A Heuristic using GOSST with 2 Connecting Strategies for Minimum Construction Cost of Network

†,††*Inbum Kim*, †*Chae-kak Kim*, ††*S. Hossein Hosseini*

†Dep.of Internet Information, Kimpo College, Gimpo-si, Kyonggi-do, South Korea
††Dep.of Electrical Engineering and Computer Science, University of Wisconsin-Milwaukee, Milwaukee, Wisconsin, USA

Summary
The goal of this research is to devise a heuristic to solve the GOSST(Grade of Services Steiner Minimum Tree) problem that could apply to the design of communication networks. GOSST problem is to find a network topology satisfying the *G-condition* with minimum construction cost. The proposed heuristic might provide a way to design of more economical network offering differential grade of services. The heuristic employs Minimum Spanning Tree, Steiner Point and two connection strategies. The implemented methods will be analyzed their performance and characteristics for examining the heuristic. Because GOSST problem is known to be NP-Complete, proposed heuristic to find a reasonable solution might have some limitation essentially. Most researches on GOSST have concentrated on geometric analyses and approximation algorithms. Since all published solutions for optimization problems require tremendous computation and memory space, it might be hard to apply those to practical problems. Therefore GOSST, one of the optimization problems, could not escape the foible. In this research, a feasible heuristic for GOSST problems is proposed, implemented and analyzed. For more exquisite designed heuristic, more studies and trials are necessary.

Key words:
GOSST, Steiner Point, Minimum Spanning Tree, Network Construction Cost, Global Connecting, Local Connecting G-condition

1. Introduction

In the network design, there is a fundamental issue for the physical construction of a network structure, which is to save the construction costs with sufficient transmission capabilities; the interconnection of many communication sites with the best choice of the connecting lines and with the best allocation of the transmission capacities over these lines. Good solutions always yield paths with enough communication capacities between any two sites, spending less network construction costs. A solution for the GOSST (Grade of Services Steiner Minimum Tree) problem could make the good use of the applications like this. GOSST problem also has applications in transportation, for road constructions and in some more potential uses of CAD in terms of interconnecting the elements on a plane such that to provide enough flux between any two elements.

GOSST problem is a variation of the ESMT (Euclidean Steiner Minimum Tree) which is a problem to find a minimum cost network interconnecting a set of given points in the Euclidean planes. The works for ESMT problem could be found in [11,12,13,14,15,16,17]. The GOSST problem is known to be NP-Complete. Therefore, to solve even small-scale problems needs tremendous computations and memory spaces. The previous many researches on GOSST have interested in geometric analysis and improvements of approximation algorithms. But the heuristics for GOSST have not been published lively.

For each edge in an edge set **E**, weight ***w*** between two nodes is specified by the cost for communicating with each other. There is a case to find acyclic subset that connects all of the vertices and whose total weight sum is minimized. Since the subset is acyclic and connects all vertices, it forms a tree. The tree grows until the tree spans all vertices and therefore, it is called as a spanning tree. The problem of determining such a tree is called as a minimum spanning tree problem. There are two dominant algorithms for solving minimum spanning tree problem; Kruskal's algorithm and Prim's algorithm. In this research, the Prim's algorithm is adopted to build a Minimum Spanning Tree for the heuristic of GOSST problem.

There is a problem to find the point **P** that minimizes the sum of the distance from **P** to each of three given points in the plane or to find the point **P** in a triangle so that the total distance from **P** to each of the triangle's

Manuscript received December 5, 2006.
Manuscript revised December 25, 2006.

vertices is minimized. This problem can be expended even further by allowing the addition of an arbitrarily number of points to find the shortest network connecting all points. Adding each point called Steiner Point and producing a tree to create the minimal network is the Minimum Steiner Tree problem. In this research, this Steiner point and Minimum Steiner Tree are employed to implement the heuristic for the GOSST problems.

This research is to propose a heuristic for GOSST problems with analyzing the experiment results of some examples and with investigating some potentiality to apply the heuristic to some practical applications of real world.

2. Problem Definition

The GOSST problem [1] can be defined as follows: Let $\mathbf{P}=\{p_1, p_2, \ldots, p_n\}$ be a set of n terminal points in the Euclidean plane, where each point p_i has a service request of grade as $grade(p_i) \in \{1, 2, \ldots, r\}$. Let $0 < c(1) < c(2) < \ldots < c(r)$ be r real numbers, where $c(\boldsymbol{i})$ is the cost for providing service $\boldsymbol{i}$. Each edge in the network might be assigned a specific grade of service, which is a number in $\{1, 2, \ldots, r\}$. Let $grade(e)$ denote the service grade of edge e. The GOSST problem asks for a minimum cost network interconnecting terminal points in set P and some Steiner Points. The Steiner points have 0 service request of grade. Between each pair of terminal points p_i and p_j in a network, there is a path whose minimum grade of service is at least as large as $\min(grade(p_i), grade(p_j))$ and the construction cost of the network is minimum. The construction cost of an edge with service of grade $\boldsymbol{g}$ in the network is the product of the Euclidean length of the edge by $c(\boldsymbol{g})$. The GOSST problem is a generalization of the ESMT problem where all terminal points have the same service request of grade.

For the GOSST problem, the concurrent combination of the two factors determines target network: the allocation of the service of grade for every edge, and the choice of the Steiner points. But the service of grade could not determined before the choice of the Steiner points, while the reverse could not, either. Most studies have done for the simplified special cases of the GOSST problem: the cases are that the grade of services request is either 2 or 3, or limited by some conditions. [2, 3, 4] The general case approximation algorithm by Michandani shows the performance ratio is $\gamma\rho+1$, where ρ is the best performance ratio of a Steiner tree heuristic and γ is the number of different grades of services request. [18]

3. Background

Fig 1 provides an example network with 6 terminal nodes and 8 edges, where each node has its location on the Euclidian plane and the capacity. The capacity might be considered as processing ability or computing power. This capacity of a node could be regarded as the cost for certain service grade of the node and the capacity of an edge as the cost for a service grade of the edge in GOSST problem. Table 1 represents the information of 6 terminal nodes and table 2 does that of edges of example network.

Fig 2 offers the lengths and the capacities of edges of example network. The length of an edge comes from the calculated distance of two nodes and the capacity of an edge derives from the minimum capacity of the edge constituent two nodes. The information of edges forming the network is in Table 2, where the cost field is the construction cost of an edge, which results from the production of edge length by its capacity.

