Chung Yuan Journal
Vol. 27. No. 1, 1999. pp. 77-82

Evaluation for Computer-Simulated Problem Solving

JIA-SHENG HEH*

Department of Information and Computer Engineering
Chung Yuan Christian University
Chung-Li, 32023, Taiwan, R.O.C.

CHENG-JU HSU

Department of Information Management
Chung Yu Junior College of Business Administration
Keelung, Taiwan, R.O.C.

(Received: November 11, 1998; Accepted: March 30, 1999)

ABSTRACT

Problem solving is a procedure to transform the given state or source problem to the goal state or destination problem. Such serial transformation from source to destination is a PS (Problem Solving) path and a collection of PS paths can be illustrated in a computer by a directed graph, called problem solving network.

To evaluate the performance of PS paths, this paper proposes an evaluation vector of CPSN (Coordinate Problem Solving Network), which gives each problem a unique (x,y) coordinate, and describes the evaluation principles. In CPSN representation, x-coordinate indicates the target offset of the problem; whereas, y-coordinate represents its deviation from the shortest path. For a given PS path, an evaluation vector can be found to indicate the performance of each solving step. Each element of evaluation vector corresponds to the judgment index of the applied operator of each solving step. This judgment index can be interpreted by the given evaluation principles. Two practical examples of Hanoi tower and one-variable linear equation show that the evaluation vector can discover many evaluation results for the problem solving.

Key words: *problem solving, problem solving path, CPSN, coordinate problem solving network evaluation .*

I. INTRODUCTION

Problem solving is a representation of thinking. (Johnson, 1972) A problem is a certain state with certain conditions, objects and pieces of information. [1] The process of problem solving is to find a way around obstacles to attain an aim that was not immediately attainable. (Polya, 1965) The goal of the process is the desired or terminal state of these problems.

Hence, problem solving is to transform the problem from the given to the goal state. This is called the *computer-simulation approach* of problem solving, which assumes that a problem solver solves problems by applying operators to problem states. In such approach, a problem state is defined as a description of the elements in the problem; whereas, an operator is any legal move that changes the problem from one state to another. (Simon, 1978, 1979) Moreover, most researchers [5] define a *problem solving path (PS path)* as an sequence of actions/ operators leading from one state to another, and those PS paths able to attain the goal state are called *successful PS paths* here.

The goal of General Problem Solver (GPS) is to find a successful PS path. The approaches to GPS include random trial-and-error, hill climbing and means-ends analysis. [5][3] However, when problem solving is applied in computer-assisted learning, different learners use different PS paths. Although it is obvious that the best (fastest) solving path from initial problem to the goal is the so-called *shortest path*, a more elaborate criterion to evaluate these PS paths is quite needed. This is called *evaluation problem of PS paths*. As the evaluation criterion of PS paths can be found, it is possible to build teacher monitoring system and virtual learning companions in networked learning. [6]

When each problem is assigned a unique (x,y)

*E-mail: jsheh@ice.cycu.edu.tw

coordinate, the problem solving graph becomes a CPSN (Coordinate Problem Solving Network). [7] Continuing the former work, this paper proposes an evaluation vector on CPSN for a given PS path to evaluate its performance. Section 2 will formulate the problem solving graph and then problem solving path. The definition of CPSN is described in Section 3. After discussing some properties of CPSN, Section 4 will propose the definition of evaluation vector and its evaluation principles. Section 5 gives two real examples which processes the PS paths for solving Hanoi tower problem and one-variable linear equation. Section 6 is a conclusion.

II. FORMULATION OF PROBLEM SOLVING

In computer-simulation problem solving, a *problem state* p_i represents the description of the elements in the problem and the set of all problem states constitutes a *problem space* (Newell, 1969; Simon, 1978) or called *state set space* $\Xi=\{p_i\}$. [5] Therefore, the above given or starting conditions are denoted as *initial state or source problem* p_s and the final or goal situation is called *goal state or destination problem* p_d. The changes of the problem from initial state to goal state are through applying possible sequences of *operators* or called *actions* $\prod=\Xi\times\Xi$. [5] These transformations of problems can be illustrated by a directed graph, called *problem solving graph (PSG)*,

$$PSG(P,T), \tag{1}$$

where the problem set $P=\{p_i\}\subseteq\Xi$ and the operator set $T=\{t_k\}\subseteq\Xi$.

