Solving the MCP Problem Heuristically

Wen-Lin Yang
Department of Information Technology
National Pingtung Institute of Commerce
#51, Ming-Sheng East Road, Pingtung City, Taiwan.
Email: wly@npic.edu.tw Fax: 886-8-7223962

ABSTRACT

The multi-constrained path (MCP) selection problem occurs when the quality of services (QoS) are supported for point-to-point connections in the distributed multimedia applications deployed on the Internet. This NP-complete problem is concerned about how to determine a feasible path between two given end-points, so that a set of QoS path constraints can be satisfied simultaneously. Based on the branch-and-bound technique and tabu-searching strategy, an optimal algorithm and a tabu-search based heuristic algorithm are developed in this paper for solving the MCP problem with multiple constraints. The experimental results show that our tabu-search based heuristic algorithm not only outperforms the previous published method in [4], but also demonstrates that it is indeed a highly efficient method for solving the MCP problem in large-scale networks.

Keywords: Quality-of-service, Branch-and-bound, Tabu-search, Multi-constrained path

Contact author: Wen-Lin Yang

Workshop: Workshop on Computer Networks

1. INTRODUCTION

For real-time multimedia applications deployed on the broadband integrated services networks, various QoS requirements, such as bandwidth, delay, cost, delay jitter, packet loss rate, etc., must be supported in order to provide appropriate service quality [8,9,11]. To provide such QoS-based services, a routing problem concerning how to select a feasible path from a source node to a destination node, so that all the given QoS constraints can be satisfied simultaneously. This QoS routing problem is often referred as a multi-constrained path selection problem (MCP) [1,2,3,4].

The MCP problem is known to be NP-complete [1,12]. To cope with the NP-complete, a number of heuristics were developed for the MCP problem in the past [1,2,3,4,6]. In [1,2,6,10], the MCP problem with only two independent additive QoS constraints was studied, and polynomial-time heuristic algorithms were proposed. For the MCP problem with multiple additive QoS constrains, two heuristic methods have been published in [3,4]. A limited path heuristic modified from extended Bellman-Ford algorithm [5] was developed in [3], where the simulation results show that this method works well for a small mesh of 64 nodes. However, since it requires $O(|N|^2 \ln(|N|))$ space in each node to store all the legal partial paths, the required running space could be enormous for networks with a large amount of nodes. Hence, this approach is not suitable for the MCP problem in large-scale networks. In [4], a randomized heuristic algorithm with $O(N^2)$ time complexity and $O(N)$ storage complexity was proposed for the MCP problem with multiple constraints. Although this heuristic algorithm can efficiently solve small-size MCP problems, its performance degrades very fast as the distance between two given end-points is increased. A simulation study of this algorithm is presented in section 5.

In this paper, an optimal algorithm and a heuristic algorithm are presented for solving the MCP problems with multiple constraints. The optimal algorithm is developed based on the branch-and-bound technique, and its performance is much more efficient than the extended Bellman-Ford based optimal algorithm proposed in [5,13]. As for the heuristic algorithm, it was developed based on the tabu searching strategy and the branch-and-bound based optimal algorithm.

A number of simulations are presented in section 5. The experimental results show that our tabu-search based heuristic method not just performs very well on a number of different network topologies, but also demonstrates that it is indeed a highly efficient method for solving the large-scale MCP problems.

2. THE MCP PROBLEM

Consider a network that is modeled by a directed graph $G(V,E)$, where V is a set of nodes and E is a set of links. Each link $(u,v) \in E$ is associated with K positive additive QoS parameters: $w_i(u,v),\ i = 0,1,....,(K-1)$. For non-additive QoS parameters like bandwidth, a pre-processing procedure can be invoked to remove links with bandwidth less than requirement. Hence, only positive additive QoS parameters are considered in this study.

For any link e from node u to node v, the following notation: $w(u,v) = w(e) = (w_0(e), w_1(e), ..., w_{K-1}(e))$, represents K QoS parameters assigned on link e. In addition, for a path P and a QoS parameter i, the path weight $w_i(P)$ is defined as the summation of $w_i(e)$ on every link e along the path P.

Given K constraints C_i, $0 \le i \le (K-1)$, and a pair of nodes *S* and *T* representing a source node and a destination node respectively, the goal of the MCP problem is to find a path *P* from *S* to *T* such that $w_i(P) \le C_i$, where $0 \le i \le (K-1)$. A path that can simultaneously satisfy all the QoS constraints is called a feasible path.

3. THE OPTIMAL ALGORITHM

In this section, we present an optimal algorithm for the MCP problem. The algorithm is developed based on the branch-and-bound technique and given in Figure 3. A heuristic procedure modified from the optimal algorithm is used as a kernel function in our tabu-search based heuristic algorithm, which is represented in section 4.

3.1. The branch-and-bound algorithm

In order to solve the MCP problem optimally, a data structure called state-space tree is generated from a given network G to record all the feasible paths. Based on the state-space tree, a branch-and-bound based algorithm is applied to search for a feasible path. The state-space tree is constructed in the following ways.