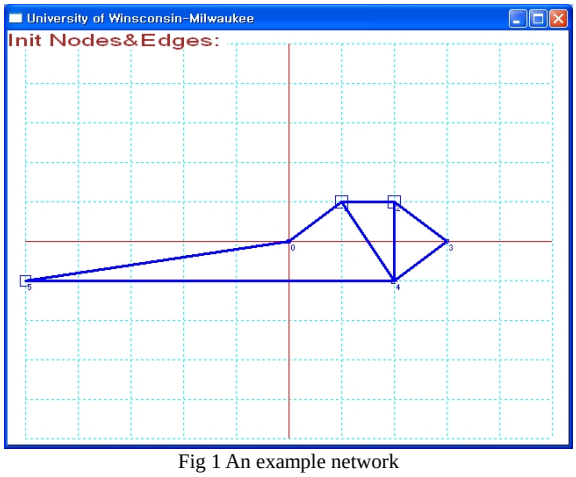


Fig 1 An example network

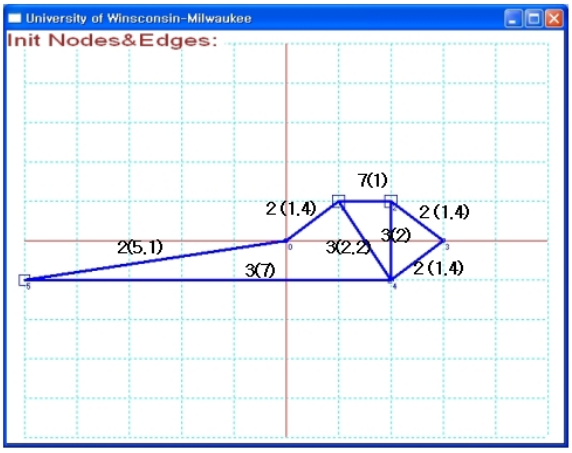


Fig 2 Capacities and lengths of the edges in example. The lengths of the edges are written in a pair of parenthesis.

Node Name	Location (x, y)	Capacity
0	(0, 0)	2
1	(1, 1)	7
2	(2, 1)	7
3	(3, 0)	2
4	(2,-1)	3
5	(-5, -1)	6

Table 1 Locations and capacities of terminal nodes in example network

Edge Name	Constituent Nodes & Their Capacity	Length (A)	Capacity (B)	Cost= A×B
a	0(2), 1(7)	1.4	2	2.8
b	1(7), 2(7)	1	7	7
c	2(7), 3(2)	1.4	2	2.8
d	2(7), 4(3)	2	3	6
e	3(2), 4(3)	1.4	2	2.8
f	4(3), 5(6)	7	3	21
g	5(6), 0(2)	5.1	2	10.2
h	1(7), 4(3)	2.2	3	6.6

Table 2 Lengths, capacities and costs of given edges

3.1 Minimum Spanning Tree Problem

For interconnecting of a set of *n* nodes, a selection of *n*-1 edges can be used; the one using the least amount of edges is usually craved for. With **V**, a set of nodes and **E**, a set of possible interconnections between pairs of nodes, a connected, undirected graph **G** can be described as **G**=(**V**, **E**). For each edge $(u, v) \in \mathbf{E}$, weight $\boldsymbol{w}(u, v)$ is defined by the cost for connecting *u* and *v*. MST(Minimum Spanning Tree) problem is to find acyclic subset $\mathbf{T} \subseteq \mathbf{E}$ that connects all of the vertices with minimum total weights. As **T** is an acyclic tree and connects all vertices, it is called a spanning tree. A minimum spanning tree problem is to look for the tree like **T**.

There are two prominent algorithms for a minimum spanning tree; Kruskal's algorithm and Prim's algorithm. In Kruskal's algorithm, the set **A** is a forest and an edge added to **A** should be always a minimum weight edge. The tree connects to a vertex not belong to the tree at that time and only one edge is added at a time in each iteration for growing the minimum spanning tree. A set of edges **A** is a subset of final minimum spanning tree during the iterations.

Prim's algorithm operates much like Dijksra's algorithm for finding shortest paths in a graph. In the algorithm, the edges in the set **A** always form a single tree. The tree starts from an arbitrary root vertex ***r*** called first vertex and grows until the tree spans all the vertices in **V**. At each step, a least weighted edge is added to the tree **A** for connecting **A** to an isolated vertex. This is greedy strategy, since added edge is the one that contributes the minimum amount possible to the tree weight at one time. The key point of Prim's algorithm is to make it easy to select an edge added to the tree.

Table 3 shows the Prim's minimum spanning tree algorithm. The connected graph **G,** first vertex ***r*** and weights ***w*** are input parameters of the algorithm. During execution, all vertices not connecting to the tree reside in a min-priority queue **Q**. For each vertex **v**, *key*[**v**] is the minimum weight of any edge connecting **v** to a vertex in the tree, and *key*[v]=∞ represents there is no such edges. π[**v**] is the parent of **v** in the tree. When algorithm terminates, the min-priority queue ***Q*** becomes empty and minimum spanning tree **A** is constructed finally.

```
MST-PRIM(G,w,r)
1.   FOR each u ∈ V[G]
2.     DO key[u] ← ∞
3.        π[u] ← NIL
4.   key[r] ← 0
5.   Q ← V[G]
6.   WHILE Q ≠ ϕ
7.     DO u ← EXTRACT-MIN(Q)
8.        FOR each v ∈ Adj[u]
9.           DO if v ∈ Q and w(u,v) < key[v]
10.             then π[v] ← u
11.                  key[v] ← w(u,v)
12.  END WHILE
```

Table 3 Prim's Algorithm for MST

The performance of Prim's algorithm depends on how the min-priority queue ***Q*** is implemented. If ***Q*** employs binary min-heap, the total time of the algorithm is O(**E***log***V**). With Fibonacci heap for ***Q***, the running time of the algorithm can be improved to O(**E**+**V***log***V**) if |**V**| is much smaller than |**E**|.[10]

In this research, cost preferring MST is used, which employs cost as weight, where cost is the production edge length by its capacity. Fig 3 shows the Cost Preferring MST by the following steps with given network.

3.2 Steiner Minimum Tree

Minimizing a network's length has been one of the important optimization problems. One of the problems is to find the point **P** that minimizes the sum of the distance from **P** to each of three given points in the plane or to find the point **P** in a triangle so that the total distance from **P** to the triangle's vertices is minimized.

The general solution is that either **P** is inside of the triangle formed by the points and the angles formed by the lines connecting **P** to each of the points are all 120°, or **P** is one of the three vertices and the angle formed by

connecting **P** to the other vertices of the triangle is greater than or equal to 120°. A mathematician ***Jakob Steiner*** made researched in this problem and expanded it to include an arbitrarily large set of points in the plane. As this involved only one point, forming a star-like shape, it is called Steiner star, when **P** was joined to each other of the points.

