行政院國家科學委員會專題研究計畫　成果報告

以影響圖為基礎之遠距教學課程設計

計畫類別：個別型計畫
計畫編號：NSC91-2520-S-032-011-
執行期間：91 年 08 月 01 日至 92 年 07 月 31 日
執行單位：淡江大學教育政策與領導研究所

計畫主持人：張家宜
共同主持人：葛煥昭，施國琛
計畫參與人員：林修名、劉毅人、吳佩瑛、蔡惠如

報告類型：精簡報告

處理方式：本計畫可公開查詢

中　華　民　國 93 年 2 月 12 日

計畫名稱:以影響圖為基礎之遠距教學課程設計

計畫類別：■ 個別型計畫　　□ 整合型計畫
計畫編號：NSC　91－2520－S－032－011－
執行期間：2002 年 08 月 01 日至 2003 年 7 月 31 日
計畫主持人：張家宜 教授
共同主持人：施國琛 教授、葛煥昭教授
計畫參與人員：林修名、劉毅人、吳佩瑛、蔡惠如

ABSTRACT

A Web-based courseware contains course contents and on-line tests. However, most Web document development tools are not incorporated with a quantitative evaluation mechanism, to allow a quantitative analysis of distance learning courseware. We propose an evaluation mechanism and a multimedia tool, based on our Courseware Diagram, to allow a quantitative justification of course contents. Courseware produced by our development system allows an instructor to choose different instruction *sequences* based on the outcomes of an exam. Alternatively, the courseware allows self-guided study in a Web-based distance learning program. This paper explains the courseware diagram, the evaluation algorithm, and the implementation of our courseware development system.

Key words: distance learning, courseware authoring, influence diagram, automatic assessment

1. INTRODUCTION

Distance learning authoring tools and platforms, such as Blackboard [http://www.blackboard.com/], LearningSpace [http://www.lotus.com/home.nsf/welcome/learnspace], and WebCT [http://www.webct.com/] were widely available. Although the systems provide assessment information, it is hard to find an automatic method which guides students from time to time in their learning process. In addition, student assessment by the above systems is nearly related to course contents in most cases, as it should be in a typical classroom. Therefore, it is important to investigate a strategic assessment method, on the content of courses and tests, to help both the instructors and the students. In our earlier paper [1], we proposed a model based on Influence Diagram. In this paper, we propose a newly developed quantitative analysis model, and an evaluation algorithm for the implementation of a working system.

2. EVALUATION OF A COURSEWARE DIAGRAM

In our courseware diagram, the topology of a courseware contains two types of parallelism – concepts that can be learned in parallel and parallel learning sequences for different levels of studies. The first results in a multiple accumulation or knowledge links from a course unit. The second is based on the test outcomes. Courseware parallelism also results in an important consequence – the differences of learning impacts. The evaluation of these learning impacts is the most important contribution of our proposed courseware diagram. After a courseware diagram and its contents (course units and test units) are designed by an expert, the courseware can be used by different instructors and students through different topologies. Topology parallelism of a course diagram can be used in different ways, with a quantitative guidance from our system:

- **Differentiated Instruction (via Norm-referenced Evaluation):** While an instructor is teaching a course, feedbacks from students are gathered from test outcomes. The instructor may choose a suitable learning path, which is recommended by our system. The final path used in the instruction can be compared against the path of an optimal (maximal) learning impact. The comparison can be used as a reference if the same course is taught again.
- **Self-guided study (via Criterion-referenced Evaluation)**: For life-long learning and employee training, a courseware using our system will help students to find out a suitable learning path. The topology of learning can be designed to include different speed of learning paths. Some of the longer paths may cover details for students who are lack of particular background. Other paths may represent a short cut or an overview. Students who visit these types of Web site can have a self-guided study.

We need to present a few definitions before our evaluation algorithm can be discussed.

Definition 1: A *Concept Weight* (CW) is a value associated with each course unit. A Concept Weight reflects the importance of knowledge presented in the course unit.

Definition 2: A *Learning Impact Value* (LIV) is a value factor accumulated according to Concept Weights.

A LIV reflects the effectiveness of learning in a particular learning sequence w.r.t. a check point (as a course unit). Thus, the accumulated value does not only reflect the impact of an individual course unit. LIVs are reduced by test outcomes and CWs. In general, a Concept Weight is given by the courseware designer, who should decide how much effort a student should spend in order to understand the course unit. But, Learning Impact Values are deduced from the diagram. Since courseware diagram allows alternative sequences of instruction and test outcomes for each student (or class average) are different, it is possible that the accumulated concept weights (i.e., LIVs) are different. However, the upper and lower bounds of LIVs are computable.