Let the source node of network G be the root of the state-space tree. As shown in Figure 2, for each state node S_j in the state-space tree, two labels mark it: one is the node number j in the original network G and another one is an attribute vector. Let Y_j denote the attribute vector of state node S_j. Let K denotes the number of QoS constraints, and Y_j can then be defined as follows:

(1) $Y_j = (obj\_value_j, w_{j,1}, ..., w_{j,K})$, where $w_{j,i} = \sum_{(u,v)\in P} w_{j,i}(u,v)$, $1 \le i \le K$.

(2) In Figure 3, obj_value_j in the attribute vector Y_j is computed based on the following equation:

$$obj\_value_j = \sum_{i=1}^{K} [w_i(s->j) - C_i], \quad \text{..... (1)}$$

where obj_value_j = a positive integer, if $[w_i(s->j) - C_i] > 0$, for any QoS parameter i.

In the above equation, we use $w_i(s->j)$ representing the summation of $w_i(e)$ on every link e along the path from source node s to node j.

For any state node S_u with node-label u, a new state node S_v is created for each down-stream node v, if the link (u,v) is in the network G. In addition, these new state nodes are made to be children of S_u, and they are at the same level in the state-space tree. For any intermediate state node S_j with node-label j, down-stream nodes of node j cannot be added in the state-space tree if the $w_{j,i} > C_i$, where C_i is a given constraint and $1 \le i \le K$. Since one of QoS constraints is violated, the path from root to state node S_j is not a feasible path for the MCP problem. As a result, the state node S_j becomes a leaf node in the state-space tree.

By applying the above branching process recursively, the entire state-space tree is then obtained for the MCP problem. The destination node must appear at some of the leaf nodes of the state-space

tree. Since the goal of our problem is to find a feasible path that satisfies all the path constraints, we may speed up this searching process by giving priorities to the state nodes that are eligible for branching. That is, the state node *j* with the smallest obj_value_j should have the highest priority to be selected for branching. This searching strategy is based on an observation that a state node with smaller obj_value would have more chance to lead to a feasible path. A priority heap maintained in Figure 3 is used to keep all the state nodes, where the node with the smallest obj_value should be always on the root of the heap. As shown in lines 24~25 of Figure 3, only state nodes that contain non-destination nodes and their obj_value are not greater than zero, are eligible for further branching, and are thus stored in the heap.

The time complexity of our branch-and-bound algorithm is bounded by $O(d^n)$ with n denoting the number of nodes in the network and d representing the largest node-degree, since at the worst case the height of the state-space tree is at most $(n-1)$ and the number of children of any state node could be as large as d.

3.2. The example

To illustrate our branch-and-bound method for the MCP problem, a numerical example is given in Figure 1 and 2. Based on a six-node network given in Figure 1, the nodes 0 and 5 are assumed to represent the source node and destination node respectively, and the paths between these two nodes must satisfy two QoS constraints, C_0 and C_1, which are no more than 5. A state-space tree is then constructed in Figure 2, where all the state nodes are numbered based on their creating sequence.

For example, state nodes s_4 and s_5 are created earlier than state nodes s_6 and s_7, because the obj_value of s_3 is -6 and the obj_value of s_2 is -7. The branching process occurs on state node s_2 is earlier than state node s_3. For the same reason, the branching process occurs on state node s_3 is earlier than state node s_5. Note that the path $s_0 \to s_1 \to s_3 \to s_7$ is not feasible. The branching process stops after the first feasible path $s_0 \to s_2 \to s_5 \to s_8$ is found.

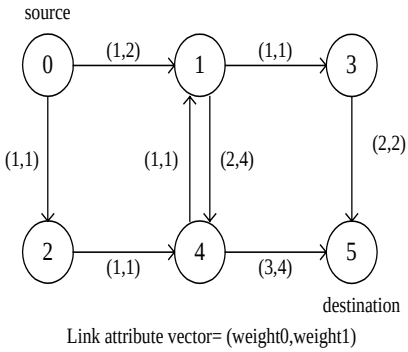

Figure 1. An example of 6-node network

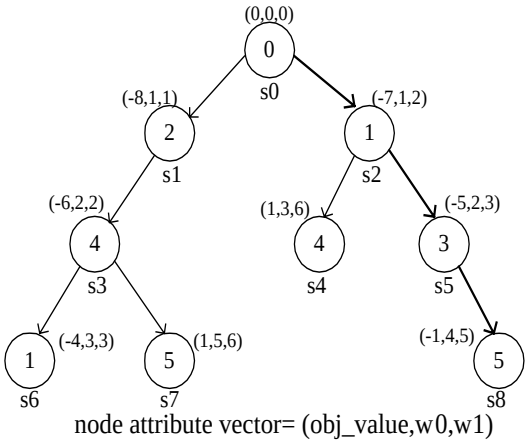

Figure 2. State-space tree construction based on branch-and-bound algorithm

The branch-and-bound based procedure {

(1) Initialize a node heap H;

(2) Let S denote the source node, and T denote the destination node;

(3) Let C_j = the jth constraint, $1 \le j \le K$;

(4) Let $w_j(u,v)$ = the jth QoS parameter on link (u,v);

(5) Let state S_0 store the source node S and be the root of the state-space tree;

(6) Add S_0 to heap H;

