iterates enough times, we guarantee $T \to 0$. In order to optimize the cost, when multiple feasible paths are found, the heuristic will choose the path that has the least cost.

3.2 Pseudo-code Description

Fig. 1 shows the pseudo-code of the algorithm, where SA_MCOP is the main function. The input of SA_MCOP includes a given graph with multiple QoS weights, a QoS request from s to t and a constraint vector $c = (c_0, c_1, \cdots, c_{k-1})$. In addition, we can configure the initial temperature (T_0), the gradient ($grad$) for cooling down the temperature and the iteration times (I). If the k-dimensional weight $d[t]$ of the forward least energy path from s to t satisfies the constraint c, the algorithm returns the path successfully. Otherwise, it refuses the request. Table 1 shows the notations used in the pseudo-code.

1. **Function SA_MCOP** In function SA_MCOP, we first use Dijkstra's algorithm to compute the least-cost SPT rooted by s (Line 2), where the initial solution is the path along the SPT from s to t. We then compute the new SPT by simulated annealing (SA_Dijkstra) backwards and forwards iteratively, including (1) computing the SPT rooted by t (Line 5); (2) computing the SPT rooted by s (Line 8). After the complete SPT is computed each time by Dijkstra's algorithm or SA_Dijkstra, $d[.]$ is updated to save the newly computed weights from each node to the root of the new SPT. On the other hand, SA_Dijkstra computes a new SPT based on the $d[.]$ updated last time (Line 1 in function SA_Relax). In addition, after a new SPT is constructed, the path along this SPT from s to t is checked to see whether it satisfies constraint c (Line 3, 6 and 9). Once it does (i.e. a feasible path is found), the algorithm seeks a least-cost path by OPT_Dijkstra. If it is not feasible, we then change the temperature T for simulated annealing to construct a new SPT by SA_Dijkstra iteratively (Line 5, 7, 8 and 10).

OPT_Cheapest()
1. $ret = INFINITY$;
2. $minCost = INFINITY$;
3. FOR each node v in NB
4. IF $r[v] + d[v] < c$ //feasibility
5. IF $minCost > cost[v]$ //optimize cost
6. $ret = v$
7. IF $ret = INFINITY$ RETURN no node
8. ELSE RETURN ret

OPT_Relax(u, v)
1. IF $r[u] + w(u, v) + d[v] < c$
2. IF $cost[u] + cost(u, v) < cost[v]$
3. relax v to u's child

SA_AddNode(u)
1. $NB = NB - u$ // remove u from NB
2. $SPT = SPT + u$ // add u to SPT
3. FOR each node v in u's neighbor
4. IF v is not in SPT
5. $NB = NB + v$ // add u's neighbor

SA_Relax(u, v)
1. $tmp = max_{l=0}^{k-1}(r_l[v] + w_l(u, v) + d_l[v])/c_l$
2. IF $g[v] > tmp$ // relax v to be u's child
3. $g[v] = tmp$
4. $r[v] = r[u] + w(u, v)$
5. $cost[v] = cost[u] + cost(u, v)$
6. $Pr[v] = u$

Fig. 1. Pseudo-code of the proposed heuristic

2. **Function SA_Cheapest** This function presents the idea of simulated annealing: a non-optimal node will be selected with a certain probability and the probability decreases to zero when temperature T decreases enough. In the first line of SA_Cheapest, $\max_{l=0}^{k-1}(r_l[v] + d_l[v])/c_l$ is the energy of a path defined by Def. 1. $r_l[v]$ is the forward weight, i.e. the l'^{th} weight of the path from the root of the current SPT to node v. $d_l[v]$ is the backward weight, i.e. the l'^{th} weight of the path from node v to the root of the old SPT calculated last time. The backward weight is saved when the old SPT is computed last time (Line 12-13 in function SA_Dijkstra). SA_Cheapest first selects the least energy g^* of the neighbors of the current SPT (Line 1). It then computes the energy $E(v)$ for simulated annealing (Line 2-3), which guarantees the least energy to be 0. Then the normal factor $Z(T)$ in Eq. 1 is calculated (Line 4). Finally, a node u, which will be added to the partially created SPT, is selected according to the probability distribution in Eq. 1 (Line 5-9).
3. **Function SA_AddNode**

