Qi, S. and Feng, J. *(2018).* A Revised Minimum Spanning Table Method for Expanding Competence Sets. *Bulletin of Business and Economics,* 7(2), 40-45.

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

Shoubin Qi
School of Economics and Management, Nanjing University of Science and Technology, China

Junwen Feng
School of Economics and Management, Nanjing University of Science and Technology, China

Abstract

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
JEL Codes: D85

[1] Supported by Social Sciences Foundation of China (16BZZ074)

I. 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. As stated by Feng (1998), the competence sets 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 the table to discuss the competence set expansion problems. 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 to 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.

In the table-based method proposed by Feng (2001), an *expansion table*, is a matrix representation of a digraph, in which the component of the row I and column j stands for the cost function c (I, j) to acquire skill x_j from x_i. 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 expanded table is the *out-node*, and node x_j in the column is the *into–node.* For more detailed description of the table-based method for competence set expansion, please refer to Feng and Yu (1998) and Feng (2001).

II.I. 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 a candidate list*, *Sorting*, *Marking and Detecting cycle*, *Compressing*, *Unfolding* and *Outputting*. The details of these procedures are given in the following. The revised method starts augmenting the expansion table by appending two rows, *marked index (MI)* and *Cycle Index (CI)*, to the expansion table as Table 1.

Table 1 Augmented Expansion Table

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

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.

II. II. 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 1.

II. III. Choosing procedure

Selecting the optimal, that is minimal, conn-element for each column in the augmented expanded 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.

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

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

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

II. IV. 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 2 in the following.

Table 2 Candidate List Table

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

II. V. 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*.

II. VI. 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*.

II. VII. 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*.

II. VIII. 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

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

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

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 $\boldsymbol{S^1, S^2, S^3, S^4, S^5}$ 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.

II. IX. 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.

II. X. 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.

III. An Numerical Example

The example used by Feng (2001) is applied to illustrate the utility of the revised method. By the TBM program, the result, i.e. the output table is shown in Table 3. 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 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 1. Optimal competence set expansion unit is 16

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

Table 4 Output Table

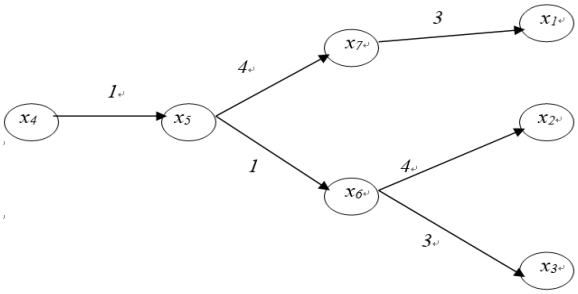

Figure 1 the optimal spanning result of competence

set expansion example

IV. Conclusions and Possible Future Research Topics

For facilitating the coding of a computer program, this paper proposed a revised version of the 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 cannot 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 terms 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 of 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 every possible step in maximizing instead of minimizing. One possible future research on the optimal spanning table method is that to consider multiple criteria in the competence set to expand. 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, gray, coarse or any other uncertain. 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.

References

Feng, J.W. and Yu, P.L. (1998). Minimum spanning table and optimal expansion of competence set. *Journal of Optimization Theory and Applications*, 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*, 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 for competence set analysis. *Mathematical Social Sciences*, 20, 251-299.