(7) While ($H \notin \varphi$) {

(8) $S_u = remove\_top(H)$;

(9) For each node v adjacent to node u stored at state S_u {

(10) If (v is not on the path from S_0 to S_u, and v is unvisited from u) {

(11) Create a new state node S_v for v based on the information stored in state node S_u of u;

(12) $S_v -> branch = YES$; $S_v -> path = S_u -> path + (u,v)$;

(13) Let P_v denote the path from S_0 to S_v;

(14) $obj\_value_v = \sum_{j=1}^{K}[w_j(P_v) - C_j]$, where $obj\_value_v = 1$ if $[w_j(P_v) - C_j] > 0$;

(15) If ($obj\_value_v > 0$) {

(16) $S_v -> branch = NO$; }

(17) Else {

(18) If ($v == T$) {

(19) $S_v -> branch = NO$;

(20) If ($obj\_value_v \le 0$)

(21) Return P_v;

```
(22)              }
(23)              Else {
(24)                 Add  v  to heap  H ;
(25)                 Make the new state node  S_v  to be a child-node of state node  S_u ;
(26)     }}}} }
(27)     Output "No feasible path found.";
(28)  }
```

Figure 3. The branch-and-bound based optimal algorithm

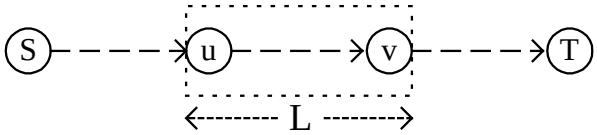

Figure 4. A partial path of length L.

4. A TABU-SEARCH BASED ALGORITHM

Since the MCP problem is NP-complete, the optimal algorithm presented in the previous section performs well only for networks with a small number of hops between source and destination nodes. For large-scale networks, it may take too much CPU time to traverse the whole state-space tree to find a feasible path. Hence, a tabu-search based heuristic algorithm is developed in this section to solve the MCP problem in large-scale networks.

Given an infeasible path *P* from node *S* to node *T* as shown in Figure 4, our tabu-search based heuristic algorithm was designed to iteratively reduce the cumulative link weights on randomly selected partial paths R of *P*. This reduction is achieved by replacing old path R with a new partial path R' of lower cost, where cost is computed by a cost function defined in the equation (2). The searching process of our heuristic algorithm can be outlined as follows:

(a) To find an infeasible path *P* from *S* to *T*, where $P \in M$ and M is defined in the following:

$M = \{ \Psi_i \mid \Psi_i$ *represents the path with the smallest value of* $w_i(P)$ *among all paths* P *from S to T, where* $w_i(P) = \sum_{e \in P} w_i(e)$ *and* $1 \le i \le K \}$.

The set M can be determined by finding a set of the shortest paths between source and destination nodes, where each shortest path is found based on a criterion that is to minimize the

summation of all link-weights along the path based on a QoS parameter. Hence, the cardinality of set M is equal to the number of QoS parameters. The heuristic procedure stops if there is a feasible path found in the set M.

(b) A pair of nodes *u* and *v* is selected on path *P*, where $P = \overrightarrow{S->u} + \overrightarrow{u->v} + \overrightarrow{v->T}$. All the links on path $\overrightarrow{u->v}$ are then removed. Note that the length of the selected path $\overrightarrow{u->v}$ should not be greater than a given integer *L*.

(c) A branch-and-bound based heuristic (*BBH*) procedure given in Figure 5 is applied to search for a new partial path $R^{'}$ for nodes *u* and *v*. The *BBH* procedure is modified from the branch-and-bound based optimal algorithm given in Figure 3. The main goal of the *BBH* procedure is to find a new path $R^{'}$, which is from node *u* to node *v*. The path $R^{'}$ is determined by the following cost function $Y(R^{'})$:

$$Y(R^{'}) = \underset{R^{'}}{Minimization}\{\sum_{i=1}^{K} A_i(R^{'})\}, \quad \text{.....(2)}$$

where $A_i(R^{'}) = w_i(R^{'}) - C_i(R^{'})$, if $w_i(R^{'}) \le C_i(R^{'})$;

$A_i(R^{'}) = 10 * (w_i(R^{'}) - C_i(R^{'}))$, if $w_i(R^{'}) > C_i(R^{'})$;

$$C_i(R^{'}) = C_i(\overrightarrow{S->T}) - C_i(\overrightarrow{S->u}) - C_i(\overrightarrow{v->T}), \ 1 \le i \le K .$$

In the above equations, $C_i(R^{'})$ represents the *ith* residual QoS constraint for the path $R^{'}$. Since our goal is to rebuild a complete feasible path $P^{'}$ from node *S* to node *T* at step (d), it is not necessary to require the partial path $R^{'}$ obtained at this step must be feasible with respective to $C_i(R^{'})$. In fact, we only need a 'good' partial path $R^{'}$, which can lead to a path $P^{'}$ found at step (d), where $P^{'}$ could be infeasible. Hence, based on the cost function defined in equation (2), the *BBH* procedure is developed to search for a new path $R^{'}$ whose cost value Y is minimal. Note that for the case that path weight is greater than path QoS constraint (i.e., $w_i(R^{'}) > C_i(R^{'})$), we make the path undesirable by multiplying the cost value of the path by 10.

