International Journal of Business and Management
www.ccsenet.org/journal.html
Vol. 4, No. 8
August 2009

Competence Set Expansion Strategy and Application with General Connectivity Parameters

Jianxun Chen
School of Economics and Management, Nanjing University of Science and Technology
Nanjing 210094, China
E-mail: cjianxun@mail.njust.edu.cn

Junwen Feng
School of Economics and Management, Nanjing University of Science and Technology
Nanjing 210094, China
E-mail: Fengjunwen8@hotmail.com

Abstract

Each decision making problem can be satisfactorily solved by using the expanding technologies of the Competence Set Analysis, where the competence set consists of the decision maker's ideas, knowledge, information, experience, skills and capacities, etc., directly or indirectly related to the decision making problem. This article proposes a heuristic method to find the optimal competence set expansion strategy with the connectivity parameters being general, that is, either symmetric or asymmetric. The optimality of the method is proven, and its applications in the personnel recruitment and training planning problems are discussed. Some conclusions and suggestions to be developed in a further work are included.

Keywords: Competence set expansion, Optimal strategy, Habitual domains

1. Introduction

Based on the decision maker's acquired competence set and the truly needed competence set to solve the decision making problems, the competence set expanding technology concentrates its study on the strategy of how and from where to start to expand the acquired competence set to the needed competence set to enable the decision maker or makers to confidently and successfully solve his, her or their decision making problems. Given the connectivity parameters $m(\cdot,\cdot)$ between the elements of the competence set (CS) on the habitual domains (HD), where $CS\subseteq HD$, the problem of how to expand from one CS to another CS is studied analytically and mathematically by Yu and Zhang (1990) when $m(\cdot,\cdot)$ is symmetric, and Shi and Yu (1996) when $m(\cdot,\cdot)$ is asymmetric. Li and Yu (1994) did some research work by means of deduction graphs without cycles when there are intermediate elements and multilevels. Under risky and uncertainty cases, Feng (2001) discussed the competence expansion problems and given several expansion strategies. Very Recently, Yu and Larbani (2009) discussed application of the competence set expansion in the game theory and several new ideas are put forwards. But they all studied using mathematical programming algorithms, especially, the integer programming algorithms, so even a simple problem often needed to be solved by using some software package. Furthermore, even though all these papers theoretically deal with the general cases, practically all can be only used to find the expanding process from a given skill, instead of a given set of skills.

In the article, as far as the general connectivity parameters $m(\cdot,\cdot)$ (maybe negative or positive) are concerned, based upon the digraphs, a heuristic method is given to find the optimal expansion strategy within a certain competence set or habitual domain. This heuristic method can also be used to optimally expand the acquired competence set to the needed competence set or habitual domain. The optimality and applications of the method are also discussed. This heuristic method differs from the methods developed in papers [1,2,3] in the following aspects: (1) It can deal with the case that there are cycles in the digraphs; (2) It can begin from any given acquired skills set; (3) It is a heuristic method rather than a analytical method; (4) The expansion process obtained may be not in the form of sequences which are the arrangements of the required skills; (5) It can handle the case that there are multivalues between the skills; (6) It is some kind of extension of the deduction graph method [3].

2. The Heuristic Method Development for Expansion Strategy

Suppose the discussed universal is some habitual domain HD which has a finite number (say n) of elements, that is, HD={ x_1 , x_2 , ... , x_n }, then connectivity parameters $m(\cdot,\cdot)$ between the elements in HD can be represented by an $n\times n$ matrix $M=(m_{ij})_{n\times n}$ where $m_{ij} = m(x_i ,x_j)$. Based upon the set HD and matrix M, a digraph can be constructed with the vertices corresponding to elements x_1 , x_2 , ... , x_n , each arc representing the way a connectivity may be reached between any two elements in {x_1 , x_2 , ... , x_n }, and each arc weight being the corresponding connectivity parameter between the two elements. Let the deducted digraph from the HD and M be denoted by DG(HD, M).

Definition 1. An ***arborescence*** is defined as a directed tree (there is only one directed path between any pair of vertices or skills) in which no more than two arcs are directed into the same vertex. A ***branching*** is defined as a forest (a set of unconnected trees) in which each tree is an arborescence. A ***spanning arborescence*** of the digraph is an arborescence that is also a spanning tree. A ***spanning branching*** is any branching that includes every vertex in the digraph.

