CANADIAN SOCIAL SCIENCE
Vol. **6**, No. **3**, **2010**, pp. 73-82

ISSN 1712-8056
www.cscanada.net
www.cscanada.org

A Revised Optimal Spanning Table Method for Expanding Competence Sets[1]

UNE METHODE DE TABLEAU CONSTRUIT OPTIMALE REVISEE POUR DEVELOPPER LES ENSEMBLES DE COMPETENCE

CHEN Jian-xun[2]

FENG Jun-wen[3]

Abstract: The optimal expansion problem of competence sets can be solves by either mathematical programming method or table based method developed by Feng (2001). Compared to the mathematical programming method, table based method for competence set expansion is a more efficient algorithm in using relevant tableaus to solve the optimal expansion problems. This paper proposes a revised table based method to facilitate developing a computer code. A computer program, called TBM, based on the revised algorithm, was developed to solve the large scale problems of expanding competence sets. A numerical example is given, and some possible future research topics on the related theme are discussed.
Keywords: competence set expansion; habitual domains; spanning table method

Résumé: Le problème de l'expansion optimale des ensembles de compétence peut être résolu soit par la méthode de programmation mathématique, soit par une méthode basée sur les tableaux développée par Feng (2001). Comparée à la méthode de programmation mathématique, la méthode basée sur les tableaux pour l'expansion des ensembles de compétence est un algorithme plus efficace dans l'utilisation des tableaux appropriés pour résoudre les problèmes d'expansion optimale. Cet article propose une méthode basée sur les tableaux révisé pour faciliter l'élaboration d'un code informatique. Un programme d'ordinateur, appelé TBM, basé sur l'algorithme révisé, a été développé pour résoudre les problèmes de l'expansion des ensembles de compétences à grande échelle. Un exemple numérique est donné, et quelques sujets possibles de futures recherches sur le thème sont débattues.

[1] **Supported by National Sciences Foundation of China (79870030)**
[2] School of Economics and Management, Nanjing University of Science and Technology, Nanjing, 210094, China.
[3] School of Economics and Management, Nanjing University of Science and Technology, Nanjing, 210094, China. Email: fengjunwen8@hotmail.com.
* Received on March 10, 2010; accepted on May 25, 2010

Mots-clés: expansion des ensembles de competences; domaines habituels; méthode de tableau construit

1. INTRODUCTION

Helping decision makers most efficiently and effectively acquire the needed competence sets so that they can confidently and competently solve their decision making problems is one of the important issues in decision aiding and competence set analysis and more competence management. The problem to analyze the competence set can be viewed as the problem how to acquire the needed competence set with optimal total benefit. As stated by Feng (1998), the competence set expansion problem is an optimal spanning tree problem, so traditional methods for competence set analysis including competence set expansion algorithms are discussed based on either graph theory or mathematical programming. Feng and Yu (1998) and Feng (2001) presented a new way based on table to discuss the competence set expansion problems.

Given the cost function c (i, j) from skill i to skill j among the given skills of the needed competence set (briefly called CS), the problem on how to expand from a subset of CS to the whole CS has been studied analytically and mathematically by Yu and Zhang (1990) when c is symmetric, and by Shi and Yu (1996) when c is asymmetric. When the competence set involves the compound skills, using the deduction graph without cycles, Li and Yu (1994) proposed a method to solve the expansion problems. These works all used the mathematical programming approaches to study the expansion problems. But the mathematical programming method usually results in a large number of constraints and decision variables in formulation even though the problem size is not very large. For example, assume that $Sk=\{x_0\}$, $Tr\backslash Sk=\{x_1,x_2,\ldots,x_n\}$, where Sk is the decision maker's acquired competence set of skills and Tr is the true competence set of the skills for a particular problem. According to the mathematical programming formulation given by Shi and Yu(1996), which is widely used in the field of competence set expansion analysis, both the number of decision variables and the number of constraints are n^2+2n when skills are fully connected. For instance, if there are 20 skills to be acquired in an expanding competence set problem, when the skills are fully connected, there are 380=20(20-1) connections among the skills, then the corresponding mathematical programming needs use 440 decision variables and 440 constraints in formulation.