(d) Let $P^{'} = \overrightarrow{S->u} + R^{'} + \overrightarrow{v->T}$. If $P^{'}$ is not a feasible path, then $P = P^{'}$ and jump back to

step (a). Otherwise, the procedure stops.

(e) The steps from (b) to (d) can be put together to form a path-rebuilding procedure, which is embedded into a tabu-search based iteration loop, where a circular queue is implemented as a tabu list for storing all the selected paths $\overrightarrow{u->v}$. Only selected partial path that are not in the tabu list are eligible for path rebuilding. The tabu-searched based procedure stops when a feasible path is found or a fixed number of iterations have elapsed.

Based on the above ideas, the complete pseudo-code of our heuristic algorithm is developed in Figure 6. At line 8 in Figure 6, the initial value of *L* is set to be $L_{\min}$, and the value is increased by one at line 28 if path R' cannot be obtained after a certain number of iterations. Since the function *BBH* is developed based on the branch-and-bound technique, it is only efficient for small-size meshes. Furthermore, its performance degrades very fast as the mesh size exceeds some threshold value, which is usually a small integer. Hence, the maximum value of *L* is set to be a small integer, $L_{\max}$. The value of *L* is reset back to $L_{\min}$ at line 16 if a path R' is found. In section 5.3, a set of simulations is carried out to show that the values of $L_{\min}$ and $L_{\max}$ would affect the performance of our tabu-search based method.

One may use the branch-and-bound based optimal algorithm shown in Figure 3 to search for a feasible path R' for nodes *u* and *v*. According to our simulation results, the heuristic algorithm developed based on the *BBH* procedure performs much better than the one developed based on the optimal algorithm. This phenomenon is due to the fact that it is too hard to obtain a feasible partial path R' for nodes *u* and *v*; as a result, a large amount of iterations in the tabu-searching process are wasted and the probability of finding a feasible path is low. On the other hand, although the goal of the *BBH* procedure is to find a "good" path R' and the rebuilt path P' based on R' may be infeasible, however, the tabu-search based procedure can be prevented from being stuck into some searching area by accepting an infeasible path P' at line 23 in Figure 6. Hence, the searching space explored by *BBH* procedure is much larger and the probability of finding a feasible path by

the tabu-search based procedure is much higher.

The time complexity of heuristic procedure in Figure 6 is bounded by the number of times of the *BBH* procedure at line 14 being executed. Hence, it is $O(d^n * ITERATIONS)$ at the worst case.

Function BBH (*src, dest, R,* $C_i(R)\ \forall i$) {
(1) Compute $Y(R)$ using the cost function defined in equation (2);
(2) Let $BEST\_COST = Y(R)$; $R' = \phi$;
(3) Let state S_0 store the source node *src* and be the root of the state-space tree;
(4) Add S_0 to an empty node heap H;
(5) While ($H \notin \varphi$) {
(6) $S_u = remove\_top(H)$;
(7) For each node v adjacent to node u stored at state S_u {
(8) If (v is not on the path from S_0 to S_u, and v is unvisited from u) {
(9) Create a new state node S_v for v based on the information stored in state node S_u of u;
(10) $S_v -> branch = YES$; $S_v -> path = S_u -> path + (u,v)$;
(11) Let P_v denote the path from S_0 to S_v;
(12) Let $\cos t_v$ be the path cost of P_v, and compute the value of $\cos t_v$ using the cost function defined in equation (2);
(13) If ($\cos t_v > Best\_COST$) { $S_v -> branch = NO$;}
(14) Else {
(15) If ($v == dest$) {
(16) $BEST\_COST = \cos t_v$; $R' = P_v$; $S_v -> branch = NO$;
(17) }
(18) Else {
(19) Add v to heap H;
(20) Make the new state node S_v to be a child-node of state node S_u;
(21) }}}} }
RETURN R';
}

Figure 5. The branch-and-bound based heuristic procedure

Tabu-search based heuristic procedure {
(1) Given a network $G = (V, E)$
(2) Let S denote the source node, and T denote the destination node;

(3) Let C_j = the jth constraint, $1 \le j \le K$;

(4) Let *F={P| P is a path from S to T, and P is determined by Dijkstra algorithm based on K QoS parameters}*;

(5) Let $P \in F$, and *P* contains the longest partial path that satisfies all the constraints;

(6) Let *P* be the initial path for iterations;

(7) Initialize a circular queue *Q* to be the tabu list;

(8) $j = k = h = 0$; Given two small integers $L_{\min}$, $L_{\max}$; Let $L = L_{\min}$;

(9) While (j < ITERATIONS) {

(10) Randomly select a partial infeasible path *R* on *P* where $R \subset P, |R| \le L$;

(11) Assume $P = P_1 + R + P_2$ and R is a path from node *u* to node *v*;

(12) If ($R \notin Q$) {

(13) $Q = Q \cup R$; $C_i(R) = C_i - C_i(P_1) - C_i(P_2)$, $1 \le i \le K$;

(14) $R' = BBH(u, v, R, C_i(R) \ \forall i)$;

(15) If ($R' \neq \phi$) {

(16) $L = L_{\min}$;