An operator in (1) transforms one problem state into another one, that is,

$$p_{i+1}=t_j(p_i) \tag{2}$$

or

$$t_j(p_i,p_{i+1}) = \overrightarrow{p_ip_{i+1}}, \tag{3}$$

which represents an edge in PSG. Assumed that PSG is loop free, such directed edges (2) make a partial ordering, then the predecessor and successor of a problem can be defined:

$$p_i=pred(p_{i+1}) \text{ and } p_{i+1}=succ(p_i). \tag{4}$$

As mentioned, a PS path is defined as a sequence of operators leading from one state to another, that is,

$$\overrightarrow{p_ip_j} = \overrightarrow{p_ip_{i+1}} \cup \cdots \cup \overrightarrow{p_{j-1}p_j} = (t_k(p_i,p_{i+1}),\cdots, t_1(p_{j-1},p_j)). \tag{5}$$

Then a PSG formed by a collection of PS paths can be illustrated by a more sophisticated graph, called *problem solving network (PSN)*,

$$PSN(P,T,p_s,p_d) = \left\{ \overrightarrow{p_sp_d}^{\,i} = (t_k^i,\cdots,t_1^i) \right\}, \tag{6}$$

as Figure 1 shown.

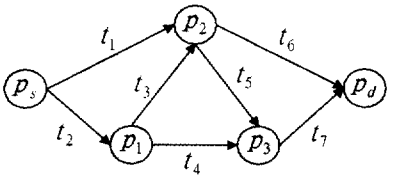

Fig. 1. Problem solving network formed by a collection of PS paths.

To evaluate the performance of problem solving, several cost functions have been defined. For example, *path cost* assigns a cost to a path and is often defined as the number of edges in the path (5) [5] .

$$path\_cost(\overrightarrow{p_ip_j}) = number\{t_k : t_k \in \overrightarrow{p_ip_j}\}. \tag{7}$$

On the other hand, search cost is associated with the time and memory required to find a solution and is always dependent on the circumstance. The combination of the above two costs is called total cost. [5] Through the cost of PS paths, the objective of the evaluation problem of PS paths is to find a criterion for one PS path, such that we can judge the performance of the solving path. The first step of our solution is to make PSG/PSN be a coordinate graph. [7]

III. COORDINATE PROBLEM SOLVING NETWORK (CPSN)

During our discussion, several assumptions have to be made to encircle our problem and to simplify the solution:

1. **Content independent:** we only investigate the general properties of a PSN and its problem states and operators, thus all the information able to be used during evaluation are only provided from the network model $PSN(P, T,p_s,p_d)$.
2. **Uniform cost:** under the above assumption, it is not able to distinguish the costs of operators, then their costs are assumed all equal and the path cost (7) is accepted.
3. **Single source (p_s) and single destination (p_d):** this assumption are only made for convenience and can be easily relaxed to multiple sources and multiple

destinations.

4. **All PS paths are successful:** this means our evaluation criterion are set for those PS paths with all problems reachable from p_s and solvable to p_d.

Given a collection of PS paths $\overrightarrow{p_s p_d}^i$'s, it is easy to obtain a *relative shortest path*, which is the fastest path from source to destination, and then should be placed in the center of PS network. Extended from such idea, the *target offset* of a problem is defined as its shortest path to the goal p_d:

$$offset(p_i) = \min_j \left\{ path\_cost(\overrightarrow{p_i p_d}^j) : \overrightarrow{p_i p_d}^j \in PSN \right\}. \quad (8)$$

Moreover, the *arborescence (directed tree) from* p_d, $ARB(P,T,p_d)$, can be found from the minimal spanning tree algorithm. [2]

Moreover, the farthest problems to the destination p_d are those *terminal problems* of $ARB(P,T,p_d)$, symbolized as:

$$Pterm(ARB(P,T,p_d)) = \{p_i : pred(p_i) = \phi(\textit{null node}) \text{ in } ARB(P,T,p_d)\}. \quad (9)$$

These terminal problems can be sorted in ascending $offset(\bullet)$'s to indicate increasing target offsets of problem solving:

$$order(p_i, ARB(P,T,p_d)) = \begin{bmatrix} p_i \in Pterm(ARB(P,T,p_d)): \\ offset(p_i, ARB(P,T,p_d) \le offset(p_j, ARB(P,T,p_d)), \\ \forall j > i \text{ and } p_j \in Pterm(ARB(P,T,p_d)) \end{bmatrix} \quad (10)$$

With these definitions, we can begin to find the coordinates of problems in CPSN.