Table 1. Notations in the pseudo-code of SA_MCOP

Symbol	Meanings	Symbol	Meanings
T_0	initial temperature	I	maximum number of iterations
E(v)	the energy of node v in formula (i)	$Pr[v]$	the precedent node of node v
grad	gradient for decreasing temperature	c	k-dimensional constraints of a QoS request
OPT_Dijkstra (G,s)	proposed heuristic finding the least-cost paths rooted by s	$d[u]$	backward weights of the path along the old SPT from its root to u
Dijkstra (G,s)	standard Dijkstra's algorithm for SPT rooted by s	NB	the set of the neighbors of the current SPT
u	an intermediate node	$g[u]$	the energy of node u
SA_Dijkstra (G,s,T)	proposed heuristic for SPT rooted by s based on simulated annealing	$r[u]$	forward weights of the path along the current SPT from its root to u
Z	the normal factor in formula (i)	v	a child node of node u
$g*$	a locally minimal energy	SPT	a partially created SPT

Similar to the original Dijkstra's algorithm, when node u is added to the partially created SPT, we use this function to change the set NB, which is the neighborhood of node u. This includes two parts: deleting node u from NB (Line 1), and adding u's neighbors that are not in the current SPT to NB (Line 3-5).

4. **Function SA_Relax** We relax node v via v's parent u. The energy of v via u is computed (Line 1). If this new energy of v is smaller than the old one (Line 2), node v will be relaxed to use the new energy (Line 3), the forward weight (Line 4), the cost (Line 5) and the precedent node (Line 6).
5. **Function OPT_Cheapest**
   OPT_Relax In order to guarantee the feasibility of newly found optimal path, when OPT_Dijkstra chooses the least-cost node and relaxes node, it has to check the feasibility of the new node. Only when the new potential paths satisfy the constraint, the node on such paths can be chose or relaxed. In OPT_Cheapest, when the feasibility is guaranteed (Line 4), least-cost node is added to the optimal SPT (Line 5-6). It should be note that, the optimal SPT may not be a complete one that connects all of the nodes in the graph. Instead, we only consider feasible ones marked by above SA_Dijkstra or Dijkstra (Line 7). If there is no more feasible node, OPT_Dijkstra returns the current optimal path as the least-cost one.

3.3 Complexity and Parameters

Since the computation complexity of an improved Dijkstra's algorithm only considering the cost is $O(m + nlogn)$, the complexity of SA_Dijkstra and OPT_Dijkstra is $O(km + knlogn)$, respectively. As a result, including the iteration, the overall computation complexity of SA_MCOP is $O(Ik(m + nlogn))$, where I is the maximum number of iterations. Because the feasibility of a path newly found is checked before the next iteration, when most of the QoS requests are feasible, only some of them need to iterate for multiple times. Therefore,

the above complexity is the worst-case one. In fact, the running time of our SA_MCOP is almost independent of the maximum number of iterations.

Simulated annealing requires a new solution selected randomly enough at the beginning, i.e. initial temperature T_0 should be high enough. Because the energy $E(v)$ is often much less than one in SA_MCOP, it suffices to set $T_0 = 1$ to satisfy Eq. 2 . In addition, simulated annealing also requires that when the temperature $T \to 0$, all of the molecules stay in the state with the least energy. Thus, in order to decrease the temperature quickly, we select $grad = 10$ according to the geometric proportion. In this way, after $2I$ times of iteration, the temperature $T(2I) = 10^{-2I} \ll E(v)$ satisfies Eq. 3. The following simulations show that such parameters perform well.

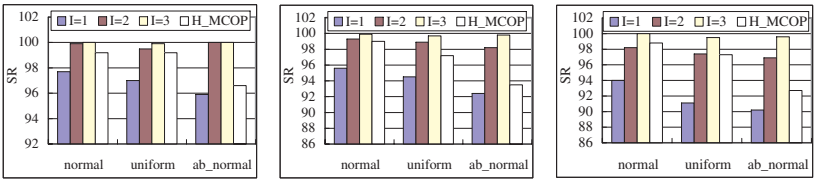
4 Performance Evaluation

We simulate purely random network graphs with N nodes [10] and generate k weights for each link, where $w_l(e) \in uniform[1, 1000]$ for $l = 0, 1, \cdots, k-1$ and $w_l(e)$ has no correlation for different e or l. We simulate 10 graphs with N being 50, 100 and 200, respectively. In each graph, we select the source-destination node pair (s, t) 100 times (a particular node may be selected more than once), where we guarantee that the minimum hop is not less than three. Each source node s uses SA_MCOP to compute the least energy path for different numbers of iterations respectively. For performance evaluation, we use the success ratio (SR), which is defined as the ratio of the number of requests satisfied using a heuristic algorithm and the total number of requests generated. We first get SR of the 100 (s, t) pairs in one graph, and then calculate the average SR of 10 graphs with same number of nodes.