(17) Rebuild a new path P' from node *u* to node *v* such that $P' = P_1 + R' + P_2$;

(18) $orig\_path = P'$; //remember the original path

(19) If (P' is a feasible path)

(20) RETURN P';

(21) Else

(22) $h = 0$;

(23) $P = P'$;

(24) }

(25) Else {

(26) $k++$;

(27) If ((k > ITERATIONS/10) *AND* ($L < L_{\max}$)) {

(28) $L++$; $k = 0$;

(29) }

(30) $h++$;

(31) If (h > ITERATIONS/5) {

(32) $P = orig\_path$; $h = 0$;

(33) } } }

(34) $j++$;

(35) }

(36) Output “No feasible path found.”;

(37) }

Figure 6. Tabu-search based heuristic procedure

5. SIMULATION RESULTS

In this section, we have several sets of experiments to compare performance and efficiency for the optimal and heuristic algorithms presented in this paper for the MCP problem. All the simulations are done with the following experimental parameters: PIII 866 MHz CPU, 512MB RAM, Linux OS. The simulation programs were developed by $C++$. For all benchmarks, the QoS parameters (link weights) on each link are randomly selected from 0 to 10.

For the MCP problem, a heuristic method may not find a feasible path; even at least one feasible path exists in the network. Hence, in this study, a parameter named "success ratio" is used to evaluate the performances of the heuristic algorithms. The "success ratio" is defined as follows:

success ratio= number of feasible paths found by a heuristic algorithm / number of feasible paths found by an optimal algorithm $*100\%$.

Three algorithms are studied in this simulation. The first one is the branch-and-bound based optimal algorithm presented in Figure 3. The other two are the tabu-search based heuristic algorithm given in Figure 6 and the TK_MK heuristic algorithm proposed in paper [4].

5.2. Benchmark generation

In order to measure the performance more accurately, two network topologies: ANSNET and mesh given in Figure 7 and 8 are used in the simulations. For each network topology, two different methods for generating benchmarks are designed to do performance comparisons for our tabu-search based heuristic and TM heuristic. These two methods are described as follows:

(a) Based on a given pair of source and destination nodes of a network, a number of benchmarks are generated by randomly assigning weights on each link of the network. The source and destination nodes are chosen in a way that the number of hops of the path between them is as largest as possible. In Figure 7, for example, assuming node 1 and node 32 be source and destination nodes respectively, a number of benchmarks are generated by randomly assigning weights on each link of the ANSNET. As for the meshes shown in Figure 8, the source and

destination nodes are located at the two-ends of the longest diagonal line of a mesh.

(b) This method is similar to the first one except that a pair of source and destination nodes is randomly selected for each benchmark generated using the method (a). That is, at most N distinct pairs of source and destination nodes may exist for N benchmarks generated.

Given a set of different number of nodes, two sets of benchmarks using the above two methods are generated for simulations carried out in section 5.2. For simulations done in section 5.3, 5.4 and 5.5, the benchmarks tested are generated by the method (a) only.

5.2. Performance

(A) ANSNET

A network topology shown in Figure 7 is modified from ANSNET [7], which was studied in [3,4]. In Table 1 and 2, each data is obtained based on 1000 benchmarks. The constraints, C_0 / C_1, are chosen in a way that the number of feasible paths found by the optimal algorithm can be spanned in a wide range.

As shown in Table 1 and 2, the average success ratios of two heuristics are more than 99% based on different values of QoS constraints. The experimental results show that our tabu-search based heuristic algorithm is almost as good as the TK_MK method for the ANSNET.

(B) Mesh

Like the ANSNET, for a $N \times N$ mesh, two sets of benchmarks are used to compare performance between two heuristic algorithms. The experimental results are shown in Table 3 and 4, where each data is computed based on 1000 benchmarks. Since optimal solutions cannot be found for meshes with more than 81 nodes, the largest benchmark simulated is a 9×9 mesh in Table 3 and 4.

In Table 4, two heuristic methods give more than 99% success ratios for all benchmarks. While in Table 3, the tabu-search based heuristic method outperforms the TK_MK method for all cases. In Table 3, two end-nodes of the longest diagonal line of a mesh are used to be the source and

destination nodes (see Figure 8). The number of hops between these two nodes on a $N \times N$ mesh is at least $2*(N-1)$. Since benchmarks in Table 3 are generated based on the same pair of source and destination nodes, the length of each feasible path found in Table 3 is at least $2*(N-1)$. While in Table 4, the average length of a feasible path is around N. As a result, it is much harder to find a feasible path for the benchmarks used in Table 3.

When the size of a mesh is increased from 49 nodes to 81 nodes, the average success ratio of the tabu-search based method is changed from 94.5% to 92.9%, while the average success ratio of the TK_MK method is decreased from 84.2% to 58.9%. Obviously, the performance of our tabu-search based method is much better and stable than the TK_MK method for $N \times N$ meshes.

