Analyzing Problem's Difficulty based on Neural Networks and Knowledge Map

Rita Kuo
Dept. of Electronic Engineering, Chung-Yuan Christian University
No.22, Pu-Jen,Pu-chung Li, Chung-Li, 320, Taiwan
rita@mcsl.ice.cycu.edu.tw

Wei-Peng Lien
Dept. of Visual Communication Design, Shu-Te University
No.59, Hengshan Rd., Yanchao Township, Kaohsiung County, 824, Taiwan
apinh@hotmail.com

Maiga Chang
Dept. of Special Education, Chung-Yuan Christian University
No.300, Jungda Rd, Chung-Li, 320, Taiwan
maiga@ms2.hinet.net

Jia-Sheng Heh
Dept. of Electronic Engineering, Chung-Yuan Christian University
No.22, Pu-Jen,Pu-chung Li, Chung-Li, 320, Taiwan
jsheh@ice.cycu.edu.tw

Abstract

This paper proposes a methodology to calculate both the difficulty of the basic problems and the difficulty of solving a problem. The method to calculate the difficulty of problem is according to the process of constructing a problem, including Concept Selection, Unknown Designation, and Proposition Construction. Some necessary measures observed in the problem construction process are also defined in this paper in order to formulate and calculate the difficulties. Beside the difficulty of the basic problem, four difficulty dimensions for problem solvers to realize what kinds of abilities they are lack of to deal with the problem, including Identification, Elaboration, Planning, and Execution, corresponding to the each step of problem solving process are also analyzed and designed by the artificial neural networks in this paper. By these difficulty measures learners can understand what kind of problems they meet and what sort of problem solving strategies they use in solving the problem. To verify our goals, an Item Generating System is constructed for demonstrating and supporting the difficulty calculation in the end of this paper.

Keywords

Difficulty of problems, Knowledge map, Neural networks, Problem solving process, Least-mean square

Introduction

Knowledge Storage is an important issue in CAI. When suitable knowledge structure is designed, the CAI system can be used to proceed with tutoring, solving problems, or misconception diagnosing. The knowledge structure, which used here, is Knowledge Map. Knowledge Map consists of two major parts, Concept Hierarchy and Concept Schema (Kuo et al., 2002). Concept Hierarchy presents the hierarchical relationship among concepts. Concept Schema stores the remaining information, which integrates the definition, example, and other relations with associated concepts. In Figure 1, the Concept Hierarchy is demonstrated in gray block with link among concepts and the Concept Schema of each concept is demonstrated in the white block.

The *Knowledge Object* is able to use for representing knowledge (Tung, 2002). For example, Physics has four Knowledge Objects as shown in Figure 2: Object, Physics Phenomenon, Physics Law, and Physics Quantity. Each Knowledge Object has some relationships to its related Knowledge Object. For example, a Physics Phenomenon changes according to some Physics Laws and may influence some values of Physics Quantity. The *Core Knowledge Object*, which expressing main idea of the knowledge, is the Physics Phenomenon in this example.

This research uses previous the Knowledge Object to formulate the difficulty of a problem. The similar study in problem difficulty determination is *computerized adaptive testing* (CAT). CAT selects items from item bank,

and the difficulty setting in each item which needs to be adjusted according to *item response theory*. The difficulty-setting strategy proposed in this paper use knowledge analysis based on *Knowledge Map* and increase difficulty reliability by neural network training.

