Multi-constrained Routing Based on Simulated Annealing

Yong Cui[1], Ke Xu[1], Jianping Wu[1], Zhongchao Yu[2], Youjian Zhao[1]

1. Department of Computer Science, Tsinghua University, Beijing, P.R.China, 100084

2. Department of Computer Science, University of Maryland, College Park, MD 20742

{cy, xuke}@csnet1.cs.tsinghua.edu.cn; jianping@cernet.edu.cn; yuzc@cs.umd.edu; zhaoyj@csnet1.cs.tsinghua.edu.cn

***Abstract*-Multi-constrained quality-of-service routing (QoSR) is to find a feasible path that satisfies multiple constraints simultaneously, as an NPC problem, which is also a big challenge for the upcoming next-generation networks. In this paper, we propose SA_MCP, a novel heuristic algorithm, by applying simulated annealing to Dijkstra's algorithm. This algorithm first uses a nonlinear energy function to translate multiple QoS weights into a single metric and then seeks to find a feasible path by simulated annealing. The paper outlines simulated annealing algorithm and analyzes the problems met when we apply it to QoSR. Extensive simulations demonstrate that SA_MCP has good scalability regarding both network size and the number of QoS constraints with high performance. Furthermore, when most QoS requests are feasible, the running time of SA_MCP is about O(*k*(*m*+*n*log*n*)), which is only *k* times that of the traditional Dijkstra's algorithm, where *k* is the number of QoS constraints.**

***Keywords*-Simulated annealing, energy function, QoS routing, multiple constraints, scalability**

I. INTRODUCTION

Providing different quality-of-services (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]. The main function of QoSR is to find a feasible path that satisfies multiple constraints for QoS applications. QoS constraints can be divided into link constraints and path constraints. The link constraints of a path can be converted to the constraints of the bottleneck link in the path, such as bandwidth. It can be easily dealt with in a preprocessing step by pruning all links that do not satisfy these constraints and computing a path from the rest sub-graph. The path constraint is the restriction of each link along the path, such as delay. We will focus on the path constraint problem in this paper.

Many heuristics have been proposed for the multi-constrained QoSR problem because of its NP-completeness [4], [5]. However, these algorithms have some or all of the following limitations [2]: (1) High computation complexity prevents their practical applications; (2) Low performance means that these algorithms sometimes cannot find a feasible path even when one does exist. (3) Some algorithms work only for a specific network. This paper proposes a novel heuristic SA_MCP (Simulated Annealing for Multi-Constrained Path problem).

This algorithm first uses a nonlinear energy function to translate multiple QoS constraints into a single metric. The shortest path tree (SPT) of the whole network graph is then computed 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. The algorithm then labels each node according to the current SPT to compute a newer SPT with a lower temperature T. When T decreases to $T \to 0$, we have $\lim_{T\to 0} P(T) = 0$. Based on the theory about simulated annealing, SA_MCP is guaranteed to find a feasible path when one exists. Extensive simulations also show that SA_MCP performs well in finding a feasible path with high probabilities.

The rest of this paper is organized as follows. In Part II we give the problem formulation and summarize the simulated annealing. SA_MCP is proposed in Part III, and extensive simulations show the performance evaluation in Part IV. Finally, conclusions appear in Part V.

II. BACKGROUND INFORMATION

A. Problem Formulation

A network is represented by a directed graph $G(V,E)$, where 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 $(w_1(e), w_2(e), \cdots, w_k(e))$, which is also called QoS metric $w(e)$. For a path $p = v_0 \to v_1 \to \cdots \to v_n$ and $1 \le l \le k$, the weight $w_l(e) \in R^+$ satisfies the additive property if $w_l(p) = \sum_{i=1}^{n} w_l(v_{i-1} \to v_i)$.

Definition 1. *Multi-constrained path*

For a given graph $G(V,E)$, source node s, destination node t, $k \ge 2$ and constraint vector $c = (c_1, c_2, \cdots, c_k)$, the path p from s to t is called multi-constrained path (MCP) if $w_l(p) \le c_l$ for any $1 \le l \le k$. We write $w(p) \le c$ in brief.