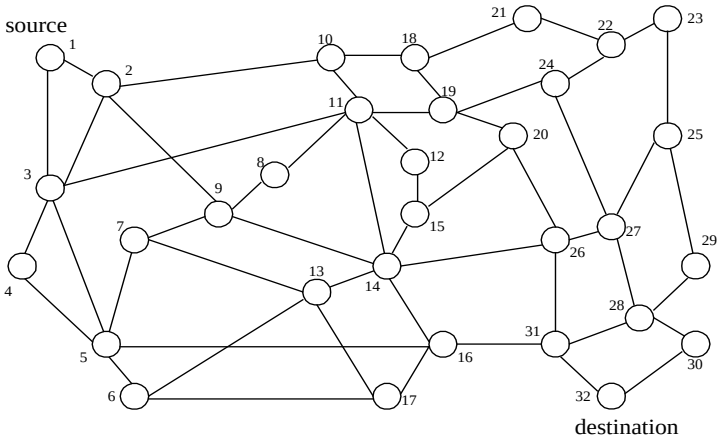

Figure 7. A network topology

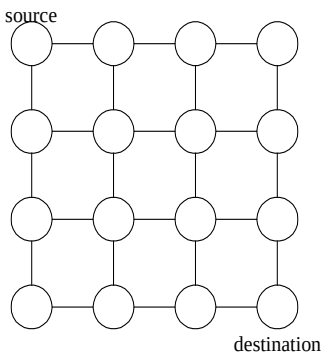

Figure 8. A 16-node mesh

5.3. Efficiency

In order to compare the executing time for two heuristic methods, several large size of meshes are used for experiments. The benchmarks tested in this set of experiments are generated by the method (a) described in section 5.1. The experimental results are shown in Table 5.

Each CPU time in Table 5 is a summation of the executing time of 100 different benchmarks. Several experimental results derived from Table 5 are summarized as follows:

- The optimal solutions cannot be obtained within a reasonable amount of time by the branch-and-bound based optimal algorithm when the mesh size is greater than 81.

- For two heuristic algorithms, the executing time is increased when the values of QoS constraints are decreased. However, for the branch-and-bound algorithm, the executing time is decreased when the values of QoS constraints are decreased. This phenomenon is due to the fact that the number of state nodes, which are eligible for branching in the state space tree constructed in the branch-and-bound algorithm, is increased when the values of QoS constraints are increased. On the contrary, it would become harder for two heuristic algorithms to find a feasible path when the given QoS constraints become tight.
- For small meshes, the executing speed of TK_MK method is faster than the tabu-search based method. However, when the number of nodes of meshes is in the range between 2500 and 10000, our tabu-search based method is much more efficient than the TK_MK method.
- Let α = (# of feasible paths by TK_MK method) / (# of feasible paths by Tabu method) $*100\%$. Since the α values are much less than 1 for most of the benchmarks used in Table 5, the performance of our tabu-search based method is much better than the TK_MK method. In addition, the α value is decreased as the mesh size is increased. For the mesh size is not less than 1600, the probability is very low for the TK_MK method to find a feasible path with at least 78 (i.e., $2*(40-1)$) hops, even the QoS constraints are so loose that the success ratio of the tabu-search based method is 1.
- For a mesh as large as 10000 nodes, it takes only around 4 seconds for the tabu-search based method to find a feasible path with at least 198 hops. Hence, it is a highly efficient method for the MCP problem in a large network.

5.4. The success ratio versus the $L_{\max} / L_{\min}$ for the tabu-search based heuristic algorithm

Based on meshes, the same set of 1000 benchmarks generated using the method (a) described in section 5.1 is used to for two sets of simulations, which are with $L_{\max} / L_{\min} = 10/5$ and $L_{\max} / L_{\min} = 12/7$ respectively. As shown in Table 6, the success ratio is improved when the values of $L_{\max}$ and $L_{\min}$ are increased. However, this improvement is at the expense of consuming more

CPU executing time.

5.5. The success ratio versus the number of QoS constraints

As shown in Figure 9, the success ratio is decreased as the number of constraints is increased for two heuristic algorithms. However, the performance of the TK_MK method affected by the number of constraints is much more serious than the tabu-search based method. This set of simulations is based on 1000 benchmarks generated for an 8×8 mesh using the method (a) described in section 5.1.

6. CONCLUSIONS

Based on the branch-and-bound technique and tabu-searching strategy, an optimal algorithm and a tabu-search based heuristic algorithm are developed in this paper for solving the MCP problem with multiple constraints. The experimental results show that our tabu-search based heuristic algorithm not only outperforms the previous published method in [4], but also is a very efficient approach for solving the MCP problem in large-scale networks.

Table 1. The first set of success ratios of two heuristic algorithms for the ANSNET

- Based on the network in Figure 7, source and destination nodes are set to be node1 and node 32 respectively.
- The number of feasible paths found by all algorithms is based on 1000 benchmarks, which are generated by the method (a) in section 5.1.
- $w_0, w_1 = 0 \sim 10$; $L_{\max} = 10, L_{\min} = 5$.