Definition 1 CPSN (Coordinate Problem Solving Network)

$CPSN(P,T,p_s,p_d,x,y)$ is one kind of $PSN(P,T,p_s,p_d)$, where the coordinate functions of problems $x:\prod \to R$ and $y:\prod \to R$ are defined as:

$$x(p_i) = -offset(p_i) \quad (11)$$

$$y(p_i) = \begin{cases} order(p_i, \cdot) - order(p_s, \cdot), \\ \text{when } p_i = terminal\ node \\ \min\{order(pred(p_i), ARB(P,T,p_d))\}, \\ \text{when } p_i = nonterminal\ node. \end{cases} \quad (12)$$

From the above definition, we can have the following algorithm for constructing CPSN from a series of PS paths. And Figure 2 gives an example mapping a PSN to its CPSN.

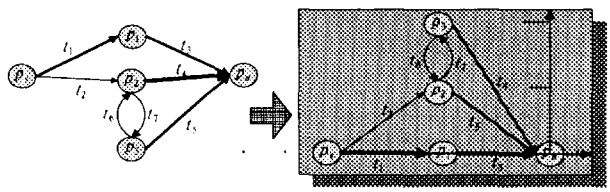

Fig. 2. An example from PSN to CPSN.

Algorithm 1 (CPSN Construction from Problem Solving Paths)

0. Given the source problem p_s and the destination problem p_d.
1. Obtain a series of problem solving path from source p_s to destination p_d, $\left\{ \overrightarrow{p_s p_d}^i = (t_k^i, \cdots, t_1^i) \right\}$.
2. Form the corresponding problem solving network $PSN(P,T,p_s,p_d)$.
3. Find the target offsets (8) for each problem.
4. Find the incoming arborescence with the root p_d, $ARB(P,T,p_d)$ by Dijsktra algorithm. [2]
5. Get the terminal problem*Pterm*s in $(ARB(P,T,p_d)$ from (9).
6. Sort to get the order of $Pterm$ $(ARB(P,T,p_d))$ (10) by the target offsets in step 3.
7. Find the (x,y)-coordinates of terminal problems and then determine the coordinates of non-terminal problems from (11) and (12).

IV. EVALUATION OF PROBLEM SOLVING PATHS

Obviously, the source problem is located on x-axis and the destination problems on y-axis: [7]

$$(x(p_s), y(p_s)) = (-\text{length}(\overrightarrow{p_s p_d}), 0) \quad (13)$$

and

$$x(p_d) = 0. \quad (14)$$

Hence, the solving flow of the resultant CPSN will keep from left to right. Moreover, when the range of a function is defined as:

$$|f| = \max_z \{f(z)\} - \max_z \{f(z)\}, \quad (15)$$

the ranges of (x,y)-coordinates in a CPSN can be found as follows.

Lemma 1 In $CPST(P,T,p_s,p_d,X,Y)$, we have **(A).** $|x| = \max_i \{offset(p_i)\}$ and **(2).** $|y| = \#(Peterm(ARB(P,T,p_d))$.

Proof. Obvious for $|x| = \max_i \{x(p_i)\} - \min_i \{x(p_i)\} = 0 - \min_i \{x$

$(p_i)\}$=max$\{offset(p_i)\}$ and $|y|$=max$\{y(p_i)\}$-min$\{y(p_i)\}$=#$(Pterm(ARB(P,T,p_d)))$. ❑

Besides, the following theorem ensures there is no two nodes occupying the same (x,y) location.

Theorem 1 In a CPSN, no two different problems, no matter whether terminal or nonterminal, occupy the same coordinate.