Note $w(e)$ and c are also k-dimension vectors. For a given QoS request and its constraints c, QoSR seeks to find the path p based on the present network-state information, which satisfies $w(p) \le c$. Dijkstra proposed the shortest path tree (SPT) algorithm, which has a low time complexity [6]. However, QoSR problem is related to multiple weights at the same time. Thus the problem is changed to an NPC one that the original Dijkstra's algorithm cannot solve in

* 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).

polynomial time. In this case, one feasible method is to translate the multiple weights into a single weight, as follows:

Definition 2. *Energy Function*

We call $g(p)=\max_{l=1}^{k}(w_l(p)/c_l)$ the energy function of path p, where $c=(c_1,c_2,\cdots,c_k)$ is the constraint vector of a specific QoS application.

Because $g(p)$ is a nonlinear function, Dijkstra's algorithm is not guaranteed to find a path with least energy in polynomial time. Therefore, we propose a novel heuristic to the NP-complete QoSR problem by applying simulated annealing.

B. Simulated Annealing

The 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:

$$\Pr\{\overline{E}=E(r)\}=\frac{1}{Z(T)}\exp(-\frac{E(r)}{k_BT}) \quad \text{(i)}$$

$\overline{E}$ 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 slowly, the molecule of the body stays in different states with different probabilities, which satisfies 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

$$T0 >> E(r)/k_B, \quad \text{(ii)}$$

then $E(r)/k_B T0 \approx 0$. As a result, $\Pr\{\overline{E}=E(r)\}\approx 1/Z(T0)$, i.e. the probabilities are approximately equal.

(2) When it cools down so that temperature becomes 0 degree, 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

$$\Pr\{\overline{E}=E(r)\}=\begin{cases}1 & r=r^*\\ 0 & others\end{cases}. \quad \text{(iii)}$$

The idea of simulated annealing was first proposed by Metropolis [7] and was applied to combinational optimization successfully in 1983 [8]. 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 each time. When the temperature decreases, $P(T)$ also decreases. When $T\to 0$, it is guaranteed to find an global optimal solution since $P(T)$=0.

III. Simulated Annealing Based Heuristic

A. The Idea of Our SA_MCP

Because only an end-to-end path is a solution in routing computation, the main problem of using simulated annealing for QoSR is how to iterate to another solution from the current solution in its neighborhood. To achieve this, we propose a method with iteration. In the process of iteration, we use Dijkstra's algorithm to guarantee that the new solution is an end-to-end path. When we calculate the SPT by Dijkstra's algorithm, we select with probability $P(T)$ a node that is not the current optimal node. Thus, our SA_MCP can overcome the local optimization problem that all heuristics face.

We assume that each node in the network maintains a consistent copy of the global network state information. For a given QoS request from s to t, node s uses our SA_MCP to seek a feasible path. It first uses Dijkstra's algorithm to calculate the least-hop SPT rooted by s and marks each node in the network. Then it uses improved Dijkstra's algorithm to label each node in iteration. In each iteration, SA_MCP computes new labels based on the labels computed last time. At the same time, it selects different links with different probabilities P(T) including non-optimal links, where $\lim_{T\to 0}P(T)=0$ satisfies the formula (iii). After each of iterations, the temperature T decreases. When the algorithm iterates enough times, we guarantee $T\to 0$.

B. Pseudo-code Description

Fig. 1 and Fig. 2 show the pseudo-code of the algorithm, where SA_MCP is the main function. The input of SA_MCP includes a given graph with multiple QoS weights, a QoS request from s to T and a constraint vector $c=(c_1,c_2,\cdots,c_k)$. In addition, we can configure the initial temperature (T0), the gradient of cooling down the temperature (grad) and the iteration time (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 I shows the notations used in the pseudo-code.

1. Function SA_MCP

In function SA_MCP, we first use Dijkstra's algorithm to compute the least hop 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 including (1) computing the SPT rooted by T (Line 5); (2) computing the SPT rooted by s (Line 8). After the whole SPT is computed each time by Dijkstra's algorithm or SA_Dijkstra, d[.] is updated to save the newly computed weights from the new SPT to source s (Line 12-13 in function SA_Dijkstra). SA_Dijkstra computes 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 judged to see whether it satisfies the constraint c. If it does, the algorithm then returns the path successfully (Line 3, 6 and 9). We need to set the temperature T for simulated annealing to construct SPT in SA_Dijkstra each time (Line 7 and 10).