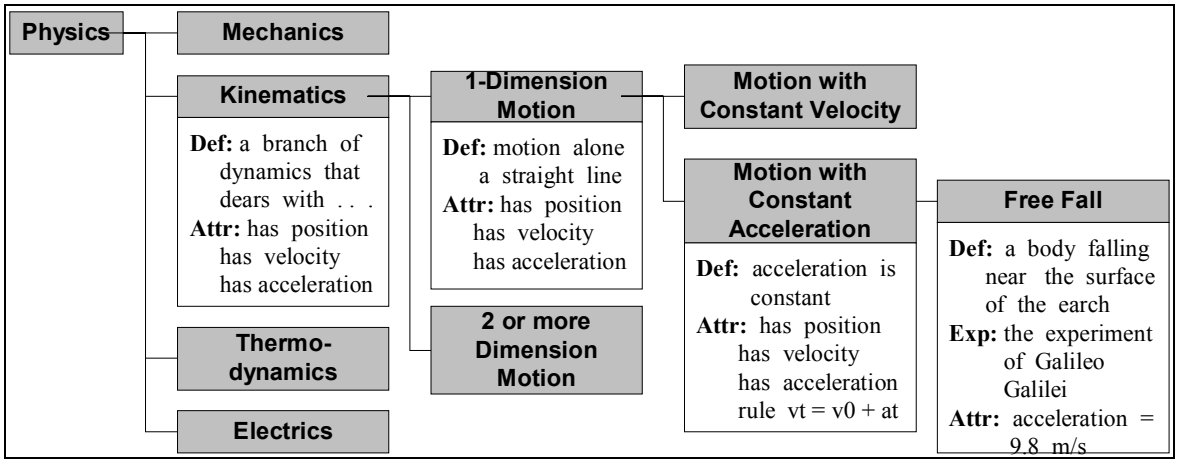


Figure 1. Example of Knowledge Map in Physics

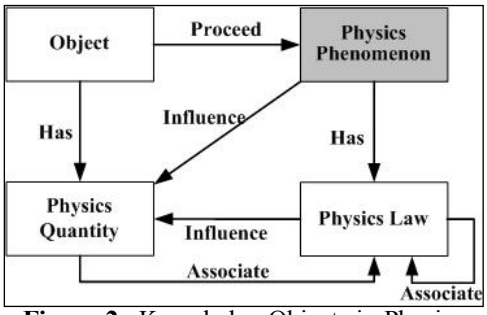


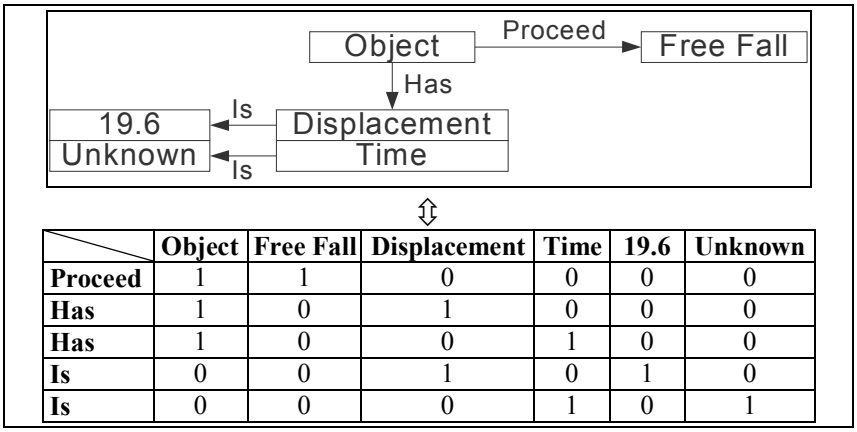
Figure 2. Knowledge Objects in Physics

Section 2 describes two metaphors for problem solving and the four-step problem solving process. The problem model and its construction process are analyzed in Section 3. Section 4 discusses the calculation of difficulty features and difficulty dimensions. An Item Generating System (IGS) which is given as an example for showing the difficulties calculations is built in Section 5. Section 6 gives a brief conclusion and some possible future studies.

Metaphors for Problem Solving

It is a problem when someone has a goal but the goal is blocked for lack of information resources (Kahney, 1993). Two metaphors, *Problem Graph* and *Problem Matrix* can be represented for solving a problem (Kuo et al., 2003). Figure 3 gives an example of *Problem Graph*, which use graph structure to illustrate a problem. There are six concepts ("Object", "Free Fall", "Displacement", "Time", "19.6", "unknown") and five links (one "Proceed", two "Has", and two "Is") compose a problem graph. These concepts and links make up five propositions and these propositions can be transformed to the related proposition matrix as shown in the bottom of Figure 3.