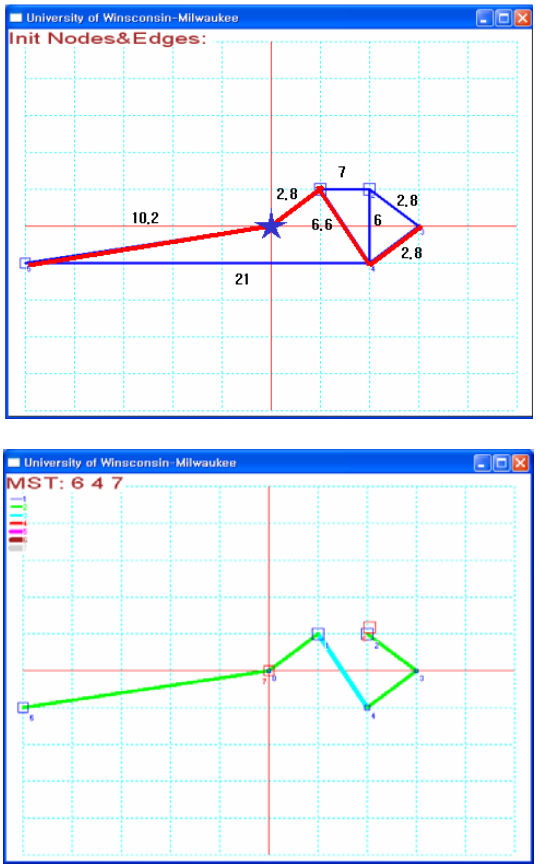


Fig 3 Calculated each edge weight (cost =Capacity × Distance) of example network (Up) and final Cost Preferring Minimum Spanning Tree whose first vertex is point 0 (Bottom**).**

This problem was expended even further by allowing the addition of an arbitrarily number of points to find the shortest network connecting all points. ***Courant*** and ***Robbins*** added the points called Steiner Point to create the minimal network and produced a tree as the final result instead of Steiner star.

Much is known about the mathematical structure of SMT (Steiner Minimum Tree). Some basic properties of the SMT are;

- All of the original *n* points will be connected 1, 2, or 3 other points
- All Steiner Points will be connected to 3 other points
- Any two edges meet at an angle of at least 120°
- At most *n*-2 Steiner Points need to be added to the network
- The length of an SMT will never be any shorter than √3/2 times the length of a Minimal Spanning tree

This problem is NP-Hard or NP-Complete when discrete points are used. Therefore, the problem cannot be solved in polynomial time.

Fig 4 shows a method to determine a candidate Steiner point **P**. To find point **P** for point **A**, **B** and **C**, an equilateral triangle **AQB** is built, using edge **AB,** the longest edge of the triangle formed by the three original points A, B and C so that point **C** lies outside of the newly formed equilateral triangle **AQB**. A circle about the equilateral triangle **AQB** is circumscribed, and a line from point **C** to the vertex **Q** on the equilateral triangle is made. The point on which the line and the circle meet is **P**, which is a candidate Steiner point.

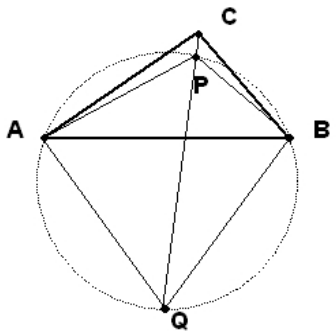


Fig 4 A method to find the candidate Steiner Point P with terminal node A, B and C

But this method doesn't always yield the real Steiner point. Fig 5 gives an example that point **P** achieved by this way could not be real Steiner point; the spanning tree length (7.99) is shorter than the Steiner length (9.51), where Spanning length is defined as the minimum value among the 3 summations; length (**AB**)+length (**BC**), length (**BC**)+length (**CA**), length (**AB**)+length (**CA**). The Steiner length is the summation; length (**AP**)+length (**BP**)+length (**CP**). This is the case that determined point **P** lies outside of the original triangle ABC.

Fig 6 shows an example such as Steiner length (11.25) is equal to the Spanning length (11.25) and this is the case

that the acquired point **P** is on one of the edges forming the original triangle **ABC**. Fig 7 presents an example Steiner length (12.21) is shorter than Spanning length (13.66) and this is the case that the yielded point **P** is inside of the original triangle **ABC**. The point **P** could be called as a real Steiner Point.

In this research, the method for gaining point **P** is applied practically to proposed heuristic for GOSST as follows; if Steiner length is shorter than Spanning length, the calculated point **P** is selected as a Steiner point, else the middle point of Spanning tree is considered as Steiner point. Here, the middle point of Spanning tree is such as point **X** if Spanning tree is formed by edge **WX** and edge **YX**. By this method, the extended Steiner length might be less or equal to the Spanning length.

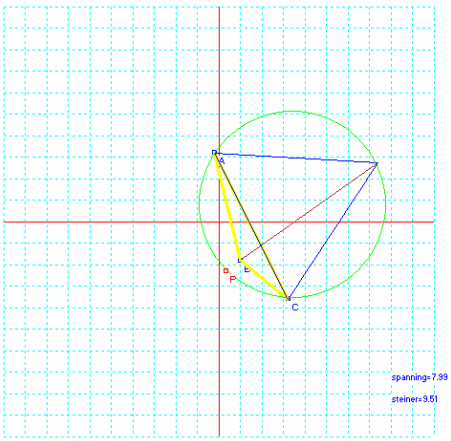


Fig 5 Steiner length (=9.51) is longer than Spanning length (=7.99). The point P lies outside of the original triangle ABC

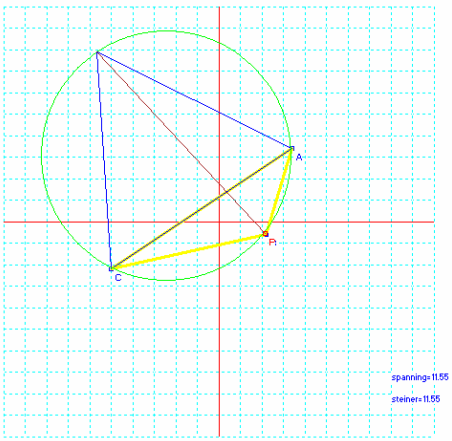


Fig 6 Steiner length (=11.55) is equal to Spanning length (=11.55). The point P lies on one of three edges of the triangle ABC

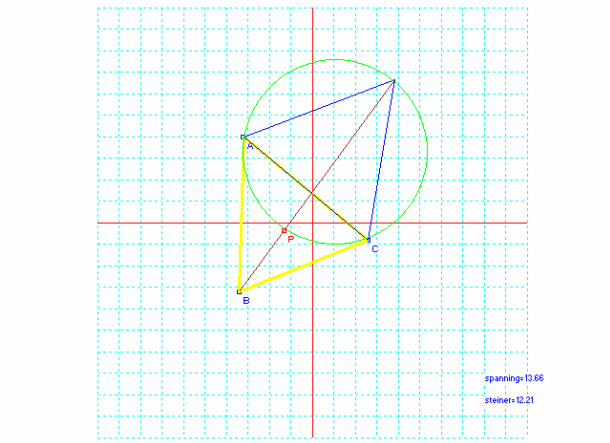


Fig 7 Steiner length (=12.21) is shorter than Spanning length (=13.66). The point P lies inside of the triangle ABC

3.3 Grade Of Services Steiner minimum Tree

The GOSST (Grade Of Services Steiner minimum Tree) problem asks for a network interconnecting the point set P and some Steiner points such as;

- Between each pair of terminal point p_i and p_j, there is a path whose minimum grade of service is at least as large as min(*grade*(p_i), *grade*(p_j)), which is called as *G-condition* in this research
- The construction cost of the network is minimum among all interconnecting networks satisfying the *G-condition*, where the cost of an edge with service of grade *u* is the product of the Euclidean edge length by the expenditure for service grade *u*.

