EXTENSION OF SPANNING TREE AND APPLICATIONS

Fumihiko Shimada and Hiroaki Ishii

Received October 2, 2000

Abstract. A notion of spanning tree is utilized in many problems as an effective condition representing a set of simple path for some pair of two different nodes. But, when we make the model from some actual problem, it is hardly to obtain a feasible solution at the problem for this condition. Then, we introduce the notion of spannig pseudotree, and degree of similarity between tree and pseudotree. Moreover, we propose an example about extention of the problem where there exist the condition, solution graph is spanning tree.

1 Introduction A notion of spaning tree is useful in many problems as an effective condition representing a set of simple path for some pair of two different nodes. It is likely that these things are based on a characteristic of spanning tree, i.e., there is only one simple path from one node to the other for each pair of two different nodes, since a condition of the problem, "there aren't extra edges on solution graph (required graph)," is almost same as above characteristic. When we model some actual problems, however, it is hardly to obtain a feasible solution of the problem for the strict condition such that solution graph have to be spannig tree. Therefore, in this paper, we suggest a new notion "spannig pseudotree", relaxed notion of spanning tree, and one criterion for decision on degree of similarity between original spanning tree and spanning pseudotree. Moreover, we mention with an example about extension of the problem in which there exists this condition, that solution graph have to be spanning tree.

In section 2, we explain about original definition of traditional spanning tree.Next, we show a sample problem on which we obtain the subgraph such that each distance from a node on the original network to two root-nodes, that is two different data on the network, is as short as possible. In section 3, we suggest two kind of the notion "pseudotree", as extensions of the original notion in section 2. In section 4, we extend the sample problem in section 3 to relax the condition of the solution graph. In section 5, we conclude this paper.

2 Definition of spanning tree We consider the simple graph $G_0 = (V, E)$ consisting of the node set $V = \{1, 2, \ldots, n\}$ and E_0 the edges set ($E \subseteq \{(i,j) \mid (i,j) \in V \times V\}$, where length $d(i,j)$ is associated with each edge (i,j).

In this paper, spanning tree $T = \{V, E'\}$, as subgraph of the graph G_0, defined as follows:

(A) T is a connected graph
(There is a simple path between any two nodes of the nodes set V)

(B) T dosen't contain circuit
(There aren't more than one simple path between any two nodes of V),

2000 *Mathematics Subject Classification.* 03E72, 05C05, 90C70.
Key words and phrases. spanning tree, network, shortest path, Pareto optimal solution, chance constrained programming.

where, circuit means simple path whose both ends are same node.

Next, we suggest a problem on which we obtain the subgraph such that each distance between a node on the original network and two root-nodes (that is two different data on the network) is as short as possible. In a pratical case, for example, this model corresponds to the construction of the road network such that distance from two facilities, where they have about the same importance to each point is as short as possible. This condition that solution-graph T is spanning tree, corresponds to the efficient installment of the path from one of two facilities to each point in an actual case. In other words, this is a model constructed by an idea that plural different paths with same ends (i,j) are unnecessary for this problem.

We suggest that an objective on the problem is to minimize of maximum of the distance from two facilities to each node, where distance between two nodes is a length of the path on the subgraph T, and obtain the solution subgraph T' of original graph G among all T. Further, in practical reason, it is appropriate to minimize the number of edges on T, that is, make T spanning tree.

2.1 Notation

G **:** original graph. $G = (V, E)$

V **:** node set, $V = \{1, 2, \ldots n\}$

E **:** edge set, $E \subseteq \{(i,j) \mid (i,j) \in V \times V\}$

$d(i,j)$ **:** length between nodes i and j

N **:** network that $d(i,j)$ is associated with each edge (i,j).