Definition 2. By an ***expansion strategy***, we mean a spanning arborescence or spanning branching, that is, a directed spanning tree or a set of unconnected directed spanning trees in each of which no more than two arcs are directed into the same skill (vertex).

Definition 3. The ***connectivity of an arborescence (branching)*** is defined as the sum of the weights of the arcs in the corresponding arborescence (branching). A ***maximum arborescence (branching)*** of a digraph is any arborescences (branching) of the digraph with the largest possible weight sum.

Definition 4. ***An expansion strategy is optimal with respect to the sum operator among the connectivity parameters*** if it is a maximum spanning arborescence (if one exists).

Definition 5. ***An expansion strategy from a specified skill (vertex)*** is a strategy with the root being the vertex, and is optimal with respect to the sum operator if it is a maximum spanning arborescence rooted at the vertex.

Definition 6. ***An expansion strategy from a specified set of the skills (vertices)*** is a strategy with the root being within the set, and it is optimal with respect to the sum operator if it is a spanning arborescence rooted at some vertex in the set with the largest sum of the weights except the weights within the set.

From the above the definitions, we know that the concept of branching is more general than that of arborescences. Given the DG(HD,M), if we can find the maximum branching, then according to the following observations, we can easily find the maximum spanning arborescence (optimal expansion strategy) by adding to each arc a large enough positive constant N.

Firstly, a spanning branching is a spanning arborescence if and only if it has exactly one less arc than vertices. No branching has more arcs than this. Secondly, an optimum branching contains no arc with negative weight, and indeed may be empty if all connectivity parameters are not positive, that is, all $m_{ij} \leq 0$. And it is worth noting that even if all parameters are positive and the digraph contains a spanning arborescence, an optimum branching need not be an arborescence. Thirdly, a spanning arborescence which is optimum relative to weights m_{ij} is also optimum relative to weights m_{ij} +k for any constant k, and m_{ij} ×k for any positive constant k(if all m_{ij} 's are not negative, since every spanning arborescence has the same number of arcs. Fourthly, if there is a spanning arborescence in some digraph DG(HD,M), then an optimum one, i.e., one which has a maximum total weight can be found as an optimum branching in the digraph DG(HD,M+h) where M+h means the matrix (m_{ij} +h), and $h > \sum |m_{ij}|$. This is because the constant h is larger than the difference in total weight (relative to weights m_{ij}) of any two branchings in the digraph, so an optimum branching in the digraph, relative to weights $m_{ij}'=m_{ij}+h$ will be a branching with a maximum number of arcs, and in particular, it will be a spanning arborescence if and only if the digraph contains a spanning arborescence.

The optimal expansion strategy from the specified skill, say x_0 , can be obtained using the above method by adjoining a new arc carrying arbitrary weight which is directed toward x_0 and from a new vertex having no other incident arcs, and letting the arcs directed toward x_0 have zero weights.

The optimal expansion strategy from the specified skills set, say Sk, can be found using the above method by setting the weights with the heads in the set be zero and adjoining a new arc carrying an arbitrary weight which is directed toward some $x_0 \in$ Sk and from a new vertex having no other incident arcs. The selection of $x_0 \in$ Sk can be arbitrary. If the optimum exists, then there must exist some $x_0 \in$ Sk such that the corresponding spanning arborescence is rooted at the vertex x_0 .

There may be many optimal strategies.

We shall now proceed to describe the heuristic method to find the maximum branching.

The maximum branching method uses two buckets, the vertex bucket and the arc bucket. The vertex bucket contains only vertices that have been examined; the arc bucket contains arcs tentatively selected for the maximum branching. The arcs in the arc bucket form a branching. Initially both buckets are empty.

The method successively examines the vertices in any arbitrary order. The examination of a vertex consists entirely of selecting the arc with the greatest positive weight that is directed into the vertex under examination (if any). If the addition of this arc to the arcs already selected for the arc bucket maintains a branching, then this arc is added to the arc bucket. Otherwise, this arc would form a cycle with some arcs already in the arc bucket. If this happens, then a new, smaller digraph is generated by "shrinking" the arcs and vertices in this cycle into a single vertex. Some of the arc costs are judiciously altered in the new, smaller digraph. The vertex and arc buckets are redefined for the new digraph as containing only their previous contents that appear in the new digraph. The examination of each vertex continues as before. The process stops when all vertices have been examined.