The minimum construction cost network is determined by the combination of two factors in the GOSST problem; the allocation of service grade for each edge and the determining Steiner points. But, both factors cannot be determined one before another.

In Fig 8, there are three terminal nodes **A**, **B** and **C** having different processing capacities (or service grades) respectively. Though node **A**'s processing capacity is 2, the edge capacity from node **A** to node **B** could not be more than 1 as node **B**'s processing capacity is 1. The capacity of path from **B** to **C** or from **A** to **C** via **B** could not be more than 1, either. If the edge capacity is regarded as bandwidth, the bandwidth waste of this network is inevitable.

To avoid this dissipation or to satisfy *G-condition*

described above, this research proposes that new created Steiner Point connects to terminal node **A**, **B** and **C** as Fig 9. In this technique, the possible capacity of path from **A** to **C** increases to 2, though the sum of edge length might be increased. The determination of the Steiner point position is a key to reduce the sum of edge length. In this research, the Steiner point positions are also considered. To minimize the network construction cost and to satisfy the *G-condition* for all paths of pairs of terminal node concurrently is a goal of GOSST problem.

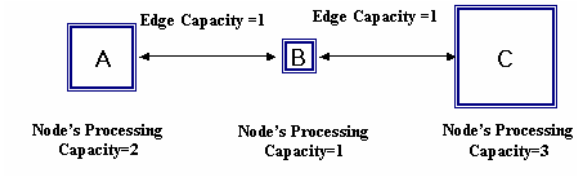


Fig 8 Three terminal nodes A, B, and C having their own different processing capacities want to communicate with other nodes.

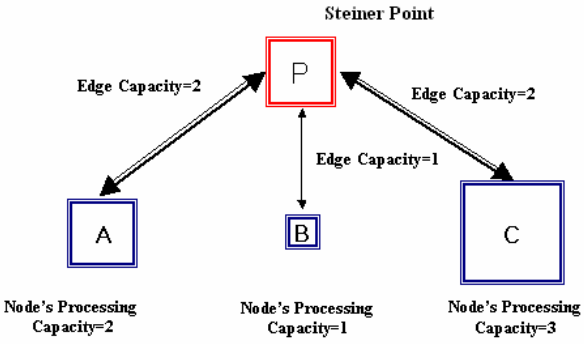


Fig 9 One Steiner point is created and connects to terminal nodes A, B and C to satisfy the *G-condition*.

4. Proposed Heuristic

In this research, a heuristic for the GOSST problem is devised. The original problem does not provide edge information; that is to find a GOSST with only given set of terminal nodes having own capacity. But in this research, to generalize the original GOSST problem, an edge set is also provided as input.

Proposed heuristic could be classified according to applied connecting strategies. Two connecting strategies are global connecting and local connecting.

4.1 Global Connecting Method

Fig 10 shows the global connecting strategy.

A *Steiner point* is created, if a path from *start* node to *end* node violates *G-condition*. And it connects to *start* node, *end* node and *g-node*, where *g-node* is a selected node on the path from *start* and *end* node. To guarantee loop-free, no necessary existing edges are removed.

4.2 Local Connecting Method

In fig 11, the local connecting strategy could be found.

A *G-condition* violation sub path between *fparent* node and *bparent* node on the path between *start* and *end* is investigated. If a violation is found, a *Steiner point* is created and connects to *fparent*, *bparent* and *g-node*, where *g-node* is a selected node being on the section between *fchild* and *bchild* node. For loop-free, unnecessary edges are swept away.

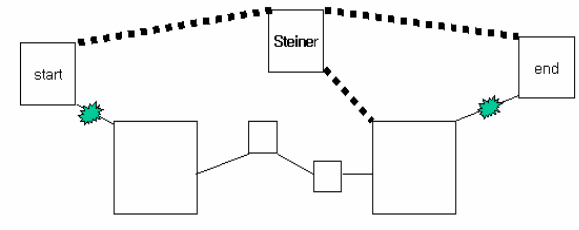


Fig 10 Global Connecting Strategy

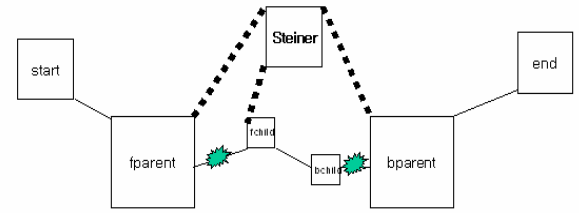


Fig 11 Local Connecting Strategy

4.3 Cost Preferring Global GOSST

Fig 12 presents the result of **Cost Preferring Global GOSST** method with terminal nodes and edges of example network shown Fig 1. The processes are as follows;

Step 1 Make a **Cost Preferring Minimum Spanning Tree** for given terminal nodes and edges with productions of their length by

capacities as shown Fig 3.

Step 2 Check the *G-condition* for each path of all pairs of given terminal nodes.

Step 3 If a *G-condition* is violated on a path from *start* and *end*,

Step 3-1 Create a candidate Steiner point **P**.

Step 3-2 Connect P to *start* and *end* node.

Step 3-3 Determine the point **P**'s capacity as minimum of the capacities of *start* and *end* node.

Step 4 Check whether the created point **P** has the qualification for real Steiner Point.

Step 4-1 If it has the qualification, connect the Steiner point **P** to *g-node*, which is located in front of *end* node on the path from *start* to *end* node and remove unnecessary edges.

Step 4-2 If **not**, throw the point **P** and new Steiner points are created on the intermediate nodes lying on the path between *start* node and *end* node. The capacities of the Steiner points are determined as minimum of the capacities of *start* and *end* node. Remove the connections between the intermediate nodes and rebuild the connections between created Steiner points and make new connections between each intermediate node and Steiner Point locating on it.

Step 5 Rebuild new **Cost Preferring Minimum Spanning Tree** with changed network components.