T **:** solution (spanning tree), $T = \{V, E'\} \subseteq N$

s_1, s_2 **:** root nodes (different data points on the network N)

$P(i,j)$ **:** simple path from node i to j on the tree T
$P(i,j) = \{i, (i,i'), i', \ldots (j',j), j \mid i \in V, (i,i') \in E'\}$

$d_1(i)$ **:** distance from data point s_1 to node i with following T

$d_2(i)$ **:** distance from data point s_2 to node i with following T

$d(i)$ **:** maximum value of $d_l(i)$ (l=1,2)
$d(i) = \max\{d_1(i), d_2(i)\}$

Then the main problem is formulated as a Two-roots Shortest Spanning Tree Problem(**TSSTP**) shown as below:

TSSTP:

$$\begin{array}{ll} \textbf{Minimize} & \sum_{i \in V} d(i) \\ \textbf{s.t.} & d(i) = \max\{d_1(i), d_2(i)\} \quad (i \in V) \\ & T = \{V, E\} \quad \text{is a spanning tree} \\ & P(i,j) \text{ is a simple path between } i \text{ and } j \text{ on } T \\ & d^1(i) = \sum_{(j,j') \in P(s^1,i)} d(j,j') \\ & d^2(i) = \sum_{(j,j') \in P(s^2,i)} d(j,j') \end{array}$$

sAlgorithm1t

Step 0: Set $N_0 = \emptyset, E_0 = \emptyset$

Step 1: Obtain shortest path $P_0(s_1, s_2)$ from s_1 to s_2

Step 2: Add node set in P_0 to N_0, and add edge set in P_0 to E_O

Step 3: Obtain $d(i)$ about each node $i \in N_0$

Step 4: Obtain $d_i(j) = d(i) + d(i, j)$ among all edges (i, j) ($i \in N_0$ and $j \notin N_0$)

Step 5: Choose an edge (i', j') such that $d_{i'}(j')$ is minimum of $d_i(j)$ among all edges

Step 6: Add j' to N_0, (i', j') to E_0

Step 7: If $N_0 = N$ then Set $T = (N_0, E_0)$ and stop the algorithm
otherwise, set $d(j) = d_i(j)$ and return to **Step 4**

3 Extension of spanning tree In this section, we suggest two kinds of extension of spanning tree, one criterion is decision on degree of similarity between original spanning tree and spanning pseudotree.

3.1 Case1: Relaxation of condition (A) First, we extend notion of spanning tree to relaxation of condition **(A)**, T is a connected graph, that is, we replace the condition **(A)** with new condition that nodes on the graph is connected as maximum possibility of connectivity.

Then, we should use the notion, "how many can we obtain the number of edges on the component?", in other words, "how many can one tree cover the nodes on the pseudotree?", as a criterion for this notion.

Where, we show nesessary definition for the notion of this pseudotree as follows:

Pseudotree T' **:** $T' = \{V, E'\} \subseteq G = \{V, E\}$, T dosen't contain circuit and nodes on T' is connected everything possible

Degre of connectivity (Maximum possibility of μ_c **) :** μ_c = (the number of edges component whose scale is maximum for all component on T') $/|E'|$

sAlgorithm2t Then, we show the algorithm to obtain μ_c. Outline of the algorithm is as follows:

Step 0: Set $k = 0$

Step 1: Choose a data point i_k, where $i_k \in N/\bigcup_{l=0}^{k-1} N_l$

Step 2: Trace edges on T' from the node i_k, and add these nodes a node set V_k

Step 3: If $V/\bigcup_{l=0}^{k} V_l \neq \emptyset$ then $k = k + 1$ and return **Step 1**

Step 4: Obtain $|V_k|$ for all set V_k, and calculate the maximum $|V_{\max}| = \max_k |V_k|$

Step 5: Set $\mu_c = |V_{\max}|/|T'|$

3.2 Case2: Relaxation of condition(B) Second, we extend notion of spanning tree as a relaxation of condition **(B)**, T dosen't contain circuit, that is, we replace the condition **(B)** with new condition that graph $T' = \{V, E'\}$ dosen't contain circuit "everything possible"

Then, we should use the notion,

$$\frac{|E'| - (\text{the number of edges which circuits on the network } T' \text{ don't include })}{|E'|}$$

as a criterion for this notion.