Another metaphor, *Problem Matrix*, focuses on the manipulation of concepts and relations which can be used for solving the problem. Figure 4 shows a Problem Matrix example for the Free Fall problem in Physics. Two Physics laws ("Displacement = 0.5 * Acceleration * Time ^ 2" and "Velocity = Acceleration * Time") present the relationships among four major concepts of Physics Quantity ("Displacement", "Time", "Velocity" and "Acceleration"). The *Problem Matrix* for such kinds of problems can be written as Figure 4 shown below.



	Object	Free Fall	Displacement	Time	19.6	Unknown
Proceed	1	1	0	0	0	0
Has	1	0	1	0	0	0
Has	1	0	0	1	0	0
Is	0	0	1	0	1	0
Is	0	0	0	1	0	1

Figure 3. Example of the Problem Graph and related proposition matrix

	Displacement	Time	Velocity	Acceleration
Displacement = 0.5 * Acceleration * Time ^ 2	1	1	0	1
Velocity = Acceleration * Time	0	1	1	1

Figure 4. Example of Problem Matrix

With these two problem metaphors described above, a four-steps problem solving strategy proposed in previous study (Hsu et al., 2002). Since problem solving has already been discussed for a long while from 1910 (Deway, 1910; Polya, 1965; Deek et al., 1999), those steps took in this paper will follow the four types of schema in problem solving proposed by Marshall in 1995 (Marshall, 1995). By using these schemas, a problem solving system was constructed in our previous works and its system architecture in Figure 5 (Cheng et al., 2001; Tung, 2002; Kuo et al., 2002). A Problem Solving System loads a problem, which will be processed through four steps: problem identification, problem elaboration, problem planning, and problem execution. All these steps get knowledge from the Knowledge Base which stores in the long-term memory and produces different metaphors, Problem Graph and Problem Matrix store in short-term memory.

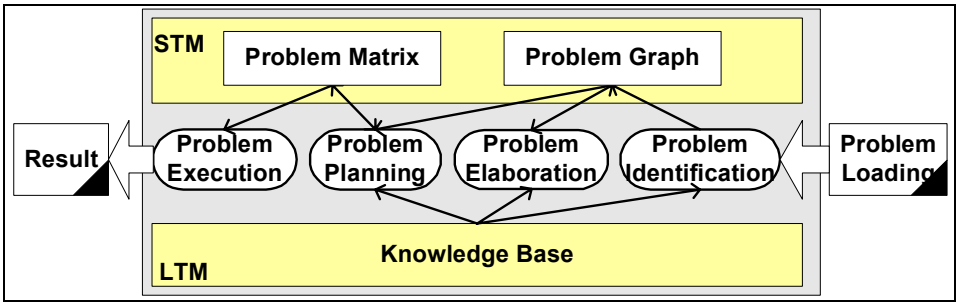


Figure 5. Four-steps of Problem Solving

With the problem metaphors and the problem solving strategies, this study proposes a difficulty definition corresponding to each stage in the process of problem solving. Since the difficulty of problems depends on what kind of problems the students dealing with, therefore, before we talking about what is difficulty and how to calculate the difficulty of problems, the problem model and the process of constructing a problem should be designed and analyzed first.

Problem Model and Difficulty Analysis

Problem Model Design

Most of the problems can be separated into several sub-problems. A problem which can be separated is called complex problem. On the contrary, a problem can not be separated is called simple problem (or basic problem). The basic problem in this paper is defined as there is only one *Core Knowledge Object* existed in the problem. Furthermore, a problem also has some unknown attributes (the goal of the problem) and given attributes (the resource for reaching the goal).

To sum up the description above, basic problem model has three major parts: *Sub-Problem Flag Attribute*, *Given Attributes* and *Unknown Attributes*, which *Sub-Problem Flag Attribute* presents the *Core Knowledge Object*. Figure 6 is an example of Free Falling problem with given attributes (*distance* and *velocity*) and the unknown attribute (*time*).

A complex problem is composed by several basic problems. The relations among basic problems are defined as *Sub-Problem Connection*. The Sub-Problem Connection contains three parts: two related attributes for the different basic problems and the relation type between these two basic problems. Figure 7 illustrates a complex problem model with two sub-problems, which has one Sub-Problems Connection for linking both of sub-problems.