Step 6 Repeat from step 2 to 5 until there is no any *G-condition* violation on each path of all pairs of terminal nodes

4.4 Cost Preferring Local GOSST

Fig 13 discloses the result of **Cost Preferring Local GOSST** method with given terminal nodes and edges shown Fig 1. The processes are as follows;

Step 1 Make a **Cost Preferring Minimum Spanning Tree** for given terminal nodes and edges with the production of their lengths by capacities as shown Fig 3.

Step 2 Check the *G-condition* for each path of all pairs of given terminal nodes.

Step 3 If *G-condition* is violated on a path from *start* to *end*,

Step 3-1 Investigate the sub-path from *fparent* to *fchild* which violates forward *G-condition* first on the path from *start* to *end* and the sub-path from *bparent* to *bchild* which violates backward *G-condition* first on the path from *end* to *start* at first.

Step 3-2 Find *g-node* locating on the path from *fchild* to *bchild* and having the shortest distance from *fparent* and *bparent*.

Step 3-3 Create a candidate Steiner point **P** and assign its capacity as minimum of the capacities of *start* and *end* node.

Step 4 Check whether the created point **P** has the qualification for real Steiner Point.

Step 4-1 If it has the qualification, connect the created Steiner point **P** to the following nodes; fparent, *bparent* and *g-node* respectively. And remove unnecessary connections.

Step 4-2 If **not**, cast away the point **P** and new Steiner points are created on the intermediate nodes lying on the path between *fparent* and *bparent*. The capacities of the Steiner points are determined as minimum of the capacities of *start* and *end* node. Remove the connections between the intermediate nodes and rebuild the connections between created Steiner points and make new connections between each intermediate node and Steiner Point locating on it

Step 5 Rebuild new **Cost Preferring Minimum Spanning Tree** with changed network components.

Step 6 Repeat step 2-5 until there is no *G-condition* violation for every pair of nodes on the all path.

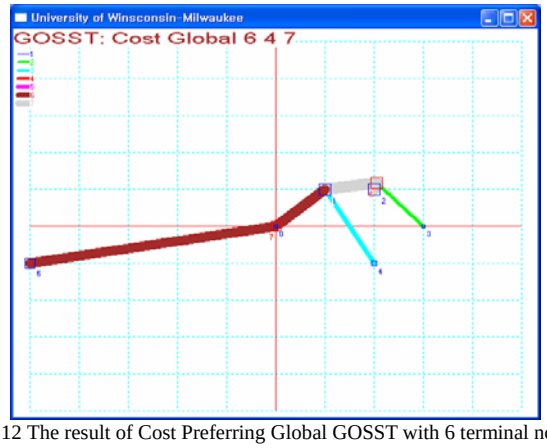


Fig 12 The result of Cost Preferring Global GOSST with 6 terminal nodes,

4 max connections per node and 7 capacity kinds.

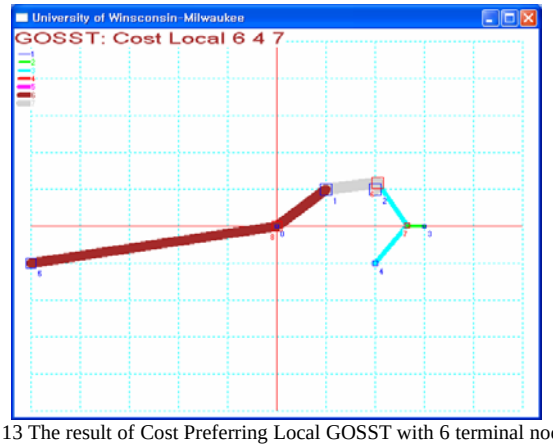


Fig 13 The result of Cost Preferring Local GOSST with 6 terminal nodes, 4 max connections per node and 7 capacity kinds

5. Experiments and Analysis

The input file contains the given network information; the number of terminal node, max connections per node, the number of capacity kinds, node and edge information. The node information consists of its name, 2-dimensional *x*, *y* position and its capacity. The module for producing a MST based on Prim's algorithm was described in Table 3.

New Steiner points might be created whenever the *G-condition* is not satisfied. Removing process of an unnecessary established connections for loop-free and reconstructing new MST module are carried out after Steiner Point and new connections are rebuilt.

Fig 14 represents the input terminal nodes and edges. Each node is described by rectangle and its capacity is by the rectangle size. The basic factors of these experiments are as follows; node number is 100, max connections per node is 3, capacity kinds is 4, and applied methods are Cost Preferring Global GOSST and Cost Preferring Local GOSST. The experiments are conducted on Intel Pentium4 (M) 1.83 GH, 1 G RAM Laptop and implemented with Microsoft Visual C++.

In fig 15, there are the results of Minimum Spanning Tree and GOSST employing by two connecting methods respectively.

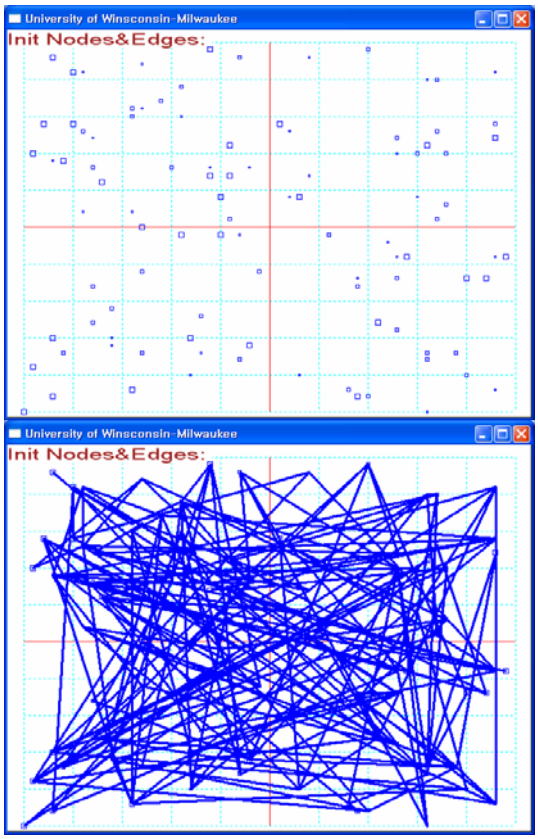
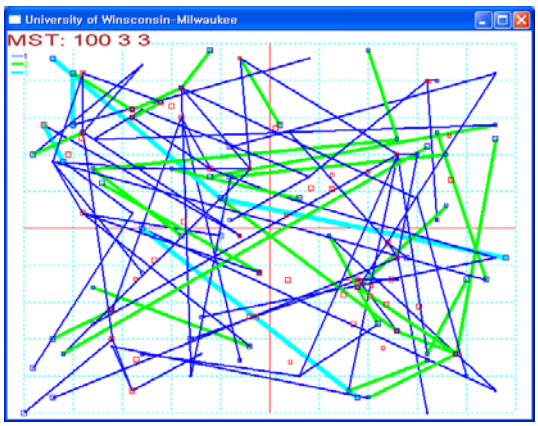