The optimal spanning table method proposed by Feng and Yu (1998) is an efficient algorithm which uses relevant tableaus instead of the mathematical programming to solve the optimal expansion problems. Feng (2001) proposed a more efficient table based method for competence set expansion. It does not need use a large number of decision variables and constraints. But when a large scale problem is met, a computer may need to aid the human hand calculation. In order to develop the computer code, here a revised optimal spanning table method is proposed by modifying some technical operations of the original method. Based on the revised method, a computer program named TBM is developed, by means of which the expansion problems with table up to 1000×1000 which involves 1000 skills and 999000=1000(1000-1) connections. And further extension from 1000 skills is also possible without difficulty.

2. TABLE BASED METHOD FOR COMPETENCE SET EXPANSION: AN OVERVIEW ON BASIC TERMINOLOGY AND PROCEDURES

In the table based method proposed by Feng (2001), an *expansion table*, as shown in Table 1, is a matrix representation of a digraph, in which the component of row i and column j stands for the cost function c(i,j) to acquire skill x_j from x_i.

Table 1: Expansion Table

	x_1	x_j	x_n
x_1	c(1,1)	c(1,j)	c(1,n)
x_i	c(i,1)	c (i, j)	c(i,n)
x_n	c(n,1)	c (n, j)	c(n,n)

In the expansion table, there are may have some empty cells indicating that the corresponding connection or say arc does not exist in the digraph. A *connecting element* or simply *conn-element* in the expansion table is defined by c (i, j). Due to a conn-element, say c (i, j), is associated with the arc from x_i to x_j, we call the node x_i in the rows in the expansion table is the *out-node*, and node x_j in the column is the *into–node.*

Summary descriptions of the table based method proposed by Feng (2001) are as follows:

The method has two processes. One is forwards and the other is backwards. In the *forward process*, first of all, mark the optimal (in the cost case, here is minimal or minimum) conn-element with □ in the original expansion table called T_0 and cross out the corresponding column from table T_0. Then mark the optimal conn-element with □ in the remaining table of T_0. After this, check whether the newly marked □ conn-elements are forming cycle or not. If there is no cycle among marked □ conn-elements, then cross out the column that the most recently marked □ conn-element is in, and repeat the above process. If a cycle is detected, then compress the nodes on the cycle into a *compressed node* and transform the original table T_0 into a smaller table T_1, which contains a compressed node. Continually apply the above process on table T_1. The process will eventually stop when the *stopping rule* that signals an optimal spanning table is obtained for the final table T_p.

The *backwards process* begins when the stopping rule is met at the final table T_p. Unfold the compressed node in T_p to obtain an optimal spanning table in the previous table T_{p-1}. If $T_{p-1}=T_0$, then the process to find the optimal spanning table of T_0 is complete. Otherwise, continue the unfolding process until an optimal spanning table of T_0 is found.

To sum up, the procedures of the optimal spanning table method consist of the following steps:

Step 0. Initializing condition;

Step 1. Selecting and marking procedure;

Step 2. Cycle detecting procedure;

Step 3. Crossing out procedure;

Step 4. Stopping rule;

Step 5. Compressing procedure;

Step 6. Unfolding procedure.