2. Function SA_Dijkstra

Based on the original Dijkstra's algorithm [6], we propose the SA_Dijkstra function, which seeks the least energy tree by simulated annealing. The function first initializes variables

```
SA_Dijkstra(G,root,T)
1. FOR each node T in G
2.     g[t]=INFINITY
3. SPT={root}
4. r[root]=0
5. NB={root's neighbors}
6. WHILE NB is not empty
7.     u=SA_Cheapest(NB, T)
8.     addNode(SPT,u,NB)
9.     FOR each node v in u's neighbor
10.        IF v is not in SPT THEN
11.            SA_Relax(u, v)
12.FOR each node T in G
13.    d[t]=r[t]

SA_MCP(G=(V,E), s, t, c, T0, grad, I)
1. T=T0
2. Dijkstra(G,s); //get an initial solution
3. IF (d[t]<c) RETURN this path
4. FOR( i=0; i<I; i++)
5.     SA_Dijkstra(G,t,T); // min g(p)
6.     IF (d[s]<c) RETURN this path
7.     T=T/grad
8.     SA_Dijkstra(G,sT); // min g(p)
9.     IF (d[t]<c) RETURN this path
10.    T=T/grad
11.RETURN failure
```

Fig. 1. Pseudo-code of SA_MCP

```
SA_Cheapest(NB, T)
1. g*= min_{v∈NB} max_{l=1}^{k} (r_l[v]+d_l[v])/c_l
2. FOR each v in NB
3.     E(v) = max_{l=1}^{k} (r_l[v]+d_l[v])/c_l -g*
4. Z = Σ_{v∈NB} exp(−E(v)/T)
5. x= Z × uniform(0,1)
6. sum=0
7. FOR each u in NB
8.     sum=sum+ exp(−E(u)/T)
9.     IF sum>=x THEN RETURN u

addNode(SPT,u,NB)
1. NB=NB-{u}
2. SPT=SPT+{u}
3. FOR each node v in u's neighbor
4.     IF v is not in SPT THEN
5.         NB=NB+{v}

SA_Relax(u, v)
1. tmp= max_{l=1}^{k} (r_l[u] + w_l(u,v) + d_l[v])/c_l
2. IF g[v]>tmp THEN
3.     g[v]=tmp
4.     r[v]=r[u] + w(u,v)
5.     Πr[v]=u
```

Fig. 2. Sub-functions of SA_Dijkstra

(Line 1-5). Then when *NB* is not empty (Line 6), it selects the optimal node *u* in *NB* by simulated annealing (Line 7) and adds the selected node to the current partial SPT (Line 8). Finally, it relaxes *u*'s neighbors that are not in the SPT (Line 9-11). After it creates the whole SPT, the weight $r[t]$ of the path along this SPT from root to each node *T* is saved in the global variable $d[t]$ for computing new SPT next time (Line 12-13).

3. 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, $\max_{l=1}^{k}(r_l[v]+d_l[v])/c_l$ is the energy of a path defined by definition 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 formula (i) 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 formula (i) (Line 5-9).

4. Function addNode

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).

5. Function SA_Relax

We relax node *v* with node *u* being *v*'s parent. The energy

TABLE I
NOTATIONS IN THE PSEUDO-CODE OF SA_MCP

Symbol	Meanings	Symbol	Meanings
T0	initial temperature	*root*	the root node for calculating the current SPT
$E(v)$	the energy of node *v* in formula (i)	$d[u]$	backward weights: the *k*-dimensional weights of the path along the old SPT from its root to *u*
grad	gradient for decreasing temperature	*v*	a child node of node *u*
I	maximum number of iterations	$\Pi r[v]$	the precedent node of node *v*
Dijkstra(*G*,*s*)	original Dijkstra's algorithm for SPT rooted by *s*	$g[u]$	the energy of node *u*
u	an intermediate node	*NB*	the set of the neighbors of the current SPT
SA_Dijkstra (*G*,*s*,T)	a heuristic for SPT rooted by *s* based on simulated annealing, where *T* is the temperature	$r[u]$	forward weights: the *k*-dimensional weights of the path along the current SPT from its root to *u*
Z	the normal factor in formula (i)	*g**	a locally minimal energy
c	*k*-dimensional constraints of a QoS request	SPT	a partially created SPT

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) and the precedent node (Line 5).