Definition 3: A Maximal LIV (LIV_{max}) represents the largest LIV w.r.t. a course unit. A Minimal LIV (LIV_{min}) indicates the smallest value. Differences of LIVs are deduced by test outcomes.

Definition 4: *Percents of Higher Group* (PHG) is the percentage of students who received a relatively higher score in a test. *Percents of Lower Group* (PLG), on the other hand, is the percentage of lower score students. A *Percentage Factor* (PF) is a pair of values from PHG and PLG associated with a test unit. These percentages are decided by the instructor. A PF divides the outcome of a test into three consequences (i.e., higher group, middle group, and lower group). If a PF is omitted, there exists only a single consequence from the test unit. As such, the entire class is a group. However, singular consequence of a test unit is not recommended.

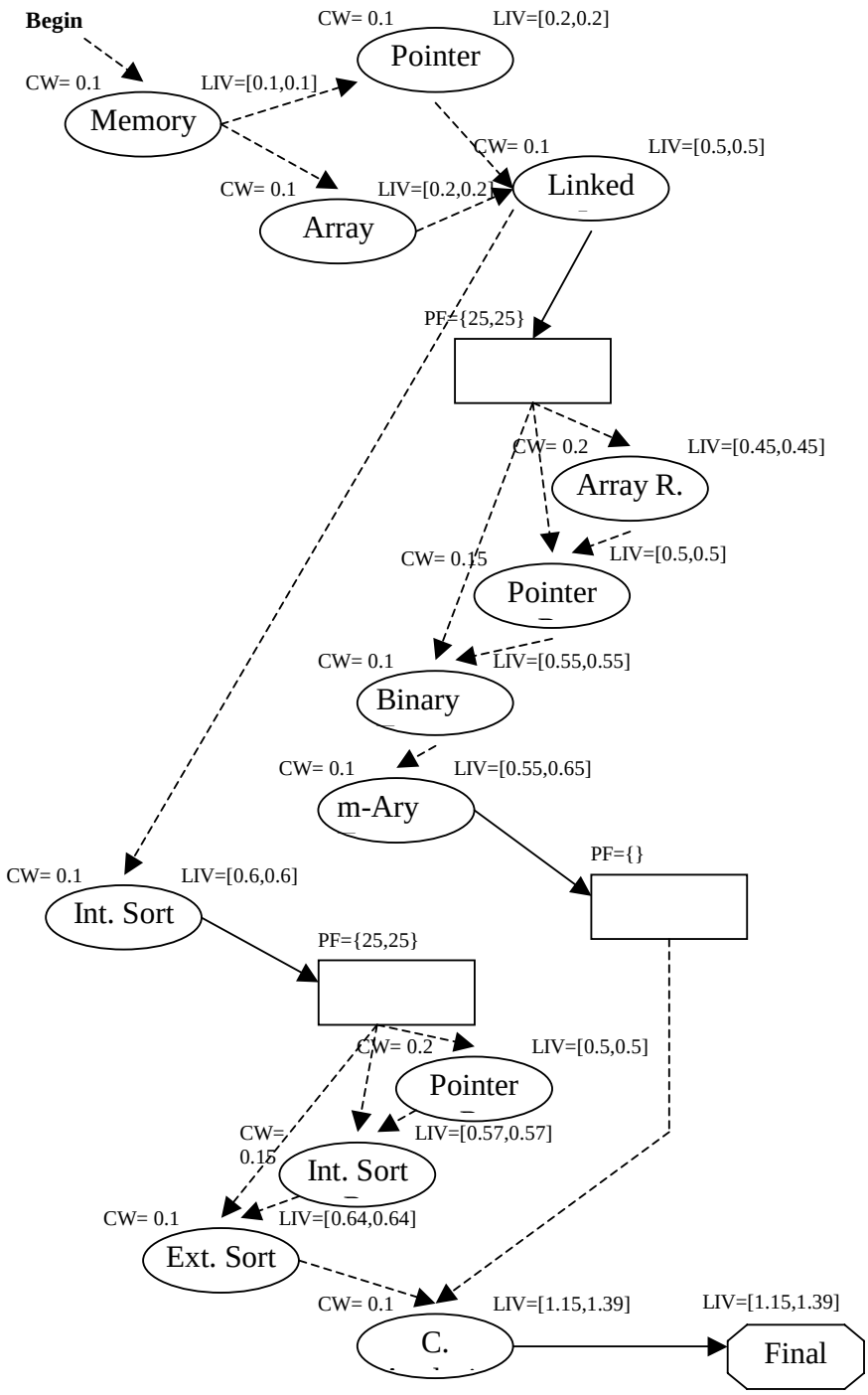

Figure 1: Courseware Diagram for Data Structure Class

As an example to show the usage of courseware diagram, we illustrate a realistic course. Figure 1 is the courseware diagram for a data structure class. The topology has two main themes: to explain tree structures and to explain sorting algorithms. The first theme is tested in Midterm 1 and Midterm 2. The sorting algorithms are tested in Midterm 3. The common prerequisites are course units Memory, Pointer, Array, and Linked Lists. In Midterm 1, we have PF={25,25}, which is the same as Midterm 3.

However, Midterm 2 has its PF omitted (since the instructor does not have enough time for the evaluation). We assume that, the top 25 percents of students have an average score percentage of 90%. The middle group has a percentage of 70%, and the lower group has 50%. These numbers reflects the learning impacts of students. The LIV_{min} and the LIV_{max} of course unit Array Review are 0.5*50%+0.2=0.45 Similarly, the LIV_{min} and the LIV_{max} of course unit Pointer Review are 0.5%70%+0.15=0.5 And, the LIV_{min} and the LIV_{max} of course unit Binary Tree are 0.5*90%+0.1=0.55 Moreover, the LIV_{min} of course unit m-Ary Tree is taken from the minimal of 0.45, 0.5, and 0.55, plus 0.1 (the CW). And the LIV_{max} is the maximal of 0.45, 0.5, and 0.55, plus 0.1. Midterm 2 does not differentiate students. Thus, the accumulated LIV in course unit Complexity Analysis (i.e., C. Analysis) is [0.55,0.65] (as a learning impact from the theme of tree structures). In the second path (i.e., the sorting algorithms), LIV is accumulated in course unit Internal Sort (i..e, Int. Sort). Midterm 3 differentiates students into three groups again. The