Upon termination, the arc bucket contains a branching for the final digraph. The final digraph is expanded back to its predecessor by expanding out its "artificial" vertex into a cycle. All but one of the arcs in this cycle is added to the arc bucket. The arc that is not added to the arc bucket is carefully selected so that the contents of the arc bucket retain a branching. This process is repeated until the original digraph is regenerated. The arcs in the arc bucket upon termination turn out to be a maximum branching.

Denote the original digraph for which the maximum branching is sought by G_0 , and denote each successive digraph generated from G_0 by G_1 , G_2 ,..... The vertex and arc buckets used for these digraphs will be denoted by V_0 , V_1 , ... and A_0 , A_1 , ..., respectively. We are now ready to state the method formally.

Optimal (maximum) Expansion Strategy Method

Initially, all buckets V_0 , V_1 , ... and A_0 , A_1 , ... , are empty. Set I=0.

Step 1. If all vertices of G_i are in bucket V_i , go to step 3. Otherwise, select any vertex v in G_i that is not in bucket V_i . Place vertex v into bucket V_i . Select an arc γ with the greatest positive weight that is directed into v. If no such arc exists, repeat step 1; otherwise, place arc γ into bucket A_i . If the arcs in A_i still form a branching repeat step 1; otherwise, go to step 2.

Step 2. Since the addition of arc γ to A_i no longer causes A_i to form a branching, arc γ forms a cycle with some of the arcs in A_i . Call this cycle C_i . Shrink all the arcs and vertices in C_i into a single vertex called v_i . Call this new digraph G_{i+1} . Thus, any arc in G_i that was incident to exactly one vertex in C_i will be incident to vertex v_i in digraph G_{i+1} . The vertices of G_{i+1} are v_i and all the vertices of G_i not in C_i . Let the weight of each arc in G_{i+1} be the same as its weight in G_i except for the arcs in G_{i+1} that are directed into v_i . For each arc (x,y) in G_i that transforms into an arc (x,v_i) in G_{i+1} , let

$$m(x,v_i)=m(x,y)+m(r,s) -m(t,y) \text{ --------------transformation equation}$$

where (r,s) is the minimum weight arc in cycle C_i , and where (t,y) is the unique arc in cycle C_i whose head is vertex y. At this point, observe that $m(r,s)\geq 0$, $m(t,y)\geq m(r,s)$ and $m(t,y)\geq m(x,y)$ since arc (t,y) was selected as the arc directed into vertex y. Let V_{i+1} contain all the vertices in G_{i+1} that are in V_i , that is, $V_{i+1} =G_{i+1}\cap V_i$. Thus, $v_i \in V_{i+1}$. Let A_{i+1} contain all the arcs in G_{i+1} that are in A_i , i.e., $A_{i+1} =G_{i+1}\cap A_i$. Thus, A_{i+1} contains the arcs in A_i that are not in C_i .

Increase i by one, and return to step 1.

Step 3. This step is reached only when all vertices of G_i are in V_i and the arcs in A_i form a branching for G_i . If i=0, stop because the arcs in A_0 form a maximum branching for G_0. If $i\neq 0$, two cases are possible:

(a) Vertex v_{i-1} is the root of some arborescence in branching A_i.

(b) Vertex v_{i-1} is not the root of some arborescence in branching A_i.

If (a) occurs, then consider the arcs in A_i together with the arcs in cycle C_{i-1} . These arcs contain exactly one cycle in digraph G_{i-1}, namely C_{i-1}. Delete from this set of arcs the arc in C_{i-1} that has the smallest weight. The resulting set of arcs forms a branching for digraph G_{i-1} . Redefine A_{i-1} to be this set of arcs.

If (b) occurs, then there is a unique arc (x,v_{i-1}) in A_i that is directed into vertex v_{i-1} . This arc (x,v_{i-1}) corresponds in digraph G_{i-1} to another arc, say arc (x,y), where vertex y is one of the vertices in cycle C_{i-1} that was shrunk to form vertex v_{i-1} .