C. Complexity and Parameters

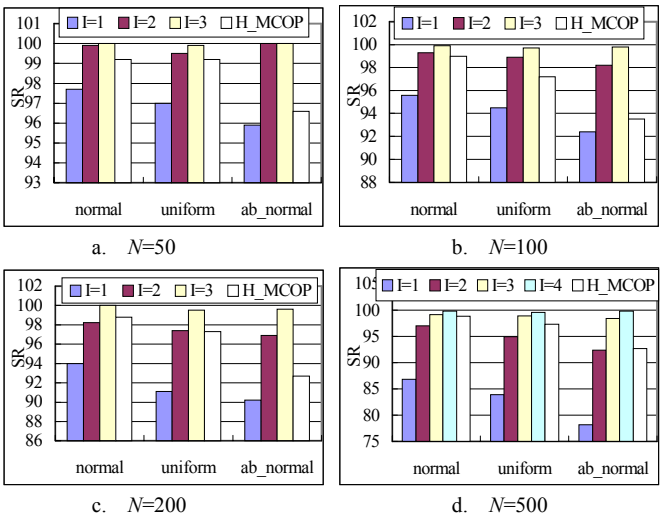
We now analyze the computation complexity of SA_MCP. In the network graph $G(V,E)$ with k metrics, we assume $m=|E|$ is the number of edges and $n=|V|$ is the number of nodes. Since the computation complexity of the improved Dijkstra's algorithm is $O(m+n\log n)$, that of function SA_Dijkstra is $O(km+kn\log n)$. As a result, including the recursion, the overall computation complexity of SA_MCP is $O(Ik(m+n\log n))$, 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_MCP is almost independent of the maximum number of iterations as we are going to show in part IV.

Simulated annealing requires selecting a new solution randomly enough at the beginning, i.e. a high enough initial temperature $T0$. Because the energy $E(v)$ is often much less than 1 in SA_MCP, it suffices to set $T0=1$ to satisfy formula (ii). 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 $T(2I)=10^{-2I} << E(v)$ satisfies formula (iii). The following simulations show that such parameters achieve high performance.

IV. Performance Evaluation

About the performance of our algorithm, we will show (1) the relationship between the distribution of QoS constraints and the performance of our SA_MCP with only two constraints, and (2) the performance with multiple constraints. In each part, SA_MCP is compared with H_MCOP [9], [10], which is a good heuristic in the literature.

In each experiment, we simulate purely random network graphs with N nodes [11] and generate k weights for each link, where $w_l(e)$ ~uniform[1,1000] for $l=1,2,\cdots,k$ and $w_l(e)$ has no correlation for different e or l. We simulate 10 graphs with N being 50, 100, 200 and 500, 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 two. Each source node s uses SA_MCP 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.



a. N=50 b. N=100

c. N=200 d. N=500

Fig. 3. Performance evaluation with two constraints

A. 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 $g(s,t)=\sum_{l=1}^{k} a_l w_l(s,t)$. Finally, we take the weights of the path p(s,t) as the QoS constraints of the pair (s,t), i.e. c(s,t)= w(p(s,t)).

In the case of two dimensions, we let $a_1 \in [0,1]$ and $a_2 = 1-a_1$ for simplicity. Because different QoS applications concern a weight to different degrees, we use the following three methods to generate a. (1) NORMAL: a_1 ~normal(0.5,0.16); (2) UNIFORM: a_1 ~uniform(0,1); (3) AB_NORMAL: a_1 ~normal(0,0.16) and $a_1 \in [0,0.5]$. 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.

The relation between the maximum number of iterations and the performance of our SA_MCP is shown in Fig. 3 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_MCP does not perform well. The main reason is that $T0$=1 is much greater than energy $E(v)$ and the strong randomicity cannot guarantee an optimal path. With more iterations, the performance of SA_MCP 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

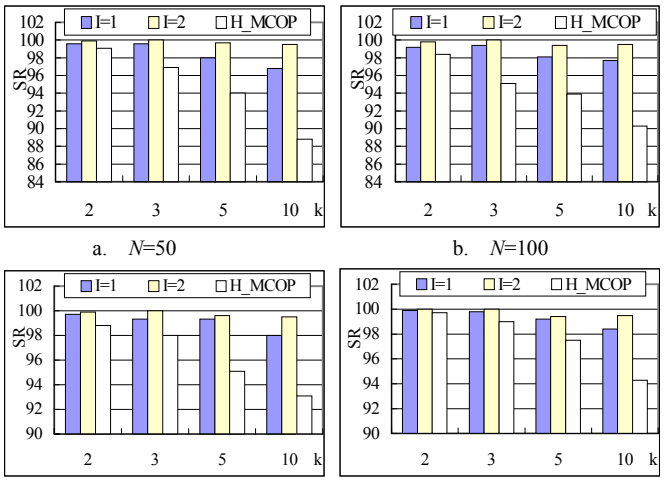


Fig. 4. Performance evaluation with multiple constraints

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. 3), H_MCOP performs well; otherwise, it will degrade, especially in the ab_normal distribution. On the contrary, our SA_MCP performs well in all conditions, including different distributions of QoS constraints and different network scales.