4.1 The Performance with Two Constraints

The evaluation depends heavily on the generated constraints of the requests, e.g. the distribution of constraints. Therefore, based on the normalized weights in the whole graph, for a given request pair (s, t), we use the method of weighted ratio simulation to generate the constraints. First, we assume that each QoS application concerns the weight w_l to a_l degree. Then we use Dijkstra's algorithm to find the path $p(s, t)$ that minimizes the linear energy $\sum_{l=0}^{k-1} a_l w_l(s, t)$. Finally, we take the weights of the path $p(s, t)$ as the QoS constraints of the pair , i.e. $c(s, t) = w(p(s, t))$.

In the case of two dimensions, we let $a_1 \in [0, 1]$ and $a_0 = 1 - a_1$ for simplicity. Because different QoS applications concern a weight to different degrees, we use the following three methods to generate a. (1) NORMAL: $a \in normal(0.5, 0.16)$; (2) UNIFORM: $a \in uniform(0, 1)$; (3) AB_NORMAL: $a \in normal(0, 0.16)$ and $a \in [0, 0.5]$. In order to guarantee that the difference between a_1 and the expectation are less than 0.5 with the probability of 99.7%, we set the standard deviation to be 0.16 in NORMAL and AB_NORMAL distributions.



(a) $N = 50$ (b) $N = 100$ (c) $N = 200$

Fig. 2. Performance evaluation with two constraints

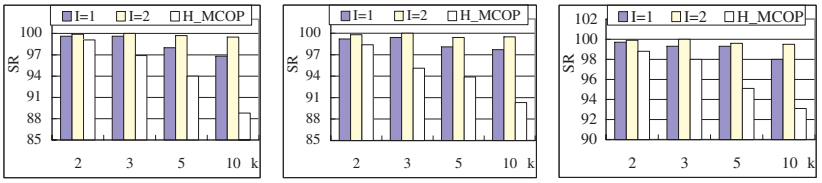
The relation between the maximum number of iterations and the performance of our SA_MCOP is shown in Fig. 2 against H_MCOP. The x-axis is the method to generate QoS constraints, and the y-axis is the success ratio (SR) representing the performance of heuristic routing. With only a few iterations (e.g. $I = 1$), SA_MCOP does not perform well. The main reason is that $T_0 = 1$ is much greater than energy $E(v)$ and the strong randomicity cannot guarantee an optimal path. With more iteration times, the performance of SA_MCOP increases rapidly and reaches almost 100%. This shows that the simulated annealing can increase the performance of QoSR greatly.

H_MCOP has different performance with different QoS constraints. The reason is that when it computes the SPT for the first time, it concerns the two weights to the same degree. Therefore, when applications concern the two weights to the same degree (normal distribution in Fig. 2), H_MCOP performs well; otherwise, it will degrade, especially in the ab_normal distribution. On the contrary, our SA_MCOP performs well in all conditions, including different distributions of QoS constraints and different network scales.

4.2 Performance with Multiple Constraints

In order to study the relation between the maximum number of iterations and the performance of SA_MCOP, we use the following method to generate the constraints for a given (s, t) pair. We first take the random number $b_l \in uniform(0, 1)$ for $l = 0, 1, \cdots, k$ and calculate $a_l = b_l / \sum_{l=0}^{k-1} b_l$. We then construct the least energy path from s to t according to the energy function $g_1'(p) = \sum_{l=0}^{k-1} (a_l w_l(p))$ and take the weights of the path as the constraints of the given (s, t), i.e. $c(s, t) = w(p_i)$.

Fig. 3 shows the performance for multiple constraints, where the x-axis is the number of constraints and the y-axis is SR. SA_MCOP performs well for large k, while H_MCOP does not. Furthermore, our SA_MCOP has good scalability on the size of network with multiple constraints.



(a) $N = 50$ (b) $N = 100$ (c) $N = 200$

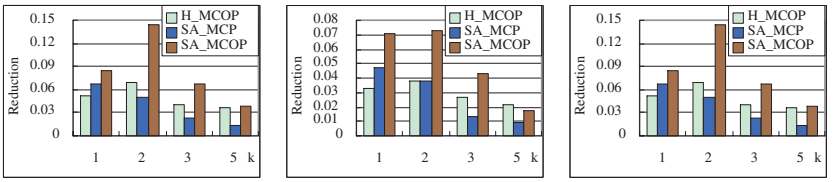
Fig. 3. Performance evaluation with multiple constraints

4.3 Optimization of Traditional Cost

The rigorous constraints that we generate for a given (s, t) pair in the above experiments restrict the number of feasible paths, even to only a single one. In order to represent the performance of optimization, we use Dijkstra's algorithm to find the shortest path p_l w.r.t. w_l for each $l = 0, 1, \cdots, k-1$. Then we take random $c_{l+1} \in uniform(0.8w_{l+1}(p_l), 1.2w_{l+1}(p_l))$ as one element of the constraint vector c.