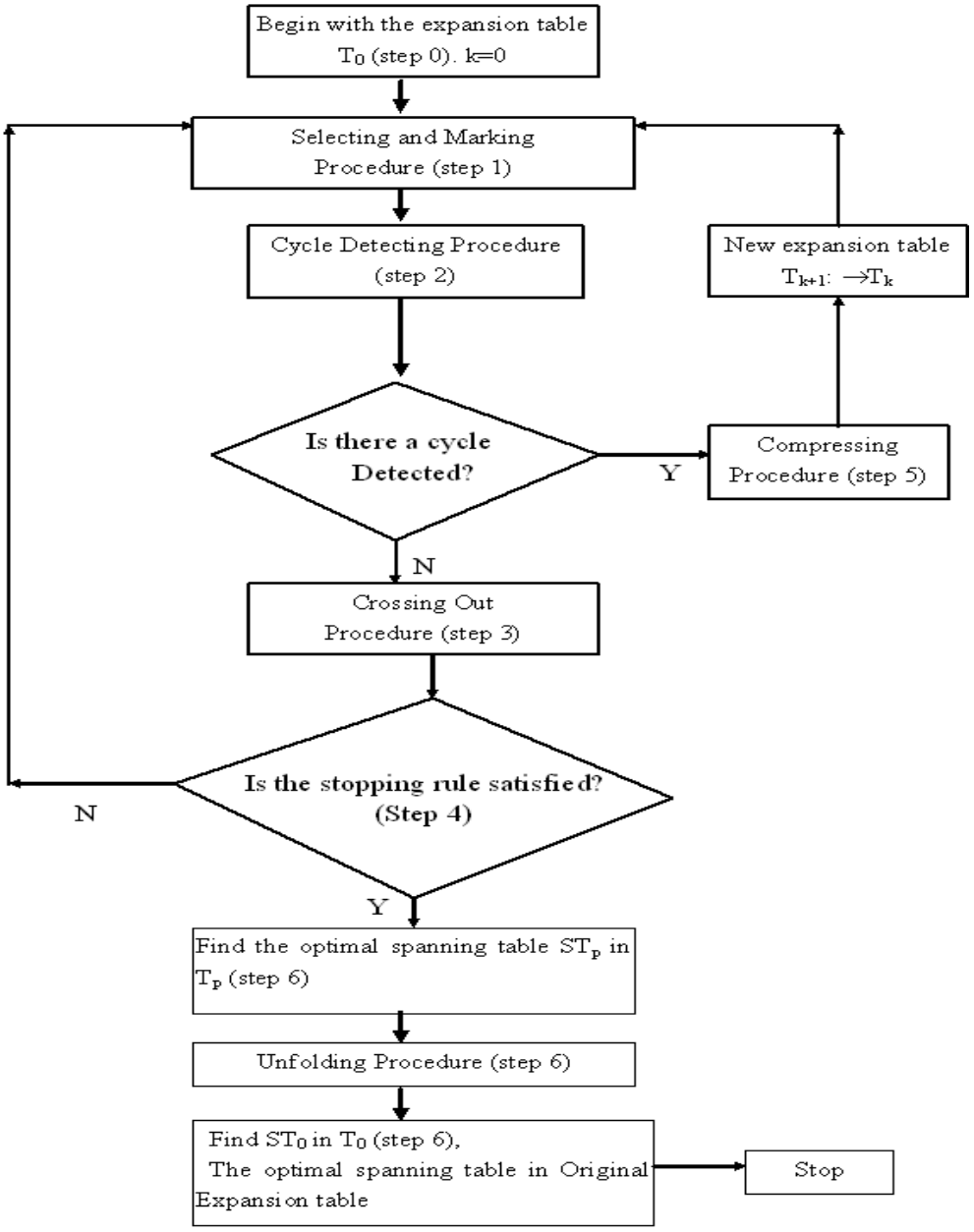

Figure 1 the flowchart of the table based method for competence set expansion

For more detailed description of the table based method for competence set expansion, please refer to Feng and Yu (1998) and Feng (2001).

3. A REVISION OF TABLE BASED METHOD

For the convenience of computer programming, a revised version of table based method for competence set expansion is split into eight procedures: Initializing, *Choosing*, *Constructing candidate list*, *Sorting*, *Marking and Detecting cycle*, *Compressing*, *Unfolding* and *Outputting* procedures. The details of these

procedures are given in the following. The basic ideas of these procedures are similar to the original method, but the technical operations of these, especially the Marking and cycle Detecting, compressing and unfolding procedures, are obviously differ.

The revision method starts augmenting the expansion table by appending two rows, *marked index (MI)* and *Cycle Index (CI)*, to the expansion table as Table 2.

Table 2: Augmented Expansion Table

	x_1 …	x_j …	x_n
x_1	c(1,1)	c(1,j) …	c(1,n)
x_i	c(i,1)	c (i, j) …	c(i,n)
x_n	c(n,1)	c (n, j) …	c(n,n)
MI	0	0	0
C I	0	0	0

The optimal conn-element of each column is chosen from the augmented expansion table. For the chosen conn-element, the corresponding out node, into node *MI* and *CI* form a *record*, and the collection of such records forms a list called *candidate list*. After the formation of the candidate list, the list is sorted by the *key values* (that is, the chosen conn-elements), so that the list is in nondecreasing order of the chosen conn-elements.