B. Performance with Multiple Constraints

In order to study the relation between the maximum number of iterations and the performance of SA_MCP, we use the following method to generate the constraints for a given (*s*,*t*). We first take the random number $b_l \sim uniform(0,1)$ for $l = 1,2,\cdots,k$ and calculate $a_l = b_l \Big/ \sum_{l=1}^{k} b_l$. We then construct the least energy path from *s* to *T* according to the energy function $g_1'(p) = \sum_{l=1}^{k} (a_l w_l(p))$ and take the weights of the path as the constraints of the given (*s*,*t*), i.e. $c(s,t) = w(p_i)$.

Fig. 4 shows the performance for multiple constraints, where the x-axis is the number of constraints and the y–asix is SR. SA_MCP performs well for large *k*, while H_MCOP does not. Furthermore, our SA_MCP has good scalability on the size of network with multiple constraints.

C. Analysis of Running Time

In order to analyze the relationship between the running time and the maximum number of iterations, we measure the running time of SA_MCP on a PC running MS Windows 2000 with a Pentium III 933 CPU and a 256M memory. As an example, Table II shows the running time with *N*=500 and *k*=10. In most cases the heuristic can find a feasible path with only one iteration. Therefore, the running time depends on the maximum number of iterations lightly instead of the linearity in theory, as shown in Table II. Such a relation further confirms the conclusion of Fig 4, i.e. a feasible path can be found with only a few iterations.

TABLE II
THE RUNNING TIME OF PROPOSED SA_MCP

the maximum number of iterations	*I*=1	*I*=2	*I*=3	H_MCOP
running time(millisecond)	19.54	20.36	21.07	21.90

V. CONCLUSION

For the *NP*-completeness of multi-constrained QoSR problem, there is no efficient algorithm up to now. We propose a novel heuristic SA_MCP based on simulated annealing. The paper summarizes simulated annealing, and analyzes both the difficulty and our solution by applying it to the routing computation. Our SA_MCP first takes the least-hop SPT as the initial solution and marks all of the nodes in the network. It then computes a new SPT and sets new labels again in simulated annealing iteration mode, until the path along the new SPT is feasible or maximum iteration time is reached. In each iteration, it uses the current labels to seek the least energy SPT with a nonlinear energy function. Extensive simulations show that SA_MCP has high performance and good scalabilities with respect to 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 $O(Ik(m+n\log n))$, which is proportional to the maximum iteration time *I*, the running time is about $O(k(m+n\log n))$, which is independent of *I*.

REFERENCE

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

[2] Y. Cui, J. P. Wu, K. Xu, et al. Research on internetwork QoS routing algorithms: a survey. Chinese Journal of Software, vol.13, no.11, pp.2065—2075, 2002.

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

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

[5] Z. Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications., IEEE Journal on Selected Areas in Communications, vol. 14, no. 7, pp. 148–154, Sept. 1996.

[6] E. Dijkstra, A note on two problems in connexion with graphs. Numerische Mathematik, vol.1, pp. 269-271, 1959.

[7] N. Metropolis, A. Rosenbluth, M. Rosenbluth, et al. Equation of state calculations by fast computing machines. Journal of Chemical Physics, vol. 21, pp. 1087-1092, 1953.

[8] S. Kirkpatrick, J. C. D. Gelatt, M. P. Vecchi. Optimization by simulated annealing. Science, vol. 220, pp. 671-680, 1983.

[9] T. Korkmaz, M. Krunz, Multi-constrained optimal path selection. IEEE INFOCOM'01, vol. 2, pp. 834 -843, 2001.

[10] G. Feng, C. Douligeris, K. Makki, et al. Performance evaluation of delay-constrained least-cost QoS routing algorithms based on linear and nonlinear lagrange relaxation. IEEE ICC'02, 2002.

[11] E. W. Zegura, K. L. Calvert, M. J.Donahoo, A quantitative comparison of graph-based models for Internet topology. IEEE/ACM Transactions on, Networking, vol. 5, no. 6, pp. 770–783, Dec. 1997.