Consider the set of arcs in A_i together with the arcs in cycle C_{i-1} . This set of arcs contains exactly one cycle in G_{i-1}, namely C_{i-1} , and exactly two arcs directed into vertex y, namely arc (x,y) and an arc in cycle C_{i-1} . Delete the latter arc from this set of arcs. The remaining arcs in this set form a branching in digraph G_{i-1} . Redefine A_{i-1} to be this set of arcs. Having redefined A_{i-1} , decrease i by one unit and repeat step 3.

The above maximum branching method can also be used to find (1) a minimum branching, (2) a maximum spanning arborescence (if one exists), (3) a minimum spanning arborescence (if one exists), (4) a maximum spanning arborescence rooted at a specified vertex (if one exists), and (5) a minimum spanning arborescence rooted at a specified vertex (if one exists), (6) a maximum spanning arborescence rooted at a specified vertices set (if one exists), (7) a minimum spanning arborescence rooted at a specified vertices set (if one exists).

3. Proof of the Optimality of the Method

Consider any digraph G_t produced by the method and consider the branching A_t produced by step 3 for digraph G_t. First, it will be shown that if A_t is a maximum branching for digraph G_t , then branching A_{t-1} is a maximum branching for digraph G_{t-1}.

To prove this, some definitions are needed. Let G′ denote the subdigraph consisting of all arcs in G_{t-1} not directed into a vertex in cycle C_{t-1} . Let G″ denote the subdigraph consisting of all the arcs in G_{t-1} not in G′. Thus, every arc of G_{t-1} is present in exactly one of these subdigraphs G′ and G″. Let A'_{t-1} denote the arcs in A_{t-1} that are in G′,and let A''_{t-1} denote the arcs of A_{t-1} that are in G″. Clearly, A'_{t-1} and A''_{t-1} are branchings in G′ and G″, respectively.

If branching A_{t-1} is not a maximum branching for digraph G_{t-1} , then there exists some branching B with greater total weight. Let B′ denote the arcs in B that are in G′, and let B″ denote the arcs of B that are in G″. Since B is a maximum branching, it follows that either B′ weighs more than A'_{t-1} or B″weighs more than A''_{t-1}.

Claim 1: A'_{t-1} is a maximum-weight branching for G′.

Claim 2: A''_{t-1} weighs as much as B″.

If both claims 1 and 2 are true, it follows that A_{t-1} must be a maximum branching for digraph G_{t-1}.

Note that the branching A_i produced by the method for the terminal digraph G_i is a maximum branching since it contains a maximum positively weighted arc directed into each vertex in G_i if such an arc exists. Since the method produces a maximum branching for the terminal digraph G_i, then if both claims 1 and 2 are true, then the method must produce a maximum branching A_{i-1} for G_{i-1}. By repeating this reasoning, we can conclude that if claims 1 and 2 are true, then the branching A_0 produced by the method is a maximum branching for the original digraph G_0.

Hence, it remains only to show that claims 1 and 2 are valid.

Proof of claim 1. Suppose that cycle C_{t-1} contains n vertices. There is one arc with positive weight directed into each of these n vertices in digraph G′ (otherwise, the method would not have formed cycle C_{t-1}). Since there are only n vertices in G′ that have arcs directed into themselves, a maximum branching for G′ cannot contain more than n arcs. Moreover, no branching in G′ can have weight exceeding the weight of cycle C_{t-1}, which consists of the maximum positive-weight arc directed into each of the n vertices in cycle C_{t-1}. However, at least one of the arcs in C_{t-1} must be absent from any maximum branching for G′ since a branching cannot contain a cycle. Thus, at least one of these n vertices, say vertex $y \in C_{t-1}$, must either have no branching arc directed into it or else have an arc (x,y), $x \notin C_{t-1}$, directed into it.

For each vertex $z \in C_{t-1}$, construct a branching B_z in G′ as follows:

(a) Include all arcs in cycle C_{t-1} except the arc in cycle C_{t-1} that is directed into vertex z.

(b) Include any maximum positive-weight arc (x, z), where $x \notin C_{t-1}$.