In the Making and cycle Detecting procedure, the revision method applies *MI* to indicate whether a conn-element is marked, and *CI* to indicate whether a conn-element forms a cycle with the conn-elements that are already marked. In the compressing procedure, some data are stored for tracing compression to *six stacks*, so that the Unfolding procedure can be more easily done. Besides, the Outputting procedure can rearrange the output of the unfolding procedure so as to easily draw the optimal spanning tree.

Suppose there are n skills in a competence set expansion problem. The whole eight procedures of the revised method are given in the following.

Initializing procedure

Augment the expansion table by appending two rows to the original expansion table, and label *marked index (MI)* and *cycle index (CI)*, respectively. Let all elements of row *MI* and *CI* be 0 at first initializing. The augment expansion table is shown in Table 2.

Choosing procedure

Selecting the optimal, that is minimal, conn-element from each column in the augmented expansion table. Randomly choose, if there is more than one optimal conn-element in each column. The chosen optimal conn-element of column j is denoted by *C(j)*, and the corresponding out-node and into-node are denoted by *O(j)* and *I(j)*, respectively. That is as follows.

$$C(j) = \min_i c(i, j), \qquad i, j = 1, 2, ..., n \tag{1}$$

$$O(j) \in \{k \,|\, \min_i c(i, j) = c(k, j)\} \tag{2}$$

$$I(j) = j, \; j = 1, 2, ..., n \tag{3}$$

Constructing the candidate list procedure

Construct the candidate list, which is a collection of records (R_1, R_2,…,R_n) that consists of 5 fields: *conn-element, out-node, into-node, MI* and *CI*. The conn-element field stores the chosen conn-elements the out-node fields stores the out-nodes, the into-node field stores the into-nodes, the MI field stores marked index, and the CI field stores cycle index. The structure of the candidate list is shown as Table 3 in the following.

Table 3: Candidate List Table

Record	conn-element	out-node	into-node	MI	CI
R_1					
R_2					
.					
R_j					
.					
.					
R_n					

Sorting procedure

Rearrange the records (R_1, R_2,…,R_n) of the candidate list. Let the conn-element field be the *key value*. Sort and reorder the records according to the key value in nondecreasing order, that is, find a permutation, saying σ , such that $C(\sigma(j)) \le C(\sigma(j+1)), 1 \le j \le n-1$. This procedure produces a *sorted candidate list*.

Marking and Detecting cycle procedure

Sequentially check each record in the sorted candidate list. For instance, check the *i*th record, denoted by R_i. First, set MI for R_i as 1, i.e. set MI(i)=1, and let temporary variable, *temp*, be the out-node of R_i. Next, select the record R_j whose into-node is that corresponding to *temp*. As R_i is found, check whether MI(j) equals to 0 or 1. If MI(j)=0, then the inclusion of R_i's conn-element does not form a cycle. In this case, clear aill cycle indexes in the list, i.e. reset all CI of the marked records that with MI=1 to be 0. Then, add *count*, the number of the marked record, by 1, i.e. *count=count*+1. After this, continue to check the next record R_{i+1}. If MI(j) equals to 1, set CI(j)=1 and let *temp* be R_j's out-node. Then continually select the record whose into-node is that corresponding to *temp*, and check its MI with 1.

Repeat the above selecting and checking MI process. If the newly selected record is identical to the starting record R_i, then a cycle is detected. Once a cycle is detected, append the records whose MI=1 to the unfolding list and go to the *compressing procedure*.

Stopping Rule: If no cycle exists after checking all the n-1 records, i.e. *count=n-1*, then the optimal spanning tree has been found. In this case, append the records whose MI=1 to the unfolding list and go to the *outputting procedure*.