	Corresponding Concepts	Example of Physics Problem
Basic Problem Model	*Sub-Problem Flag Attribute*	*Free Fall* Phenomena
	Given Attributes	Value of *Distance* Value of *Velocity*
	Unknown Attributes	Ask for Value of *Time*

Figure 6. Basic problem model for the Free Falling problem in Physics

Basic Problem 1 Model	Sub-Problem Flag Attribute
	Given Attributes
	Unknown Attributes
Sub-Problems Connection	Related Basic Problem 1 Attribute
	Related Basic Problem 2 Attribute
	Relation between two Attributes
Basic Problem 2 Model	Sub-Problem Flag Attribute
	Given Attributes
	Unknown Attributes

Figure 7. Problem model of two sub-problems (Complex Problem Model)

Using the problem model designed above to construct a problem we need three major steps, they are *Concept Selection*, *Unknown Setting*, and *Proposition Construction* as Figure 8 shown below. *Concept Selection* sets the *Sub-Problem Flag Attribute* of the basic problem. This action decides the main concept of the problem. The second step, *Unknown Setting*, determines the *Given Attributes* and the *Unknown Attributes* in the problem. The way to find the given and unknown attributes depends on the problem metaphor – *Problem Matrix*.

Problem Matrix defines the possible manipulation way of the relations and concepts of the problem. Some concepts can be set as the *Unknown Attributes* and some can be set as the *Given Attributes*. This setting will be judged if it is solvable or unsolvable by using the *Problem Matrix*. The last step is *Proposition Construction*. Using the setting of attributes in the *Problem Matrix*, the *Proposition Construction* step could be able to build question sentences in natural language. All these three steps need the *Knowledge Map* introduced in Section 1.

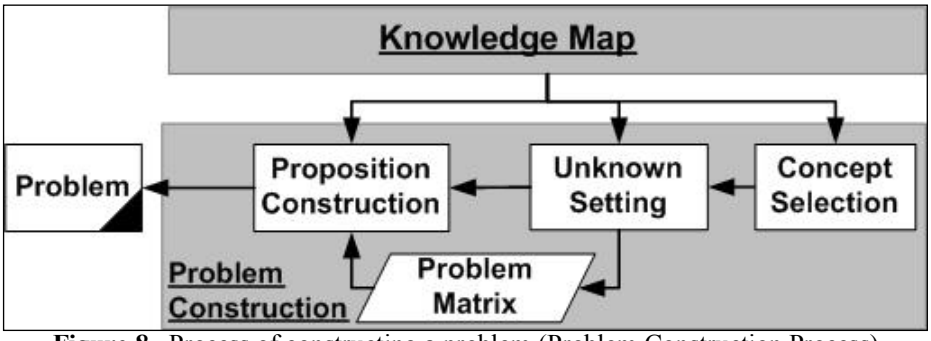


Figure 8. Process of constructing a problem (Problem Construction Process)

Difficulty Analysis

Each step of constructing a problem makes the difficulty of the problem changes. Ignoring the reading ability of learners, this paper only focuses on the first two steps of problem construction process. In *Concept Selection*,

obviously, the *number of sub-problems* will influence the difficulty of the problem. *Number of Needed Attributes* indicates how many concepts are needed to learn before students can solve such kinds of problem, and it also influences the next two features, *Learning Sequence* and *Concept Depth*. The *Learning Sequence* is a difficulty feature that indicates the learning order for learners and the *Concept Depth* shows the specialization degree of the core concept of the problem.

The second step, *Unknown Setting*, affects all attributes in the problem. *Number of Unknown* is one difficulty feature which is usually used in measuring problem. *Number of Given Attribute* indicates the information that this problem supports. *Number of Elaborating Attributes* shows the hidden information cover up in the problem. Attributes setting also influence *Mathematical Complexity*, which can be another difficulty feature of the problem. All difficulty features are arranged in Table 1. Disregarding the mathematical complexity this paper focuses on the basic problem model, it means the *Number of Sub-Problems* and *Mathematical Complexity* will be omitted in the following Section.