Select the branching B_z* with the greatest weight. From the transformation equation , branching B_z* is the branching A''_{t-1} generated by the method.

Consider any branching B_1 in G′ that is not of the form B_z. If only one of the arcs of C_{t-1} is not in B_1, it follows that B_1 cannot be a maximum branching for G′ since it is not of the form B_z. If two or more arcs of C_{t-1} are not in B_1, then either (1) each of these arcs is replaced by an arc of smaller weight directed into the same vertex or (2) no arc is directed into this vertex. In either case, this results in the decrease of the weight sum of arcs in the branching directed into the vertex. Hence, B_1 cannot be a maximum branching for G′. Thus, A'_{t-1} is a maximum branching for G′, and we can assume, without loss of generality, that A'_{t-1} is identical to B′. This concludes the proof of claim 1.

Proof of Claim 2. Two cases are possible:

(a) Branching A_t contains an arc (x, v_{t-1}) directed into vertex v_{t-1}.

(b) Branching A_t does not contain an arc directed into vertex v_{t-1}.

Case (a): By hypothesis, A_t is a maximum branching for G_t and contains an arc (x,v_{t-1}) directed into v_{t-1} . From claim 1, B′ is identical to A'_{t-1} and hence B′ contains an arc (x,y), where $x \in C_{t-1}$ and $y \in C_{t-1}$. Since B is a branching in G_{t-1}, it follows that B″ cannot contain a path of arcs from a vertex in C_{t-1} to vertex x. Thus, B″ must be a maximum branching for G″ that does not contain a path of arcs from a vertex in C_{t-1} to vertex x.

Each arc in G″ corresponds to an arc with identical weight in G_t. Moreover, each branching in G″ corresponds to a branching in G_t with identical weight. Consequently, if A''_{t-1} is not a maximum branching in G″ that contains no path of arcs from a vertex in C_{t-1} to a vertex x, then A_t is not a maximum branching in G_t that contains arc (x, v_{t-1}), which is impossible. Hence, A''_{t-1} has the same weight as B″, which proves the claim for case (a).

Case (b): Each arc in G″ corresponds to an arc with identical weight in G_t. By hypothesis A_t is a maximum branching for G_t. Since no arc in A_t is directed into v_{t-1}, it follows that every arc in A_t corresponds to an arc in A''_{t-1} . Moreover, any branching in G″ corresponds to a branching in G_t with the same weight. Hence, if A''_{t-1} were not a maximum branching in G″, then A_t would not be a maximum branching in G_t, which is a contradiction.

Thus, A''_{t-1} must be a maximum branching in G″ and have the same weight as B″, which completes the proof of claim 2. #

4. Applications in Personnel Recruiting and Training Program

Consider an available position which needs several skills, say 4, x_1, x_2, x_3, x_4, the connectivity parameters among the skills are represented by the following matrix:

Insert Table 1 here

The Research Committee's task is (1) if there is no candidate who has all of the four skills, how to select one of the candidates; (2) if each candidate only has one skill (whether different or same) and the number of the candidates is equal to or greater than that of the needed skills, how to select; (3) if the Committee prefers some candidate, how to expand his or her acquired skills so that he or she can be qualified for the position.

Suppose the only criterion to be considered is the connectivity parameters among the skills. Problem (1) is a compound problem for problems (2) and (3), so the basic problems are (2) and (3). Furthermore, problem (2) is a problem to find the optimal expansion strategy, problem (3) is a problem to find the optimal expansion strategy from some specified skill or skills set.

$\sum | m_{ij} |$=4.1, so let h=5, and $m'_{ij}=m_{ij}+h$. The adding results form the following matrix:

Insert Table 2 here

The process of finding the optimal expansion strategy is as follows.