We compare four heuristics in this section: H_MCP, H_MCOP [7], SA_MCP and SA_MCOP. H_MCP is a variation of H_MCOP by removing the optimization parts from H_MCOP [7]. SA_MCP is a variation of our SA_MCOP without optimization (i.e. in line 3, 6 and 9 SA_MCOP just returns the feasible path along the current SPT rather than compute optimal path by OPT_Dijkstra). Furthermore, SA_MCP computes the shortest path w.r.t. the hop number as the initial solution instead of the one w.r.t. cost in SA_MCOP. SR relies on the number of iteration in SA_MCP and SA_MCOP. Because the comparison of optimization is meaningful with same or close SR, we adjust iteration times to keep a close SR. In our experiments, most iteration times are one or two.

Fig. 4 shows the average performance of cost optimization. The y-axis represents the percentage of the cost reduction by other heuristics compared with H_MCP. This figure demonstrates the four points: (1) SA_MCOP performs better than H_MCOP for more reduction of cost. (2) SA_MCOP is much better than SA_MCP, so the optimization parts in SA_MCOP algorithm are necessary and efficient. (3) When $k = 1$, the method generating constraint $c_0 \in uniform(0.8w_0(p_0), 1.2w_0(p_0))$ is so strict that there are not many feasible paths as candidates. Therefore, for the limitation of the generation method, this figure does not show a high reduction with $k = 1$. If we omit the part with $k = 1$, we will find that when k increases (i.e. more weights and constraints), for the short of feasible paths, the reduction decreases. (4) The larger the network, the more the reduction. The reason is that in larger networks, there may be more feasible path, where the importance of optimization is exhibited better.



(a) $N = 50$ (b) $N = 100$ (c) $N = 200$

Fig. 4. Performance evaluation with two constraints

5 Conclusion

For the NP-completeness of the multi-constrained optimal QoSR problem, there is no efficient algorithm up to now. In the paper we summarize simulated annealing and propose a novel heuristic SA_MCOP to the general QoSR problem based on simulated annealing. SA_MCOP first takes the least-cost SPT as the initial solution and marks all of the nodes in the network. It then computes a new SPT and marks the nodes again in simulated annealing mode iteratively, until the path along the new SPT is feasible or maximum iteration time is reached. If a feasible path is found, the heuristic then starts to optimize the cost, where the feasibility is still guaranteed. Extensive simulations show that SA_MCOP achieves high performance with respect to both success ratio and cost optimization. It is also scalable in both network scale and constraint number k. Furthermore, it is insensitive to the distribution of QoS constraints. In addition, although the worst-case computation complexity is $O(Ik(m+nlogn))$, which is proportional to the maximum iteration time I, the practical running time is almost independent of I.

References

[1] Xiao, X., Ni, L. M.: Internet QoS: A big picture, IEEE Network, vol.13, no.2 (1999) pp.8–18 553

[2] Cui, Y., Wu, J. P., Xu, K., Xu, M. W.: Research on Internetwork QoS Routing Algorithms: A Survey. Chinese Journal of Software, vol.13, no.11 (2002) pp.2065–2075 553

[3] Cui, Y., Xu, K., Wu, J. P.: Precomputation for Multi-constrained QoS Routing in High-speed Networks. Proc. of IEEE INFOCOM'03 (2003) 553

[4] Wang, Z., Crowcroft, J.: Quality-of-service routing for supporting multimedia applications. IEEE Journal on Selected Areas in Communications, vol.14, no.7 (1996) pp.1228–1234 553

[5] Garey, M. S., Johnson, D. S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H.Freeman, New York (1979) 553

[6] Kirkpatrick, S., Gelatt, J. C. D., Vecchi, M. P.: Optimization by simulated annealing. Science, vol.220 (1983) pp.671–680 553, 555

[7] Korkmaz, T., Krunz, M.: Multi-Constrained Optimal Path Selection. Proc. of IEEE INFOCOM'01, vol.2 (2001) pp.834–843 561
[8] Dijkstra, E.: A Note on Two Problems in Connexion with Graphs. Numerische Mathematik, vol.1 (1959) pp.269–271 554
[9] Metropolis, N., Rosenbluth, A., Rosenbluth, M., et al.: Equation of state calculations by fast computing machines. Journal of Chemical Physics, vol.21 (1953) pp. 1087–1092 555
[10] Zegura, E. W., Calvert, K. L., Donahoo, M. J.: A Quantitative Comparison of Graph-based Models for Internet Topology. IEEE/ACM Transactions on Networking, vol.5, no.6 (1997) pp.770–783 559