Table 1. Difficulty Features and its denotation

Element	Difficulty Feature	Denotation
Selected Concepts	Number of Sub-Problems	γ_{sub_prob}
	Number of Needed Attributes	γ_{need_attr}
	Learning Sequence	γ_{learn_seq}
	Concept Depth	γ_{cpt_depth}
Attributes Setting	Number of Unknown	$\gamma_{unknown}$
	Number of Given Attributes	γ_{given_attr}
	Number of Elaborating Attributes	γ_{elb_attr}
	Mathematical Complexity	γ_{math_cpx}

Difficulty Calculation

Difficulty of the Basic Problems

The eight difficulty features discussed in the previous section compose the problem difficulty. This section concentrates on the calculation of each difficulty feature for the *Basic Problems*, hence, the *Number of Sub-Problems* and *Mathematical Complexity* are ignoring. Some definitions of prior measures are required for difficulty calculation and listed below. Most of them are related to the *Knowledge Map*.

#given_attr:	number of given attributes in the problem
#unknown:	number of unknown attributes in the problem
root:	the root of *Knowledge Map*
θ_i:	*i*th concept
$CS(\theta_i)$:	*Concept Schema* of *i*th concept
$size(CS(\theta_i))$:	number of attributes in concept from full *Concept Schema*
$height(\theta_i)$:	the height of sub-tree from *i*th concept
$attr_{LS}(CS(\theta_i))$:	learning sequence attribute stored in *Concept Schema*
$attr_{manip\_cpt}(\theta_i)$:	attributes extracted to manipulating concepts in *Problem Matrix*
$size(attr_{manip\_cpt}(\theta_i))$:	number of manipulating concepts in *Problem Matrix*
$attr_{max\_unknown}(\theta_i)$:	maximum number of unknown in *i*th concept

Using the measurement listed above, each difficulty feature of problems can be formulized as below:

$$\gamma_{need\_attr} = size(CS(\theta_i)) / \arg\max_{j \in KM} size(CS(\theta_j))$$
$$\gamma_{learn\_seq} = attr_{LS}(\theta_i) / \arg\max_{j \in sibling(\theta_i)} attr_{LS}(\theta_j)$$
$$\gamma_{cpt\_depth} = (height(root) - height(\theta_i)) / height(root)$$
$$\gamma_{given\_att} = (size(attr_{manip\_cpt}(\theta_i)) - \#given\_attr + 1) / size(attr_{manip\_cpt}(\theta_i))$$
$$\gamma_{unknown} = \#unknown / attr_{max\_unknown}(\theta_i)$$
$$\gamma_{elb\_attr} = (size(attr_{manip\_cpt}(\theta_i)) - \#given\_attr - \#unknown + 1) / size(attr_{manip\_cpt}(\theta_i))$$

Later part of the paper will give a simple example for calculating each difficulty features.

Difficulty Dimensions for Problem Solvers

Although the difficulty features corresponding to the problem metaphors and the problem construction process is analyzed and designed in this paper, learners still cannot use them easily to figure out what is going wrong while they are solving problem. Therefore, the previous discussed problem solving steps are taken for this paper to design four difficulty dimensions for problem solvers to realize what kinds of abilities they are lack of to deal with the problem, including *Identification Difficulty* (γ_{idf}), *Elaboration Difficulty* (γ_{elb}), *Planning Difficulty* (γ_{pln}), and *Execution Difficulty* (γ_{exc}). All the values of these four difficulty dimensions come from the setting of difficulty features.