Fig 14 The locations and capacities of 100 terminal nodes are indicated by the position and the size of rectangles (Top) and input connections are by edges (Bottom)



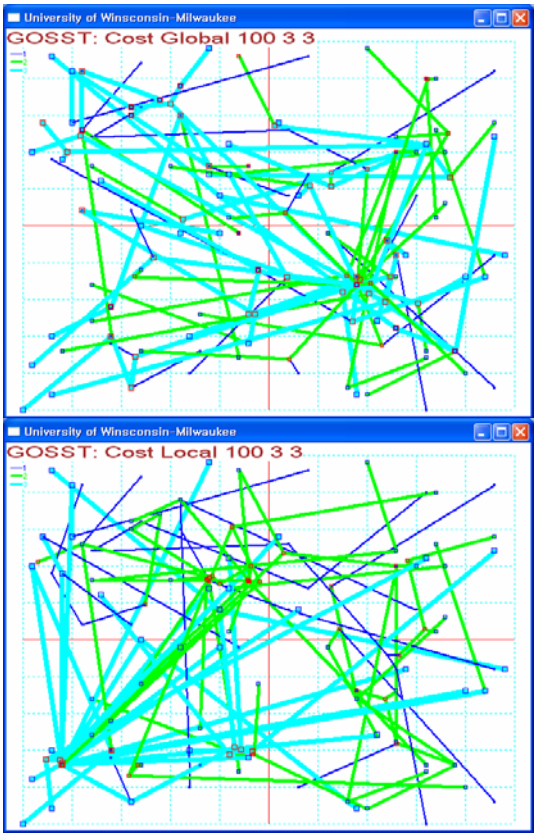


Fig 15 Minimum Spanning Tree (Top) and Grade of Services Steiner minimum Trees built by Cost Preferring Global method (Middle) and by Cost Preferring Local method (Bottom) with 100 terminal nodes, 3 max connections per node and 3 capacity kinds

5.1 Experiments of Terminal Node Number Changing

Fig 16 reveals the comparing the network construction costs of proposed two methods and MST by works of changing the number of terminal nodes. In proportion to the number of terminal nodes, the costs also increase, but the proposed two methods always require fewer costs than MST. Though at some case, global connection method needs more cost than local connection, it shows more cost saving ratio than local connection in overall considering.

According to fig 17 results, the Steiner point number and execution time increase in proportion to the number of terminal nodes. Local connection method requires more Steiner points and execution time for building GOSST.

In fig 18, global connection method shows shortest network length among two connecting methods and MST, and local connection method dose shorter network than MST. In network capacity, the method employing local connection strategy keeps more than global connection.

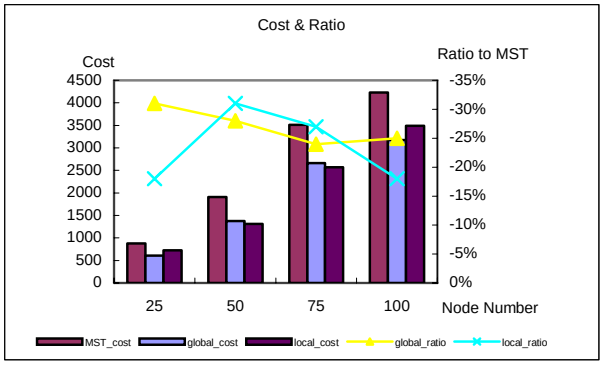


Fig 16 GOSST and MST construction costs and the cost ratios to MST with 100 terminal nodes, 3 max connections per node and 3 capacity kinds

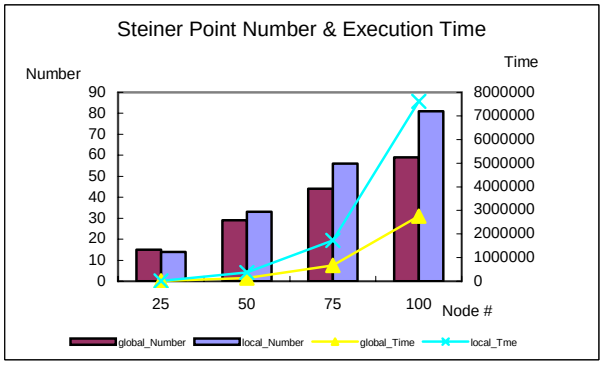


Fig 17 Execution times and created Steiner point number of two connecting methods with 100 terminal nodes, 3 max connections per node and 3 capacity kinds

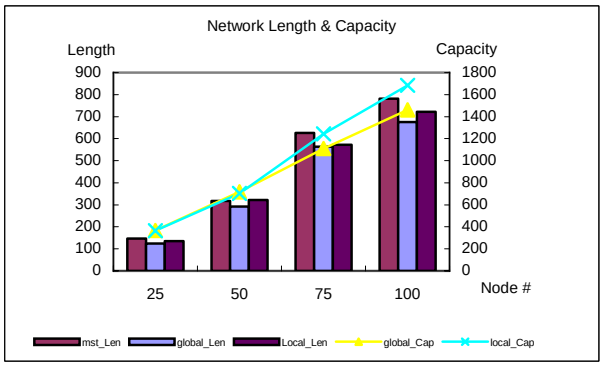


Fig 18 Network lengths and capacities of two GOSST methods and MST with 100 terminal nodes, 3 max connections per node and 3 capacity kinds

5.2 Experiments of the Connection Number per Node Changing

Fig 19 provides the comparing the network construction costs of two proposed methods and MST by works of changing the number of connections per node. Increasing the number of connection per node makes the construction costs of two proposed methods and MST decrease, but when the number is smaller, the global connection method shows more cost saving.

According to the result of fig 20, the Steiner point number and execution time aren't specially associated with the number of connections per node. But in execution time, local connection displays more fluctuation than global connecting. At every connection number per node, global connection method requires less Steiner points and execution time for building GOSST.

In fig 21, in proportion to the number of connections per node, the network lengths of two connecting method and MST become diminish. But the network by local connection method has longest length among two connecting methods and MST and at some case global connection shows longer length than MST. In network capacity experiment, in many times the method employing local connection strategy retains more capacities than global connection.