The method will arbitrarily examine the skills or vertices in the subscript numerical order. The result of the examination of the first two skills is shown as follows:

Insert Table 3 here

After the skill x_3 has been examined, the arcs in bucket A_0 no longer form a branching since they contain a cycle (x_2,x_3), (x_3,x_2). At this point, the method shrinks this cycle into a vertex (or skill) v_0. A new matrix or digraph resulting from this shrinking is following:

Insert Table 4 here

The result of the examination of the skills for the above matrix or digraph is following:

Insert Table 5 here

At this point, the method has generated a maximum branching for the above matrix consisting of arcs (x_4,v_0), (x_4,x_1). Using this branching, step 3 expands v_0 back into its original cycle and adds arcs (x_3,x_2), (x_2,x_3) to the arcs (x_4,v_0), (x_4,x_1) already in the branching. Next, arc (x_3,x_2) with the smaller weight is deleted from the branching so that only one branching arc, namely (x_2,x_3) is directed into x_3. The resulting branching consists of arcs (x_4,x_2), (x_2,x_3), (x_4,x_1). The total weight of this branching equals 5.5+5.4+5.6= 16.5, which is the maximum possible weight.

So, the committee should select the candidate who has the skill x_4. If no candidate has the skill x_4, the problem becomes problem (3).

Now suppose the committee prefers the candidate who only has the skill x_2, then how does the committee expand his or her skills so as to fit the position?

By adding a "skill" x_0, the following matrix is formed:

Insert Table 6 here

By adding the constant h=6, the following matrix is formed:

Insert Table 7 here

The following result is reached:

Insert Table 8 here

The shrinking process is performed because of the cycle (x_4, x_1), (x_1, x_4), and the corresponding result is as follows:

Insert Table 9 here

The next examination result is as follows:

Insert Table 10 here

The maximum branching is obtained which consists of the arcs (x_2, v_0), (x_0,x_2) , (x_2,x_3). Eliminating the (x_1,x_4) from cycle (x_1,x_4) , (x_4,x_1) , finally, we get the optimal expansion strategy: (x_0,x_2) , (x_2,x_3), (x_2,x_4) , (x_4,x_1) with a maximum weight 7.0+6.4+6.4+6.6=26.4. So the optimal expansion strategy is $x_2 \rightarrow x_3$, $x_2 \rightarrow x_4$, $x_4 \rightarrow x_1$.

Furthermore, suppose the candidate the committee preferred has the skills x_1 and x_2, then how could the candidate expand his or her skills so as to fit the position?

Add a new vertex x_0 and arc (x_0, x_1) to the original digraph, and form a new matrix as follows:

Insert Table 11 here

The examination result is as follows:

Insert Table 12 here

The optimal expansion strategy has been obtained: $x_0 \rightarrow x_1$, $x_2 \rightarrow x_3$, $x_1 \rightarrow x_4$, i.e., $x_2 \rightarrow x_3$, $x_1 \rightarrow x_4$ with the total weight 6.4+6.5=12.9.

5. Conclusions

A heuristic method to find the optimal expansion strategy has been provided based on the digraphic concept with the expansion parameters being general (symmetric or asymmetric, having cycles or no cycles). The optimality of the method has been proven, and its applications in the personnel recruiting and training program is demonstrated step by step. Some research problems are still open. For example, if the parameters are multidimensional or fuzzy, how can we construct the corresponding method? And if the parameters are not additive, how can we define and design the expansion strategy? Especially, if the information between the skills is provided in the form of the activation propensity function, what do we do? Furthermore, if the skills are fuzzy, possibilistic or probabilistic, what do we do?

References

Feng J.W. (2001). Competence Set Expansion under Risk and Uncertainty. *Journal of Systems Engineering and Electronics*, Vol.11, No.2, 655-679.

Larbani M. and Yu P.L. (2009). Two-Person Second-Order Games, Part2: Restructuring Operations to Reach a Win-Win Profile. *Journal of Optimization Theory and Application*, January, 2009 online.

Li, H.L., and Yu, P.L. (1994). Optimal Competence Set Expansion Using Deduction Graphs. *Journal of Optimization Theory and Applications*, Vol. 80, No.1, 75-91.

Shi, D.S. and Yu, P.L. (1996). Optimal Expansion and Design of Competence Set with Asymmetric Acquiring Costs. *Journal of Optimization Theory and Applications*, Vol.88, No. 3, 643-658.

Yu P.L. and Larbani M. (2009). Two-Person Second-Order Games, Part1: Formulation and Transition Anatomy. *Journal of Optimization Theory and Application*, January, 2009 online.