Proof. That is to say, if $p_i \neq p_j$, $(x(p_i), y(p_i)) \neq (x(p_j, y(p_j))$. The proof can be divided into three cases: (1). When both p_i and p_j are terminal problems, by ordering, their y-coordinates must be different if $p_i \neq p_j$. (2). When p_i is terminal problem and p_j is non-terminal problem, $y(p_j)$=min$(y(pred(p_j)))$=$y(p_j^*)$, where p_j^* is one of p_j's predecessors and also a terminal problem. However, the fact that $p_i \neq p_j$ makes $y(p_j^*)=y(p_j)=y(p_i)$ a contradiction. (3). The case that p_i and p_j are both non-terminal is similar to the condition (2). Let p_i^* (or p_j^*, respectively) is one of p_is (or p_j's) predecessors and also a terminal problem with $y(p_i)=y(p_i^*)$ and $y(p_j)=y(p_j^*)$. Hence, the fact that $p_i \neq p_j$ makes $y(p_i^*)=y(p_i)=y(p_j)=y(p_j^*)$ a contradiction. ❑

By the properties of arborescence, all problems in a CPSN are located at their shortest paths (minimal spanning tree, nearest locations) to the destination problem. Then we can have the following properties

Corollary 1 **(1).** There is no path from problems at level k to problems at level $(k+2)$. (2). Any path from problems at level k to problems at level $(k+1)$ is also a shortest path to the goal. [7]

With the above properties, the evaluation of a given problem solving path can be made through our CPSN.

Definition 2 (Evaluation Vector)

Let $p = \overrightarrow{p_s p_d} = \overrightarrow{p_1 p_2} \cup \cdots \cup \overrightarrow{p_{n-1} p_n}$, where p_1=ps and $p_n=p_d$ be a PS path from the source to destination p_d. Then the *evaluation vector* of $CPSN(P,T,p_s,p_s,x,y)$ in is defined as:

$$e=Eval(p, CPSN(P,T,p_s,p_d,x,y))=[e_i=x(p_{i+1})-x(p_i): i=1,...,n-1] \quad (16)$$

With evaluation vector, we can have the following evaluation principles of a problem solving path (p).

Evaluation Principles

1. $e_i \geq 2$:$t_j = \overrightarrow{p_i p_{j+1}}$ of is better than those (problem solving) operators in $CPSN(P,T,p_s,p_d,x,y)$
2. $e_i=1$:$t_j = \overrightarrow{p_i p_{i+1}}$ of is also located on another shortest path as $CPSN(P,T,p_s,p_d,x,y)$
3. $e_i \leq 0$:$t_j = \overrightarrow{p_i p_{i+1}}$ of is worse (farther) than those operators in $CPSN(P,T,p_s,p_d,x,y)$

According to Evaluation Principle 3, we can define the deviation of a problem solving path and its deviant operators.

Definition 3 **(Deviant Operator)** Any operator (chord) from problems at offset level k to problem at offset level $(k-m+1)$, where $m \geq 1$, is called a *deviant operator with deviation m*.

It is notable that a deviant operator with deviation m corresponds to the evaluation vector with $e_i=1-m$.

V. EVALUATION EXAMPLES

Example 1 (Hanoi Tower)

From the above-mentioned evaluation vector and its evaluation principles, it is possible to evaluate a problem solving path through CPSN. The first selected example is the popular problem solving example *Hanoi tower* with three disks. The given source problem is that three disks are all on the left stick and the goal state is that three of them are all on the right stick.

Dispatching the problem to different junior high-school students, we can have some problem solving results, like the following list:

- PS path A: 1-9-3-4-5-6-10-7-8-11-2.
- PS path B: 1-9-12-13-14-5-6-10-15-16-11-2.
- PS path C: 1-3-4-14-5-6-10-15-17-16-11-2.
- PS path D: 1-9-12-18-13-14-5-6-7-8-2.

The corresponding problem state of each number is shown in Figure 3. These PS paths can form a PSN. Moreover, we apply Definition 1 to find its corresponding CPSN, as Figure 3 shown.

From Definition 2, we can calculate the corresponding evaluation vector for all the steps in each PS path:

- PS path A: e = [0 1 1 1 1 0 1 1 0 1]
- PS path B: e = [0 0 1 1 1 1 1 0 0 1 1 1]
- PS path C: e = [1 1 0 1 1 1 0 0 0 1 1 1]
- PS path D: e = [0 0 0 1 1 1 1 1 1 1 1]

Then, we can interpret these evaluation vectors and make the following observations for evaluation.