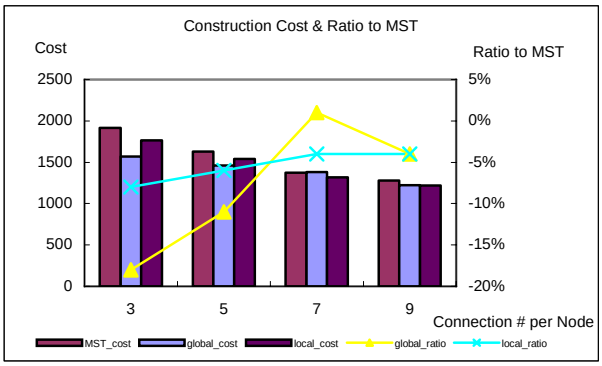


Fig 19 GOSST and MST construction costs and the cost ratios to MST with 100 terminal nodes, 3 max connections per node and 3 capacity kinds

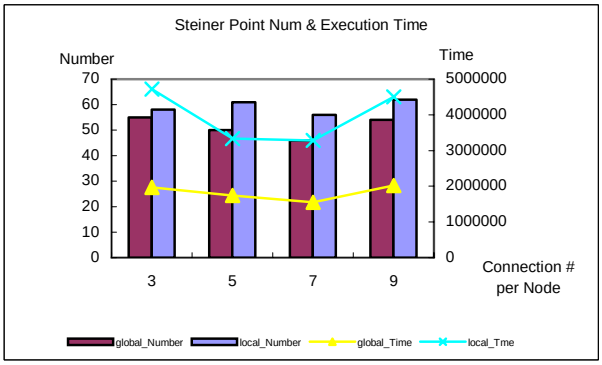


Fig 20 Execution times and created Steiner point number of 2 connecting methods with 100 terminal nodes, 3 max connections per node and 3 capacity kinds

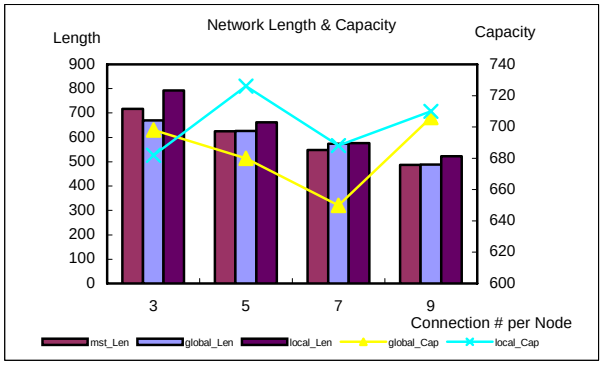


Fig 21 Network lengths and capacities of two connecting methods and MST with 100 terminal nodes, 3 max connections per node and 3 capacity kinds

5.3 Experiments of Capacity Kind Number Changing

In fig 22, the comparison of the network construction costs of two proposed methods and MST is found. Increasing the number of capacity kinds also make the costs of them increase, but the proposed two methods always require less cost than MST. At all capacity kinds, global connection method needs fewer costs and shows more cost saving ratio than local connecting.

According to the result of fig 23, the Steiner point number and execution time of two connecting methods increase in proportion to the number of capacity kinds but the number is 5. Local connection method requires more created Steiner points and execution time for building GOSST and shows sharper variations than global connection.

In fig 24, the special relationship between the network length and the number of capacity kinds could not be exhibited, but global connection method maintains shortest network length, and local connection method dose shorter length than MST when the number of capacity kinds is more than 3. In network capacity, two connecting methods are in direct proportion to the number of capacity kinds. Especially the method employing local connection strategy possesses more network capacity and its slop shows steeper than global connection.

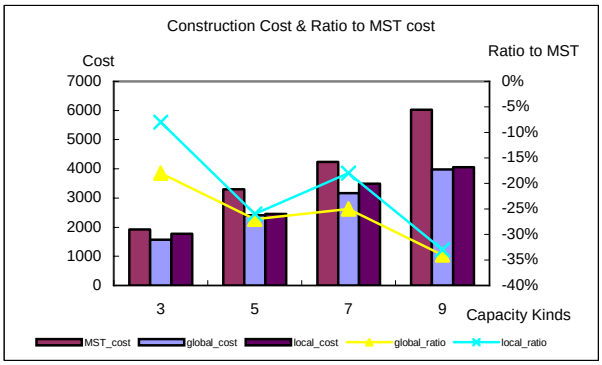


Fig 22 GOSST and MST construction costs and the cost ratios to MST of 100 terminal nodes, 3 max connections per node and 3 capacity kinds

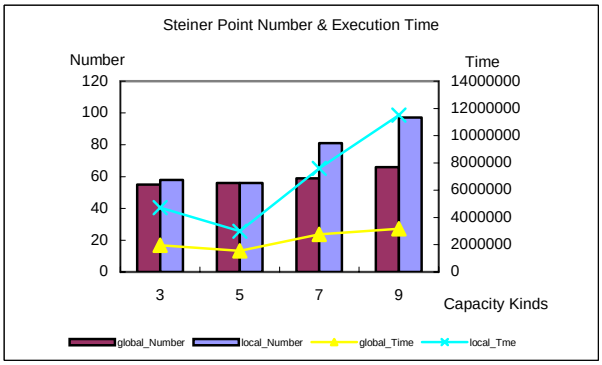


Fig 23 Execution times and created Steiner point number of two connecting methods with 100 terminal nodes, 3 max connections per node and 3 capacity kinds

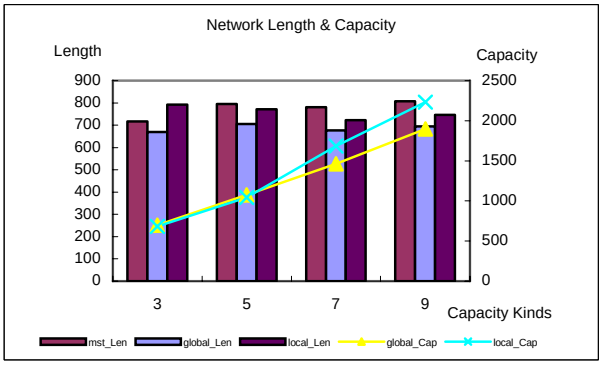


Fig 24 Network Lengths and capacities of two methods and MST with 100 terminal nodes, 3 max connections per node and 3 capacity kinds

6. Conclusions

The GOSST could apply to some useful fields in real world. In network design, the interconnection of many communication sites with the best choice of the connecting line and with the best allocation of the transmission capacities over those is one of many useful applications. Good solutions for the problem might provide paths to targets with sufficient communication capacities through the less network construction costs. This is a goal of GOSST.

The GOSST problem is known to NP-Complete, so its solution could not be acquired in poly-nominal times. Many previous researches on GOSST have focused on the geometric analysis and approximation algorithm improvements. Though such approximation algorithms are elevated little by little, but enormous amount of computation or memory spaces should be sacrificed. The most approximation algorithms for GOSST might be nearly useless in practical applications. The proposed heuristic of this research is expected to provide quite effective solution and to answer the GOSST problems with fairly large number of points quickly.