Because the *Identification Difficulty* focuses on the concept discussed in the problem, there are two difficulty features related to this dimension – *Learning Sequence* and *Concept Depth*. The *Elaboration Difficulty* dimension emphasizes the attributes, *Number of Needed Attributes*, *Number of Given Attributes*, and *Number of Elaborating Attributes*, which are needed to solve the problem. *Number of Unknown* and *Number of Elaborating Attributes* impact the *Planning Difficulty* dimension; therefore the planning difficulty dimension needs students to know the relation of unknown attributes in the problem. *Number of Unknown* also decides the difficulty dimension of *Execution Difficulty*. To transform difficulty features to difficulty dimensions, the transform formulae are listed below: (ω_{ij} indicates the customized weight)

$$\gamma_{idf} = \omega_{21} * \gamma_{learn\_seq} + \omega_{31} * \gamma_{cpt\_depth}$$
$$\gamma_{elb} = \omega_{12} * \gamma_{need\_attr} + \omega_{42} * \gamma_{given\_attr} + \omega_{62} * \gamma_{elb\_attr}$$
$$\gamma_{pln} = \omega_{53} * \gamma_{unknown} + \omega_{63} * \gamma_{elb\_attr}$$
$$\gamma_{exc} = \omega_{54} * \gamma_{unknown}$$

All these weights can be set and adjusted by training the neural networks. A neural network is defined as a machine which modeling a particular task of function (Haykin, 1999). This research applies *least-mean square algorithm* for training customized weights which transforming difficulty features to difficulty dimensions. There are four linear filters of neuron model in the network as shown in Figure 9. In the previous definition *Identification Difficulty* have two influence features, *Learning Sequence* and *Concept Depth*. This idea is constructed in the first neuron model (N_1), which has one output (γ_{idf}), two inputs (γ_{learn_seq}, γ_{cpt_depth}), and two synaptic weights (ω_{21}, ω_{31}). Other three neuron models also have similar construction. With supervised by teachers, the weights among neurons could be changed possibly. Because of the teacher may find out the difficulty dimension faced by the students (problem solvers) are different to the output of neural networks as Figure 9 shown below.

Although using neural network can find objective difficulty features, those values can not be adjusted according to different capability learners. Two possible solutions can deal with this problem. One of them is problem-process record, which registers each problem solving steps from learners including previous solving problem. The other one is *item response theory* which mentioned in the beginning of this paper. The critical point of integrating *IRT* into difficulty setting is the limitation of unidimensionality in *IRT*.

Item Generating System

In previous research, the Item Generating System needs Problem Matrix to construct a problem (Kuo et al., 2003). To integrate with the difficulty calculation the system needs another data structure, Difficulty Table, for estimating problem difficulty as shown in Figure 10.