Now, we make a nesessary definitions for the notion of this pseudotree as follows:

Pseudotree T' **:** $T' = \{V, E'\} \subseteq G = \{V, E\}$, T' is a connected graph and T' dosen't contain circuits everything possibility defind as the follow:

Degree of uniqyeness (The smallest possibility of circuit) μ_t **:** $\mu_c = (|E'| -$ "the number of edges which circuits on the network T' don't include ") $/|E'|$

sAlgorithm3t Then, we show the algorithm to obtain μ_t. Outline of the algorithm is as follows:

Step 0: Set $k = 0$, $T'' = T'$

Step 1: Choose a data point i_k

Step 2: Trace edges on T'' from the node i_k, label these nodes v_k and add these nodes a node set V_k
if path from i_k is branched into the plural path, record this junction and trace there respectively.

Step 3: If every path arrives at some leaf (a node that its degree is one), in other word, T'' dosen't contain circuit, set $\mu_t = |T''|/|T'|$ and stop this algorithm.

Step 4: Trace from each end of the latest edge to common ancestor when arrive at labeled node, and add these path and the latest edge to set C_k

Step 5: Shrink the circuit C_k to one node c_k on the netowork T''

Step 6: If there exist self-loop on T'', delete it from T''

Step 7: Set $k = k + 1$ and return to **Step 2**

4 Extension of the shortest-path tree problem In this section, we suggest that Example1, the Two-roots Shortest Spanning Tree Problem, is extented with the notion of pseudotree. Here, we use the extended notion of spanning tree by the relaxation of condition **(B)** as that of spanning pseudotree.

Therefore, one condition, "solution graph T must be spanning tree," is relaxed to "it is desirable that solution graph T is spanning tree." Then, we can replace above condition with "solution graph T must be spanning pseudotree,"

Moreover, we suggest that an objective on the problem is minimization of maximum of the distance from two facilities to each node and degree of similarity μ_t.

Now we define the following notations further.

T' **:** solution (spanning pseudotree), $T' = \{V, E'\} \subseteq N$

TC' **:** a graph removed some edges and nodes unrelated to formation of the circuits to the pseudotree T'

TT' **:** subgraph of T', $TT' = T' - TC'$

C^k **:** the kth component on the graph TC'

T^k **:** the kth component on the graph TT'

$T(T')$ **:** graph shrinked each component C^k to one node c^k, $T(T') = \{V(T'), E(T')\}$

$V(T')$ **:** a node set of $T(T')$, $V(T') = \{i' \mid i' \in (T' - TC')\} \cup \{c^k, k = 1, 2, \ldots\}$

$E(T')$ **:** an edge set of $T(T')$, $E(T') = \{(i', j') \mid (i', j') \in TT', i', j' \notin TC'\} \cup \{(i', c_k) \mid (i', j') \in TT', j' \in C_k\}$

$\mu_t(T')$ **:** ratio of $|E'|-$ "the number of edges which circuits on the network T' don't include" to $|E'|$

s_1, s_2 **:** root nodes (different data on the network N)

$P(i, j)$ **:** simple path from node i to j on the tree T'
$P(i, j) = \{i, (i, i'), i', \ldots (j', j), j \mid i \in V, (i, i') \in E'\}$

$d_1(i)$ **:** distance from data point s_1 to node i with following T'

$d_2(i)$ **:** distance from data point s_2 to node i with following T'

$d(i)$ **:** maximum value of $d_l(i)$ (l=1,2)
$d(i) = \max\{d_1(i), d_2(i)\}$

$D(T')$ **:** total value of $d(i)$ for all node i on T'
$D(T') = \sum_{i \in V} d(i)$