Compressing procedure

Assume the cycle detected has m nodes. Compress the nodes in the cycle, denoted by **C**, into a compressed node, denoted by x_{n+r}, where *r* stands for the *r*th compression. Then transform the expansion table into one in the next stage, which has *(n-m+1)* nodes, and the corresponding cost function is defined as follows:

$$c(x_i, x_j) = c(i, j) \quad if \quad x_i \notin \mathbf{C} \quad \text{and} \quad x_j \notin \mathbf{C} \tag{4}$$

And for any node xi not in **C**, define

$$c(x_{n+r}, x_i) = \min\{c(y, x_i) : y \in C\} \quad (5)$$

$$c(x_i, x_{n+r}) = \min\{c(x_i, y) + c(x_s, x_t) - c(x_y, y) : y \in C\} \quad (6)$$

Where $c(x_s,x_t)$ is the largest conn-element in cycle **C**, and x_y is such that (x_y,y) is the conn-element in cycle **C**. The equations (5) and (6) are called the transformation equations defined by Feng and Yu (1998). For the detailed meaning of the *transformation equation*, please refer to the work by Feng and Yu (1998).

In order to facilitate the trace of the compression for unfolding procedure, six stacks, named $\boldsymbol{S^1}$, $\boldsymbol{S^2}$, $\boldsymbol{S^3}$, $\boldsymbol{S^4}$, $\boldsymbol{S^5}$ and $\boldsymbol{S^6}$, are employed for storing $c(x_i,x_{n+r})$, x_i, y, $c(x_{n+r},x_i)$, y and x_i in this procedure. The way to implement these stacks is using a two-dimensional array, say $\mathbf{S}_{p,q}$ in which p stands for the total number of compressing up to this point, and q for the number of the nodes not on the cycle. Let y be the component which solves equation (6) for $c(x_i,x_{n+r})$. Store $c(x_i,x_{n+r})$, x_i, y in $\boldsymbol{S_1}$, $\boldsymbol{S_2}$ and $\boldsymbol{S_3}$ respectively. Thus $S^1_{r,k} = c(x_i, x_{n+r}), S^2_{r,k} = x_i, S^3_{r,k} = y$, where the subscript (r,k) indicates the stack's row and column number respectively, r indicates the r^{th} compression and k indicates the k^{th} node that does not in the cycle. That is, $(S^2_{\bullet k}, S^3_{\bullet k})$ is the arc which solves (6) with $c(x_i, x_{n+r}) = S^1_{\bullet k}$.

Similarly, let y be the component which solves equation (6) for $c(x_{n+r},x_i)$. Store $c(x_{n+r},x_i)$, y and x_i respectively in $\boldsymbol{S^4}$, $\boldsymbol{S^5}$ and $\boldsymbol{S^6}$. Thus, $S^4_{r,k} = c(x_{n+r}, x_i), S^5_{r,k} = y, S^6_{r,k} = x_i$ .That is, $(S^5_{\bullet k}, S^6_{\bullet k})$ is the arc which solves (6) with $c(x_{n+r}, x_i) = S^4_{\bullet k}$. After creating the expansion table of a new stage and storing the data to stacks, go to the Choosing procedure and continue the Choosing procedure to the Marking and Detecting cycle procedure.

Unfolding procedure

If there are p times of compression to find the final optimal spanning tree, i.e. the stopping rule of the forward procedure is reached after p times of compression, then the unfolding procedure must be operated p times. So sequentially and backwardly unfold the compressed nodes x_{n+r}, where $r=p,p-1,\ldots,1$. There are two steps to complete the unfolding procedure.

Step (a). Determine whether or not the compressed node x_{n+r} is the root of the tree. For this purpose, check if there is a record whose into-node of the unfolding list is x_{n+r}. Denote such a record by $\overline{R}$. If $\overline{R}$ does not exist, it means that x_{n+r} is the root. In this case, first discard the worst, i.e. maximum, conn-element in the cycle that is detected in stage $r-1$. To do this, select the record whose stage number equals $r-1$, and whose conn-element is the largest in such a stage. Then, let all fields of that record be "*".

Therefore, select the record whose out-node is x_{n+r}. Denote such a record by $\underline{R}$. Note that the $R^6_{r,k}$'s element is identical to $\underline{R}$'s into-node. Then, replace $\underline{R}$'s conn-element, out-node and into-node with $S^4_{r,k}, S^5_{r,k}$ *and* $S^6_{r,k}$, respectively.