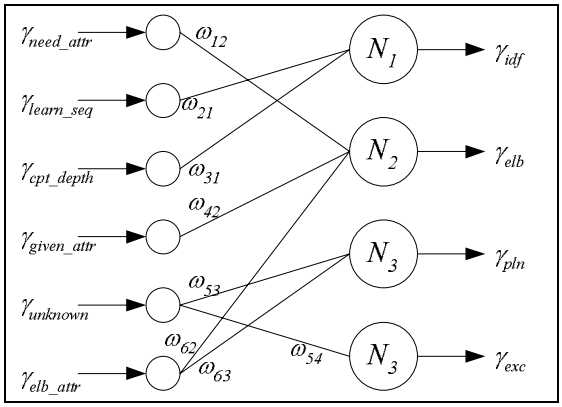


Figure 9. Training Weights of Neural Network

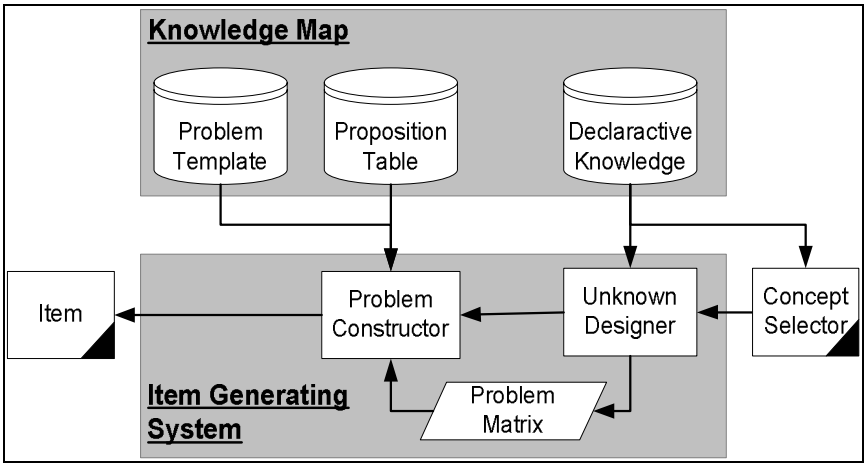


Figure 10. Problem Generating System with difficulty calculation

Figure 11 shows a snapshot of the generated problem: "In sky, raindrop exceeds the motion of constant acceleration. The initial velocity is 0 m/s. Acceleration is 5 m/s^2. The final time is 5 s. Ask for the final value of velocity." The *Knowledge Map* in the example is the sub-tree of Kinematics in Figure 1. This problem has one unknown and three given attributes. In the *Knowledge Map*, there are eleven attributes in *Concept Schema* of the "Motion of Constant Acceleration"; seven of them are manipulating concepts and three of them are manipulating relations. Those manipulating attributes construct the *Problem Matrix* as Figure 12. According to these settings, six difficulty features are listed below in Table 2.

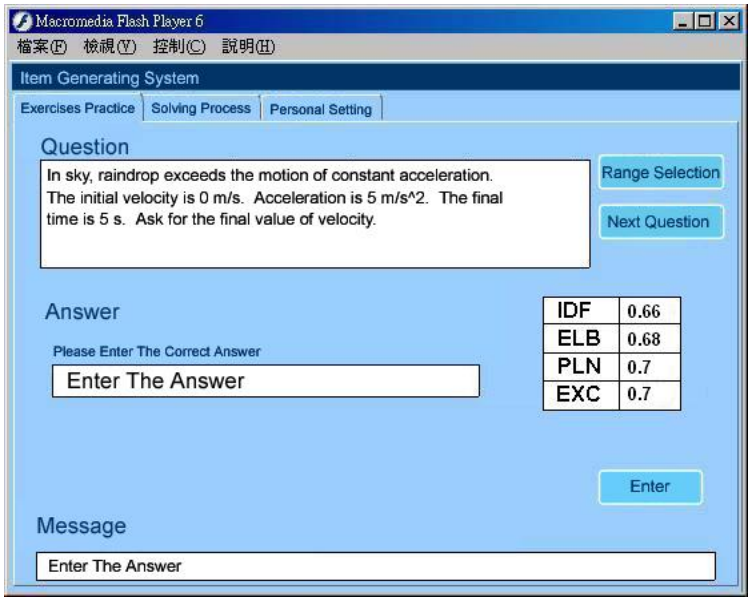


Figure 11. Difficulties of problem solving steps calculated by the Item Generating System

	init distant	final distant	init velocity	final velocity	init time	final time	accelary
	d_i	d_f	v_i	v_f	t_i	t_f	a
		unknown	0			5	5
$v_f = v_i + a * (t_f - t_i)$	0	0	1	1	1	1	1
$d_f = d_i + v_i * (t_f - t_i) + 0.5 * a * (t_f - t_i)^2$	1	1	1	0	1	1	1
$v_f^2 = v_i^2 + 2 * a * (d_f - d_i)$	1	1	1	1	0	0	1

Figure 12. Problem Matrix in example item

Table 2. Six Difficulty Features of the example

$\gamma_{need\_attr} = 11 / 13 = 0.8$
$\gamma_{learn\_seq} = 2 / 2 = 1$
$\gamma_{cpt\_depth} = (4 - 2) / 4 = 0.5$
$\gamma_{given\_attr} = (7 - 3 + 1) / 7 = 0.7$
$\gamma_{unknown} = 1 / 2 = 0.5$
$\gamma_{elb\_attr} = (7 - 3 - 2 + 1) / 7 = 0.4$

After getting those difficulty features, four difficulty dimensions could be transformed into difficulty dimensions. The weights are trained by neural networks which as shown in Figure 9, and the weight values among neurons are:

$\omega_{21} = 0.7122$	$\omega_{62} = 0.3815$
$\omega_{31} = 1.6557$	$\omega_{53} = -0.8568$
$\omega_{12} = 0.9500$	$\omega_{63} = 0.5786$
$\omega_{42} = -0.0676$	$\omega_{54} = -0.9987$