1. These PS paths are only partial solutions; however, the

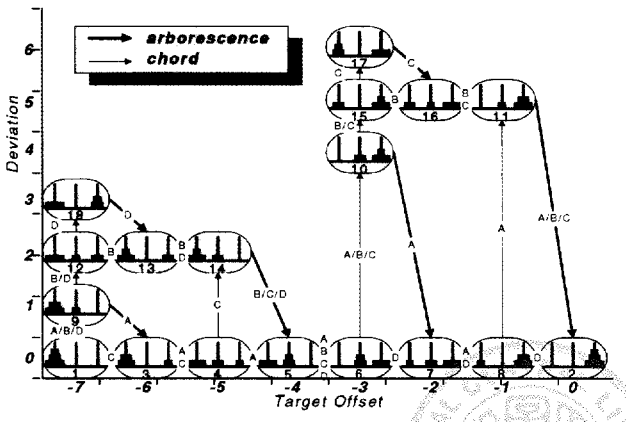

Fig. 3. CPSN of Hanoi tower example.

CPSN in Figure 3 shows a large problem space of real problem and the possible evaluation of problem solving paths.

2. All paths possess evaluation $e_i \geq 1$, then the largest possible deviation is $m=1$ (upward chords); this indicates that the solver will not make a large mistake ($m \geq 1$) but maybe have a series of detours.
3. Particularly, those transformations with $e_i=1$ are located on the arborescence; whereas, those with $e_i=0$ are chords.
4. The solving patterns of the solvers A, B, C and D can be found out from the corresponding evaluation vectors:

- PS path A has some (three) non-continuous deviations (**0**) since **e** = [**0** 1 1 1 1 **0** 1 1 **0** 1].
- PS path B has 2 continuous deviations (**0 0**) since **e** = [**0** 0 1 1 1 1 **0 0** 1 1 1].
- PS path C has a burst deviation.(**0 0 0**) since **e** = [1 1 **0** 1 1 **0 0 0** 1 1 1].
- PS path D has an initial deviation (**0 0 0**), but the following procedure is good since **e** = [**0 0 0** 1 1 1 1 1 1 1].

5. The similarity of two groups of problems: 1,9,12,18,3,13,4,14,5 and 6,10,15,17,7,16,8,11,2 with the critical solving step 5→6 constitute an important strategy of this problem solving:
   (i). **moving smaller two disks from left (1) to middle (5):** Problems 1,9,12,18,3,13,4,14,5.
   (ii). then **moving the largest disk from left (5) to right (6):** Problem 5→6.
   (iii). and then **moving the smaller two disks from middle (6) to right (2):** Problems 6,10,15,17,7,16,8,11,2. (similar to step (i))

Example 2 (One-Variable Linear Equation)

A more practical example is *one-variable linear equation* of the form:

$$ax+b=cx+d, \tag{17}$$

where a, b, c and d are all real numbers. Such problem is taught in junior high school and is also a first step for student to solve equations. The given source problem is

$$3x-2=4x+3 \tag{18}$$

and the goal state is

$$x=-5. \tag{19}$$

As this problem is solved by different junior high-school students, we can have some problem solving results, like the following list:

- PS path A: $(3x-2=4x+3)\rightarrow(3x-5=4x)\rightarrow(-x-5=0)\rightarrow(-x=5)\rightarrow(-2x=10)\rightarrow(x=-5)$
- PS path B: $(3x-2=4x+3)\rightarrow(-2=x+3)\rightarrow(-5=x)\rightarrow(x=-5)$
- PS path C: $(3x-2=4x+3)\rightarrow(-2=x+3)\rightarrow(-x-2=3)\rightarrow(-x=5)\rightarrow(x=-5)$
- PS path D: $(3x-2=4x+3)\rightarrow(-x-2=3)\rightarrow(-x=5)\rightarrow(x=-5)$
- PS path E: $(3x-2=4x+3)\rightarrow(3x-5=4x)\rightarrow(-5=x)\rightarrow(x=-5)$

These PS paths can form a PSN. Moreover, we apply Definition 1 to find its corresponding CPSN, as Figure 4 shown.

From Definition 2, we can calculate the values deviations m's for all the steps in each PS path:

- PS path A: 1 0 1 0 1
- PS path B: 1 1 1
- PS path C: 1 0 1 1
- PS path D: 1 1 1
- PS path E: 1 1 1

Then, we can make the following observations:

1. The solvers B, D, E have the shortest path solution, although their approaches are quite different.
2. Problems $(-x=5)$ and $(-5=x)$ have the same target offset.
3. Problems $(-x-2=3)$, $(-x-5=0)$ and $(-2x=10)$ are all derived by deviant paths [$(-2=x+3)\rightarrow(-x-2=3)$, $(3x-5=4x)\rightarrow(-x-5=0)$, and $(-x=5)\rightarrow(-2x=10)$]. But the first problem $(-x-2=3)$ is pointed by another shortest path $(3x-2=4x+3)\rightarrow(-x-2=3)$ with $m=1$, so it is not a deviant path. The other two problems can be treated as deviant because

- the problem $(-x-5=0)$ lets both first-order and constant terms be located on the same side of equality.
- the problem $(-2x=10)$ makes a redundant multiplication on both sides.