Yu, P.L., and Zhang, D. (1990). A Foundation for Competence Set Analysis. *Mathematical Social Sciences*, Vol.20, 251-299.

Table 1. The skill matrix

$m(x_i, x_j)$	x_1	x_2	x_3	x_4
x_1	/	0.1	0.2	0.5
x_2	0.2	/	0.4	0.4
x_3	0.3	0.5	/	0.2
x_4	0.6	0.5	0.3	/

Table 2. The adjusted skill matrix

$m'(x_i, x_j)$	x_1	x_2	x_3	x_4
x_1	/	5.1	5.2	5.5
x_2	5.2	/	5.4	5.4
x_3	5.3	5.5	/	5.2
x_4	5.6	5.5	5.3	/

Table 3. The first two skills examination result

vertex examined	V_0	A_0
x_1	x_1	(x_4,x_1)
x_2	x_1, x_2	(x_4,x_1), (x_3,x_2)
x_3	x_1, x_2, x_3	(x_4,x_1), (x_3,x_2), (x_2,x_3)

Table 4. The new skill matrix after first shrinking process

	v_0	x_1	x_4
v_0	/	/	/
x_1	(5.2, 0.0)	/	5.5
x_4	(5.4, 5.3)	5.6	/

Table 5. The examination for the new skill matrix

vertex examined	V_0	A_0
v_0	v_0	(x_4,v_0)
x_1	v_0, x_1	(x_4,v_0), (x_4,x_1)
x_4	v_0, x_1, x_4	(x_4,v_0), (x_4,x_1)

Table 6. The skill matrix for new problem

$m(x_i,x_j)$	x_0	x_1	x_2	x_3	x_4
x_0	/	/	1	/	/
x_1	/	/	0.1	0.2	0.5
x_2	/	0.2	/	0.4	0.4
x_3	/	0.3	0.5	/	0.2
x_4	/	0.6	0.5	0.3	/

Table 7. The skill matrix after adding

$m'(x_i,x_j)$	x_0	x_1	x_2	x_3	x_4
x_0	/	/	7	/	/
x_1	/	/	6.1	6.2	6.5
x_2	/	6.2	/	6.4	6.4
x_3	/	6.3	6.5	/	6.2
x_4	/	6.6	6.5	6.3	/

Table 8. The vertex examination

vertex examined	V_0	A_0
x_0	x_0	
x_1	x_0, x_1	(x_4,x_1)
x_2	x_0, x_1, x_2	$(x_4,x_1), (x_0,x_2)$
x_3	x_0, x_1, x_2, x_3	$(x_4,x_1), (x_0,x_2),(x_2,x_3)$
x_4	x_0, x_1, x_2, x_3, x_4	$(x_4,x_1), (x_0,x_2),(x_2,x_3), (x_1, x_4)$

Table 9. The vertex shrinking

	v_0	x_0	x_2	x_3
v_0	/	/	/	/
x_0	0	/	7	
x_1	(6.1, 6.4)	/	/	6.4
x_4	(6.2, 6.2)	/	6.5	/

Table 10. The 2nd vertex examination

vertex examined	V_0	A_0
v_0	v_0	(x_2,v_0)
x_0	v_0, x_0	(x_2,v_0)
x_2	v_0, x_0 , x_2	(x_2,v_0), (x_0,x_2)
x_3	v_0, x_0 , x_2 ,x_3	(x_2,v_0), (x_0,x_2) , (x_2,x_3)

Table 11. The 2nd vertex adding

$m'(x_i,x_j)$	x_0	X_1	x_2	x_3	x_4
x_0	/	7	/	/	/
x_1	/	/	0	6.2	6.5
x_2	/	0	/	6.4	6.4
x_3	/	0	0	/	6.2
x_4	/	0	0	6.3	/

Table 12. The 3rd vertex examination

vertex examined	V_0	A_0
x_0	x_0	
x_1	x_0, x_1	(x_0,x_1)
x_2	x_0, x_1, x_2	(x_0,x_1)
x_3	x_0, x_1, x_2 , x_3	(x_0,x_1), (x_2,x_3)
x_4	x_0, x_1, x_2 , x_3 , x_4	(x_0,x_1), (x_2,x_3), (x_1, x_4)