And the values of difficulty dimensions can be calculated as follow:

$\gamma_{idf} = 0.7122 * 1 + 1.6557 * 0.5 = \underline{\mathbf{0.66}}$
$\gamma_{elb} = 0.9500 * 0.8 - 0.0676 * 0.7 + 0.3815 * 0.4 = \underline{\mathbf{0.68}}$
$\gamma_{pln} = - 0.8568 * 0.5 + 0.5786 * 0.4 = \underline{\mathbf{0.7}}$
$\gamma_{exc} = - 0.9987 * 0.5 = \underline{\mathbf{0.7}}$

These four difficulty dimensions can help learners to understand the emphatic problem solving techniques they need to have when solving the problem.

Conclusions

This study proposes a methodology for calculating the problem difficulty based on the four-step problem solving strategy. This difficulty helps learners realize the emphasized problem solving strategy in the basic problem. To calculate the difficulty of the problem, this paper first analyzes the problem construction process of basic problems that is Concept Selection, Unknown Setting, and Proposition Construction. Ignoring the last step, some difficulty features can be observed during the problem is constructed. Six of them can be translated to four difficulty dimensions, according to the problem solving strategies – Identification, Elaboration, Planning, and Execution. The simplest neural network and its weights among neurons are applying to the transformation process which transforms the difficulty dimensions from the difficulty features.

An example system, Item Generating System, is used to generate item for learners to practice dynamically. The system is also constructed to demonstrate the difficulty calculation. Learners can observe the lack of abilities in solving problems from the system when they get wrong during the solving process. The possible future works include how to calculate the difficulty for the complex problems and how to measure the Mathematical Complexity which is also a big issue to affect the difficulty calculation for problems. Another research subject is the way to diagnose learners' problem-solving abilities which is the application of using difficulty features. How to integrate IRT into this difficulty setting method also becomes an interesting issue in this study.

References

Cheng, K. F., Kuo, R., Yang, K. Y., & Heh, J. H. (2001). Intelligent Agent Construction for Problem Solving with Knowledge Schemas. *Paper presented at the 17th Workshop on Science Education*, Kaohsiung, Taiwan.

Deway, J. (1910). *How We Think. Boston*, MA: Heath.

Deek, F. P., Turoff, M., & McHugh, J. A. (1999). A common model for problem solving and program development. *IEEE Transactions on Education*, *42* (4), 331 –336.

Haykin, S. (1999). *Neural networks: A comprehensive Foundation* (2nd ed.), Upper Saddle River, NJ: Prentice Hall, Inc.

Hsu, C. K., Kuo, R., Chang M., & Heh, J. S. (2002). Implementing a Problem Solving System for Physics based on Knowledge Map and Four Steps Problem Solving Strategies. *Paper presented at the IEEE International Conference on Advanced Learning Technologies*, September 9-12, 2002, Kazan, Russia.

Kahney, H. (1993). *Problem Solving: Current Issue* (2nd ed.), Milton Keynes, UK: Open University Press.

Kuo, R., Chang, M., Dong, D. X., Yang, K. Y., & Heh, J. S. (2002). Applying Knowledge Map to Intelligent Agents in Problem Solving Systems. *Paper presented at the World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-Media 2002)*, June 24-29, 2002, Denver, Colorado, USA.

Kuo, R., Lien, W. P., Chang, M., & Heh, J. S. (2003). Problem Model Analysis in Inter-Correlated Knowledge by Implementing Item Generating System in Knowledge Map. *Paper presented at the World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA 2003)*, June 23-28, 2003, Honolulu, Hawaii, USA.

Marshall, S. P. (1995). *Schemas in Problem Solving*, Cambridge, UK: Cambridge University Press.

Polya, G. (1965). *Mathematical Discovery*, New York: Wiley.

Tung, T. H. (2002). *Apply Knowledge Map to Develop Physics Problem-Solving System*. Master Thesis, Department of Information and Computer Engineering, Chung Yuan Christian University, Chung-Li, Taiwan.