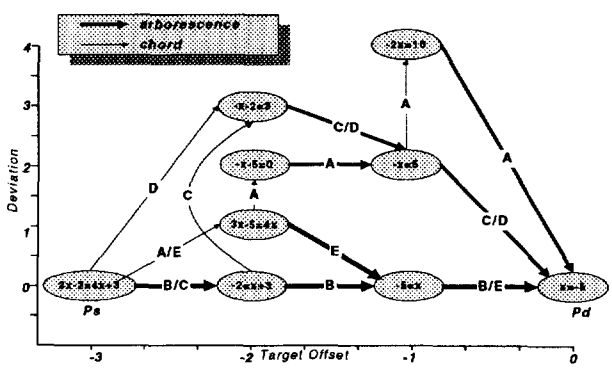

Fig. 4. Resultant CPSN in Example 2.

VI. CONCLUSION

When computer is used to simulate the process of problem solving, it can be modeled as a procedure of transforming a source problem to another destination problem. A collection of PS paths can be illustrated by a problem solving network. As each problem is assigned a unique (x,y) coordinate, the problem solving graph becomes a

CPSN (Coordinate Problem Solving Network).

This paper proposes an evaluation vector on CPSN for a given PS path to evaluate its performance and describes the evaluation principles. For a problem, the uniqueness of (x,y)-coordinate is proven in Section 4. The evaluation vector is formed from the process of PS paths in CPSN. Each element of this vector indicates the performance of the applied operator of each solving step. From the practical examples of Hanoi tower and one-variable linear equation, it can be concluded that evaluation vector can discover many evaluation results for the problem solving.

REFERENCES

1. R. E. Richard, *Thinking, Problem Solving, Cognition*, 2nd ed., W. H. Freeman and Company, 1992.
2. K. Thulasiraman and M. N. S. Swamy, *Graphs: Theory and Algorithms*, John Wiley & Sons, Inc., New York, 1992.
3. L.-Y. Cheng, *Cognitive Psychology–theory and application*, Wu-Nan Book Co., Taipei, 1993.
4. R. G. Parker, *Deterministic Scheduling Theory*, Chapman & Hall, London, 1995.
5. S Russell and P. Norvig, *Artificial Intelligence–A Modern Approach*, Prentice-Hall, 1995.
6. J.-L. Lu, K.-Y. Chu, T.-W. Chan, K.-Y. Yang and J.-S. Heh, "Social Learning Systems for Mathematical Problem Solving," *13th Science Education Workshop*, Taipei.
7. C.-J. Hsu, J.-L. Lu and J.-S. Heh, "Problem Solving Evaluation through Coordinate Graph," *2nd Global Conference on Chinese Computer Education*, Hong Kong, June 1998.

電腦模擬問題解決法之評估

賀嘉生

中原大學資訊工程學系

台灣省中壢市普仁22號

許呈如

崇右企專資管科

摘 要

問題解決法是一種將指定的來源題目，轉換爲目標題目狀態的程序。一連串這種將來源轉爲目標的轉換，可看作是一條解題路徑；而在電腦中，一群解題路徑可表示爲一種有向圖，稱作解題網路。

本論文提出一種在座標化解題網路上的評估向量，用來要評估解題路徑的表現，並說明其評估的準則。在座標化解題網路中，每個問題具有唯一的一組(x,y)座標值，其中x座標表示該問題的目標差距，而y座標則爲到最短路徑的路徑偏移。對於任意給予的解題路徑，我們可以求得一組評估向量，用來指示每個解題步驟的表現。而評估向量中的每個元素的數值，對應到在每個解題步驟中所採取轉換運算的判定指標。經由所提供的評估準則，了解該判定指標對解題表現的意義。在文末河內塔的實例中可以看出：評估向量的確對問題解決法，提供相當多的評估結果。

關鍵詞：問題解決法、解題路徑、座標化解題網路、評估結果。