If x_{n+r} is not the root, thus $\overline{R}$ exists. Then, the element of $S^2_{r,k}$ is $\overline{R}$'s out-node. Replace $\overline{R}$'s conn-element, out-node and into-node by $S^1_{r,k}, S^2_{r,k}$ *and* $S^3_{r,k}$, respectively.

Continue the above processes of Step (a) until $r-1$, and then go to the Step (b).

Step (b). Check the records of the unfolding list backwardly, starting from the last record to the first one. When there are two or more records which have the same into-node, delete all records except the one with the highest index of r, i.e. the compression index.

Outputting procedure

Since a optimal spanning tree in accordance to the unfolding list is not easy to be drawn after the unfolding procedure, the following outputting procedure is used to rearrange the unfolding list so that the optimal spanning tree can be easily found.

Step (c). Let i=1,j=1 and l=1. Locate the root of the optimal spanning tree by searching the node that does not appear in the into-node field of the unfolding list. Denote such a node by x_{root}. Then let $root(i,j)=x_{root}$.

Step (d). Search all of the records of the unfolding list to locate the records whose out-node is *root(i,j)*. Once such a record is located, below three sub-steps are followed: (1) append this record to the outputting table; (2) let the *root (i+1,l)* be this record's into-node; (3) let $l=l+1$.

Step (e). Let $j=j-1$ and check whether or not $j>0$. If so, continue Step (d); otherwise, let $i=i+1,j=l$, and check whether $u<v$ or not, where u represents the total number of the records in the output table, v represents the total number of the records in the unfolding list. If the condition is satisfied, continue Step (d) and Step (e); otherwise, outputting procedure is completed.

According to the above procedures, a computer program called TBM is developed for the purpose of computing the optimal expansion competence set by the Visual Basic for Application (VBA) on EXCEL 2007. It is a friendly program, which can be used to compute the optimal spanning tree with the revised method. The maximum size of the expansion table is 1000×1000, which can be applied to almost all the real-world applications.

4. AN NUMERICAL EXAMPLE

The example used by Feng (2001) is applied to illustrate the utility of the revised method.

Table 4: Expansion Table from Feng and Yu (1998)

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	*	9	8	4	5	9	4
x_2	9	*	4	6	4	9	9
x_3	3	4	*	2	3	5	4
x_4	4	8	5	*	1	7	8
x_5	5	9	8	8	*	1	4
x_6	3	4	2	7	3	*	6
x_7	3	4	8	5	9	4	*

By the TBM program, the result, i.e. the output table is shown in Table 5. For the length of the paper, the detailed procedures are omitted. The interested readers may follow the process to find the optimal spanning table and check the result with the following Table 5.

Table 5: Output Table

Conn-element	out-node	into-node
1	x_4	x_5
4	x_5	x_7
1	x_5	x_6
3	x_7	x_1
4	x_6	x_2
2	x_6	x_3

Once the output table is established, the optimal spanning tree can be easily drawn by following the

sequence of the output table, as shown in Figure 2. Optimal competence set expansion unit is 16