PS **:** a set of Pareto optimal solution $(D(T'), \mu_t(T'))$

Now, Pareto optimal solution is defined as below:

In this paper, vector $\mathbf{v^1} = (\mathbf{D}(\mathbf{T^1}), \mu_\mathbf{t}(\mathbf{T^1}))$ dominates vector $\mathbf{v^2} = (\mathbf{D}(\mathbf{T^2}), \mu_\mathbf{t}(\mathbf{T^2}))$, if $D(T^1) \leq D(T^2)$, $\mu_t(T^1) \geq \mu_t(T^2)$ and $\mathbf{v^1} \neq \mathbf{v^2}$. Vector $\mathbf{v} = (\mathbf{D}(\mathbf{T}), \mu_\mathbf{t}(\mathbf{T})$ is called nondominated vector if there exists no vector $\mathbf{v}'$ that dominates $\mathbf{v}$. Further, T with nondominated vector $\mathbf{v}$ is called Pareto Optimal solution.

In definition of TC' and TT', $T(T')$ is tree. Then, we can define the following value:

$\Delta_t(T', (i, j))$ **:** the number of edges that TC' include by addition the edge (i, j) to T'
$\Delta_t(T', (i, j)) =$ (length of the path (i,j) on the tree $T(T')$) +1

Then the main problem is formulated as a Bicriteria Two-roots Shortest Spanning Tree Problem(**BTSSTP**) shown as below:

BTSSTP:

Minimize $\sum_{i \in V} d(i)$
Maximize $\mu_t(T')$
s.t. $d(i) = \max\{d_1(i), d_2(i)\} \quad (i \in V)$
$T' = \{V, E\}$ is a spanning pseudotree
$P(i, j)$ is a simple path between i and j on T'
$d^1(i) = \sum_{(j,j') \in P(s^1, i)} d(j, j')$
$d^2(i) = \sum_{(j,j') \in P(s^2, i)} d(j, j')$

sAlgorithm4t

Step 0: Obtain an initial feasible solution T_0 on condition that aolution graph is spanning tree **(optimal solution of example1)**

Step 1: Set $T(T_0) = t_0$, $l = 0$

Step 2: Obtain $\Delta(T_0, (i, j))$ for each edge $(i, j) \in G/T_0$

Step 3: Choose an edge (i', j') among the some edges whose parameter $\Delta(T_0, (i, j))$ is minimum such that $\sum_{i \in V} d(i)$ is minimum

Step 4: If $D(T_{l+1}) < D(T_l)$ then, add a solution vector ($D(T_{l+1}), \mu_t(T_{l+1})$) to PS

Step 5: Set $T_{l+1} = T_l \cup (i, j)$

Step 6: Obtain $T(T_{l+1})$ by reduction of new circuit formed by adding the edge (i, j)

Step 7: Calculate $\Delta_t(T_{l+1}, (i, j))$ for each edge (i, j) refered to $\Delta_t(T_l, (i, j))$

Step 8: Set $l = l + 1$ and return to **Step 3**

5 Conclusion In this paper, we suggest two kinds of the notion of "spanning pseudotree", the extention of the notion of spanning tree, and one criterion for decision on degree of similarity between original spanning tree and spanning pseudotree.

Moreover, we show a sample problem on which we obtain the subgraph such that an each distance from a node on the original network to two root-node, that is two different data points on the network, is as short as possible, and extend it to relax the condition of the solution graph.

Lastly, we think that this notion can be extendable to the many problem exploreity the condition that solution graph must be spanning tree.

References

[1] D. Dubois & H. Prade, "*Fuzzy Sets and Systems*", Academic Press, New York, 1980.

[2] N. Christfides, "*Graph Theory: an algorithm approach*", Academic Press, New York, NY, 1975

Department of Applied Physics, Graduate School of Engineering
Osaka University 2-1 Yamada-oka, Suita, Osaka 565-0871, Japan
E-mail: shimada@ap.eng.osaka-u.ac.jp