The basic processes of heuristic are as follows; Minimum Spanning tree (MST) can be achieved through relatively easy ways such as Prim's algorithm. MST could guarantee the minimum edge length sum for forming a network, but yield MST could not be said always to gratify the *G-condition*. Therefore the satisfaction of *G condition* of each path of all pairs of nodes on MST is examined and if a violation occurs somewhere on a path at one time, a candidate Steiner point is created at once and its qualification for real Steiner point is investigated. If it has, the point connects to related points and needless existing connections are removed. If it has not, the point is thrown away, and new Steiner points are created on the interim node of the path or sub-path. Appropriate connecting and removing are conducted Next new MST is rebuilt with the changed network components. These processes are iterative until *G-condition* is satisfied for every path of all pairs of given terminal nodes.

In this research, two connecting varieties of a heuristic are proposed for approaching the GOSST goal. Among them, Cost Preferring Global GOSST method shows more positive results in network construction cost, which is the most important factor of GOSST problem. In addition, the Cost Preferring Global GOSST method makes good results in the required Steiner point number, execution time, cost saving ratio and the length of built network. The Local Connecting strategy shows fair characteristic in network capacity.

The experiments for the heuristic said many important characteristics. In proportion to the number of terminal node, the network construction cost, required Steiner point number, network length and network capacity increase. The increasing the number of connections per node results in the reduction of network length and construction cost. Many capacity kinds make the GOSST augment in construction cost, required Steiner point , execution time and network capacity.

While two connecting method of the heuristic for GOSST problem are proposed, there are still remaining some works. First of all, the improving the heuristic is more necessary. Less network construction cost, less execution times and required Steiner Points, shorter network length, and more network capacity are the goals of elaborate heuristic. More studies on the location of *g-node* are needed. *g-node* is a node to which created Steiner point connects. In this research, the *g-node* is the previous node of *end* node on the path from *start* and *end* node in Global connection strategy or nearest node from *parent* or *child* nodes on the sub-path between *parent* and *child*, which is a partial path from *start* to *end* in Local connection strategy.

Second the more useful applications with GOSST should be also searched or devised. Various types of network construction application and qualification of services could be applied by this heuristic. The applications of constructing networks for maximum networks capacity and/or for minimum networks length, with given terminal nodes and their capacities could be considered as well.

Third more analysis and evaluation of proposed heuristic might be necessary. Works like those would be helpful to make the heuristic more efficiency and exquisiteness.

References

[1] Xue, G.L., Lin, G.H. and Du, D.z. (2001), Grade of Service Steiner Minimum Trees in Euclidean Plane, Algorithmica, 31, 479-500.

[2] Duin, C. and Volgenant, A. (1991), The Multi-weighted Steiner Tree problem, Annals of Operations Research 33, 451-469

[3] Balakrishnan, A., Nagnanti, T.L. and Mirchandani, P. (1994), Modeling and Heuristic Worst Case Performance Analysis of the two level Network Design Problem, Management Science 40, 846-867

[4] Current, J.R., Revelle, C.S. and Cohon,J.L.(1986), The hierarchical network design problem, European Journal of Operational Research 27, 57-66.

[5] J.Kim, M.Cardei, I.Cardei and X.Jia, A polynomial Time Approximation Scheme for the Grade of Service Steiner Minimum Tree Problem

[6] Arora, S.(1996), Polynomial-Time Approximation Schemes for Euclidean TSP and other Geometric Problems, Proceeding of 37th IEEE Symposium on Foundations of Computer Science, 2-12.

[7] Winter, P and Zachariasen, M.(1998), Large Euclidean Steiner Minimum trees : an improved exact algorithm, Networks 30, 149-166.

[8] G.-H. Lin and G.L. Xue(1999), Steiner tree problem with minimum number of Steiner points and bounded edge-length, Information Processing Letters, 53-57

[9] Joonmo Kim and Inbum Kim(2003), Approximation ratio 2 for the Minimum Number of Steiner Points, Journal of KISS, 387-396

[10] T.H. Cormen, C.E Leiserson, R.L. Rivest and C.Stein, Introduction to Algorithms, McGraw-Hill Book Company

[11] F.K. Hwang, D.S. Richards and P. Winter(1992), The Steiner Tree Problem, Annals of Discrete Mathematics, Vol.53, North-Holland

[12] F.K. Hwang(1991), A Primer of the Euclidean Steiner problem, Annals of Operations Research, Vol.33, pp.73-84

[13] E.J. Cockayne and D.E. Hewgrill(1986), Exact Computation of Steiner minimal trees in the plane, Information Processing Letters, Vol.22, pp.151-156

[14] E.J. Cockayne and D.E. Hewgrill(1992), Improved computation of plane Steiner minimal tree, Algorithmica, Vol.7, pp.219-229

[15] D.Z. Du and F.K. Hwang(1990), An approach for providing lower bounds; solution of Gilbert-Pollak conjecture on Steiner ratio, Proceedings of IEEE 31sy FOCS, pp.76-85

[16] M.J. Smith and B. Toppur(1997), Euclidean Steiner minimal trees, minimum energy configurations and the embedding problem of weighted graph in E3, Discrete Applied Mathematics, Vol.71, pp.187-215

[17] P. Winter(1985), An algorithm for the Steiner problem in the Euclidean plane, Networks, Vol.15, pp.323-345

[18] P. Mirchandani(1996), The multi-tier tree problem, INFORMS Journal on Computing, Vol.8, pp.202-218

Inbum Kim received the B.S. and M.S. degree in computer engineering from Seoul national university, South Korea, respectively. He worked as research engineer at Daewoo Telecom Ltd. and Oracle Korea, Seoul, South Korea. He is currently an associate processor at the department of Internet Information, Kimpo College, Kyonggi-do, South Korea and is working toward his Ph.D degree at university of Wisconsin Milwaukee. His interests are in network construction, mobile ad

hoc networks, computer architecture, computer theory and database system.

Chae-kak Kim received his B.S., M.S. and Ph.D. degrees in computer science from Soongsil university, South Korea, Yonsei university, South Korea and Soongsil university, South Korea, respectively. He worked as system engineer and software development team manager at LG Electronics and TriGem Computer Inc.. He is currently an associate processor at the Dept. of Internet Information, Kimpo College, Kyonggi-do, South Korea. His interests are in mobile and network security.

S. Hossein Hosseini received his Ph.D. degrees from University of Iowa. He is an associate professor and also co-chair of the department of Electrical Engineering and Computer Science, University of Wisconsin-Milwaukee, Wisconsin, USA. His research Interests are computer networks, high Performance distributed and parallel systems, fault tolerance, computer Architecture, performance evaluation, reliability, operating systems and real time systems.