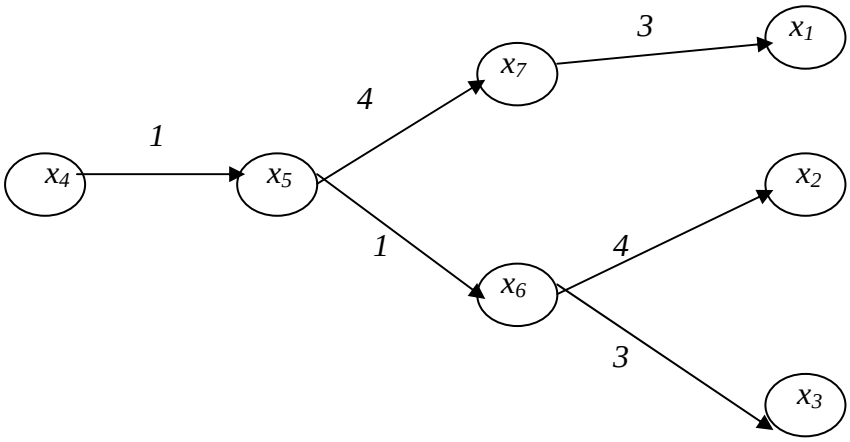

Figure 2: the optimal spanning result of competence set expansion example

5. CONCLUSIONS AND POSSIBLE FUTURE RESEARCH TOPICS

For facilitating the coding of a computer program, this paper proposed a revised version of optimal spanning table method developed by Feng and Yu (1998). The program can be used to handle large scale competence set expansion problems that the mathematical programming method can not manage. With BVA programmed OST software package, the optimal competence set expansion process based on the table can be easily solved as long as the expansion table is no bigger than 1000×1000 which is also extendable. By means of the aid of TBM package, the calculation of tableaus by the original optimal spanning table method developed by Feng and Yu (1998) and Feng (2001) is no longer tedious and time-consuming. In addition, TBM package is very useful in teams of dealing with the real-world competence set expansion problems.

The original optimal spanning table method by Feng and Yu (1998) and table based method by Feng (2001) is developed in the case of the cost function when considering the expanding criterion. If the criterion is benefit-type, the similar process can be easily followed by either transforming the benefit-type to cost-type or considering the every possible step in maximizing instead of minimizing.

One possible future research on optimal spanning table method is that to consider multiple criteria in the competence set expanding. In this case, we need to consider every possible variable or one-dimensional datum to vector or multiple-dimensional data. Whether the optimal spanning table method could be operated efficiently and effectively is still a problem.

Another possible future research topic on the optimal spanning table method is that to consider the cost or benefit randomly. In this aspect, cost or benefit function could be stochastic, grey, coarse or any other uncertain. Generally, in this case, the mathematical programming method is easy to formulate the problem but is difficulty to solve because of the large scale and nonlinearity. For the stochastic case, one may only consider the mean and standard variance for two criteria and transform the stochastic problem into the two criteria competence set expansion problem.

Still other topics on competence set analysis or expansion problems are needed to further discuss. For example, traditionally, the competences or skills of the decision maker were considered as the same in quality, so the concept of competence set was used. In fact, the competences or skills needed for a decision making problem being effectively and efficiently solved could be much more complicated, so in most cases, a competence *set* is a system made of competences called a *competence system*. Besides, traditional competence set analysis only discussed the problem of expanding the competence set, so we

called *analysis*. When we are mentioning the competence problem, we often mean one or more of the followings: competence scaling, competence leveraging, competence developing, competence expanding, competence cultivating, competence protecting, competence planning, competence organizing, competence controlling, competence leading, in generally, we may say *competence management*. If we consider the system characteristics of the competence, we may extend the competence set analysis into the field of the *competence system management*. Then we can use the system analysis approaches such as system modeling, system forecasting, system simulation, multiple criteria decision-making, optimization, system evaluation to study the competence system management problems. We can also apply uncertainty system theory and method such as stochastic system, grey system to study the competence system management problems. Some works are in progress, and hopefully some new methods for competence management may be proposed and developed.

REFERENCES

FENG, J.W. and YU, P.L. (1998). Minimum spanning table and optimal expansion of competence set. *Journal of Optimization Theory and Applications, Dec. 99(3)*: 655-679.

FENG, J.W. (1998). Table operations method for optimal spanning tree problem. *Journal of Systems Engineering and Electronics, 9(4):* 31-40.

FENG, J.W. (2001). Table based method for competence set expansion. *Transactions of Tianjin University. Jun. 7(2)*: 101-108.

LI, H.L. and YU, P.L. (1994). Optimal competence set expansion using deduction graphs. *Journal of Optimization Theory and Applications, 80(1)*: 75-91.

SHI, D. S. and YU, P.L. (1996). Optimal expansion and design of competence ser with asymmetric acquiring costs. *Journal of Optimization Theory and Applications, 88(3)*:643-658.

YU, P.L. and ZHANG, D. (1990). A foundation fro competence set analysis. *Mathematical Social Sciences, 20*: 251-299.