C_0 / C_1	# of feasible paths found by optimal algorithm	Tabu (success_ratio)	TK_MK (success_ratio)
28/30	916	99.6%	98.9%
28/28	890	99.4%	99.1%
24/28	754	99.5%	99.6%
28/20	534	99.3%	100.0%
24/20	395	98.2%	99.8%
20/20	230	100.0%	100.0%
Average		99.3%	99.6%

Table 2. The second set of success ratios of two heuristic algorithms for the ANSNET

- The number of feasible paths found by all algorithms is based on 1000 benchmarks, which are generated by the method (b) in section 5.1.
- $w_0, w_1 = 0 \sim 10$; $L_{\max} = 10, L_{\min} = 5$.

C_0 / C_1	# of feasible paths found by optimal algorithm	Tabu (success_ratio)	TK_MK (success_ratio)
28/28	989	99.8%	100.0%
20/24	872	99.8%	99.7%
16/16	605	99.3%	100.0%
16/12	463	99.6%	100.0%
10/12	332	99.7%	100.0%
8/10	210	100.0%	100.0%
Average		99.7%	99.9%

Table 3. The first set of success ratios of two heuristic algorithms for $N \times N$ meshes

- The number of feasible paths found by all algorithms is based on 1000 benchmarks, which are generated by the method (a) in section 5.1.
- $w_0, w_1 = 0 \sim 10$; $L_{\max} = 10, L_{\min} = 5$.

Nodes	C_0 / C_1	# of feasible paths found by optimal algorithm	Tabu (success_ratio)	TK_MK (success_ratio)
81	64/56	966	96.6%	69.2%
	56/56	872	93.1%	57.7%
	48/56	606	89.9%	58.4%
	48/48	313	92.0%	50.2%
Average			92.9%	58.9%
64	56/49	940	96.7%	77.2%
	49/49	818	94.3%	66.9%
	42/49	553	93.1%	70.3%
	42/42	289	91.7%	70.9%
Average			94.0%	71.3%
49	48/42	880	96.5%	83.1%
	42/42	712	96.9%	81.3%
	36/42	444	95.0%	84.7%
	36/36	236	89.4%	87.7%
Average			94.5%	84.2%

Table 4. The second set of success ratios of two heuristic algorithms for $N \times N$ meshes

- The number of feasible paths found by all algorithms is based on 1000 benchmarks, which are generated by the method (b) in section 5.1.
- $w_0, w_1 = 0 \sim 10$; $L_{\max} = 10, L_{\min} = 5$.

Nodes	C_0 / C_1	# of feasible paths found by optimal algorithm	Tabu (success_ratio)	TK_MK (success_ratio)
81	48/40	984	99.8%	98.4%
	32/24	663	98.9%	98.3%

	24/24	546	98.9%	99.5%
	16/16	287	99.7%	100.00%
Average			99.3%	99.0%
64	42/42	995	99.9%	98.6%
	28/35	866	99.4%	99.0%
	28/21	621	99.2%	99.7%
	14/14	259	99.6%	100.00%
Average			99.5%	99.3%
49	36/36	984	99.7%	98.9%
	30/24	846	99.5%	99.5%
	18/18	483	99.2%	100.00%
	12/12	227	99.6%	100.00%
Average			99.5%	99.6%

Table 5. Efficiency comparisons based on $N \times N$ meshes

- The number of feasible paths found by all algorithms is based on 100 benchmarks, which are generated by method (a) in section 5.1.
- Each CPU time is a summation of all the executing time of 100 benchmarks.
- The CPU time is marked as "n/a", if it is over 7200 secs.
- The optimal solutions are found by branch-and-bound algorithm.
- α = (# of feasible paths by TK_MK method) / (# of feasible paths by Tabu method) $*100\%$

Nodes	C_0 / C_1	Optimal		Tabu		TK_MK		α
		# of feasible paths	CPU (sec)	# of feasible paths	CPU (sec)	# of feasible paths	CPU (sec)	
81	65/65	100	4522	100	0.4	86	0.2	86.0%
81	60/50	79	1170	72	5.6	53	0.2	73.6%
81	50/50	50	464.1	44	5.2	28	0.1	63.6%
100	70/70	n/a	n/a	100	1.9	73	0.3	73.0%
100	70/55	n/a	n/a	84	17.2	56	0.3	66.7%
100	55/60	n/a	n/a	58	34.7	21	0.2	36.2%
1600	310/280	n/a	n/a	100	13.4	11	68.8	11.0%
1600	250/250	n/a	n/a	90	45.1	0	72.6	0%
1600	220/250	n/a	n/a	55	171.2	0	65.6	0%
2500	400/300	n/a	n/a	100	27.2	4	265.0	4.0%
2500	350/300	n/a	n/a	96	35.4	0	269.4	0%
2500	300/300	n/a	n/a	57	162.5	0	272.0	0%
6400	550/600	n/a	n/a	100	182.9	0	2117.0	0%
6400	550/500	n/a	n/a	92	192.4	0	2384.0	0%
6400	500/500	n/a	n/a	66	212.5	0	2413.0	0%
10000	750/700	n/a	n/a	100	435.2	0	5132.0	0%
10000	700/600	n/a	n/a	92	442.9	0	5166.0	0%
10000	650/650	n/a	n/a	76	449.1	0	5404.0	0%

Table 6. The success ratio versus the $L_{\max} / L_{\min}$ for the tabu-search based heuristic method

- The success ratio and CPU time obtained for the tabu-search based heuristic are based on 1000

benchmarks, which are generated by the method (a) in section 5.1.

- $w_0, w_1 = 0 \sim 10$;

Nodes	C_0 / C_1	$L_{max} / L_{min} = 10/5$		$L_{max} / L_{min} = 12/7$	
		success ratio	CPU (sec)	success ratio	CPU (sec)
81	64/56	96.6%	36.2	97.7%	114.5
	56/56	93.1%	58.9	96.1%	115.6
	48/56	89.9%	46.5	93.4%	116.0
	48/48	92.0%	15.9	93.3%	42.4
Average		92.9%	39.4	95.1%	97.1
64	56/49	96.7%	25.5	97.7%	48.5
	49/49	94.3%	33.1	97.2%	51.1
	42/49	93.1%	22.2	100.0%	44.0
	42/42	91.7%	10.9	95.5%	14.6
Average		94.0%	22.9	97.6%	39.6
49	48/42	96.5%	15.9	98.3%	19.5
	42/42	96.9%	12.5	97.9%	16.2
	36/42	95.0%	9.9	97.1%	11.1
	36/36	89.4%	6.1	94.1%	8.5
Average		94.5%	11.1	96.8%	13.8

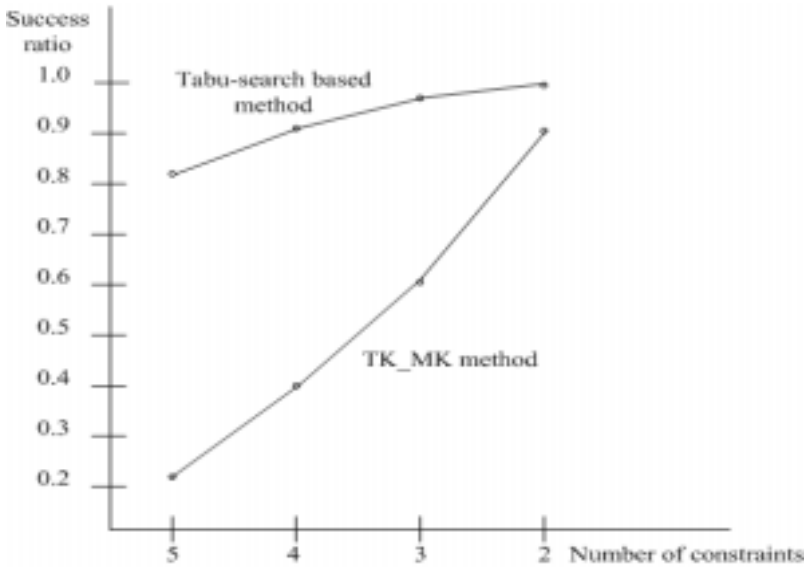

Figure 9. The success ratio versus the number of constraints for an 8×8 mesh

REFERENCES

[1] J. M. Jaffe, “Algorithms for finding paths with multiple constraints,” *Networks*, vol. 14, pp. 95-116, 1984.

[2] Shigang Chen and Klara Nahrstedt, “On Finding Multi-constrained Paths,” in *Proceedings of the ICC’98 Conference*, IEEE, 1998, pp. 874-979.

[3] Xin Yuan and Xingming Liu, “Heuristic Algorithms for Multiconstrained Quality-of-Service Routing,” *IEEE Transaction on Networking*, vol. 10, no. 2, pp. 244-256, April, 2002.

[4] Turgay Korkmaz and Marwan Krunz, “A Randomized Algorithm for Finding a Path Subject to Multiple QoS Constraints,” in *Proceedings of the IEEE Global Telecommunications Conference*, IEEE, 1999, pp. 1694-1698.

[5] R. Widyono, “The Design and Evaluation Algorithm for Real-time Channels,” *TR-94-024*, International Computer Science Institute, UC Berkeley.

[6] Turgay Korkmaz, Marwan Krunz and Spyros Tragoudas, “An efficient algorithm for finding a path subject to two additive constraints,” Computer Communications, vol. 25, pp. 225-238, 2002.

[7] D.E. Comer, *Internetworking with TCP/IP*, vol. I, Prentice Hall, 1995.

[8] Dinkar Sitaram and Asit Dan, *Multimedia Servers*, Morgan Kaufmann Publishers, 2000.

[9] Sanjeev Verma, Rajesh K. Pankaj, and Alberto Leon-Garica, “QoS based multicast routing algorithms for real time applications”, *Performance Evaluation*, vol. 34 pp. 273-294, 1998.

[10] Q. Ma and P. Steenkiste, “Quality-of-Service Routing for Traffic with Performance Guarantees,” *Proceedings of IFIP Fifth International Workshop on Quality of Service*, May 1997.

[11] Ariel Orda, “Routing with End-to-End QoS Guarantees in Broadband Networks,” *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 365-374,1999.

[12] Zheng Wang and Jon Crowcroft, “QoS Routing for Supporting Resource Reservation,” *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228-1234, Sept. 1996.

[13] Wen-Lin Yang, “A Branch-and-Bound Approach for Solving QoS Routing Problem,” Proceedings of 2002 Symposium on Digital Life and Internet Technologies, NCKU, Taiwan, June 2002.