differences of outcome result in the two review units (Pointer R. and Int. Sort R.), and the Ext. Sort unit. The LIVs of there three course units are computed in the similar manner. And, finally, the LIV of course unit Complexity Analysis (C. Analysis) is accumulated (i.e., LIV=[0.5+0.55+0.1, 0.64+0.65+0.1]=[1.15,1.39]). The LIV of the single and the last course unit (i.e., C. Analysis) will be used as the LIV of the final unit.

3. DIAGRAM REDUCTION AND LIV COMPUTATION

In the courseware diagram, course weights and percentage factors are given by the instructor. However, learning impact values are computed. Courseware diagrams are acyclic graphs, with directed edges (i.e., DAG). The first step is to reduce the diagram to a multi-level DAG structure. A *Basic DAG* is a compound object, with a singular entry node, and a singular exit node. For instance, in figure 1, Midterm 1 is an entry node and Binary Tree is an exit node. Similarly, Memory is an entry node and Linked L. is an exit node. The reduced courseware diagram shown in figure 2 has the above two Basic DAGs shown in bold lines. In addition, the whole reduced courseware diagram is a Basic DAG. Essentially a courseware diagram is a recursive structure. It is possible that, in a Basic DAG which represents an exam and its alternative outcomes, there is an embedded Basic DAG, either for another test, or for a compound course unit sequence (such as the Basic in figure 2). Thus, the computation algorithm for LIVs should be able to cope with this recursion. Note that, LIVs are accumulated from the top to the bottom of the diagram. Thus, even a test unit does not have a LIV, for the sack of completeness, Midterm 2 in figure 2 has a LIV, which is equal to [0.5, 0.56]. As a matter of fact, a Basic DAG also has a LIV, which represents the accumulated LIV value.

We present the recursive algorithm to evaluate a courseware diagram as the following. To compute the LIV of **Final**, which is the expected LIV of the diagram, the system calls ***CD-Eval***(Memory, [0.0, 0.0]). Note that, the initial LIV is shown as [0.0, 0.0]. ***CD-Eval*** has two formal parameters. Node is a pointer to any node. And, [LIV_{min}, LIV_{max}] is a pair of real numbers. We use an auxiliary function, ***BFS-Eval***, which take as input parameters a Node pointer, a LIV, and an ENode pointer. The ENode pointer points to an exit node of a Basic DAG. The ***BFS-Eval*** function is a revised breadth first search algorithm. However, the search terminates at the exit nodes. Also, ***BFS-Eval*** and ***CD-Eval*** are mutually recursive functions.

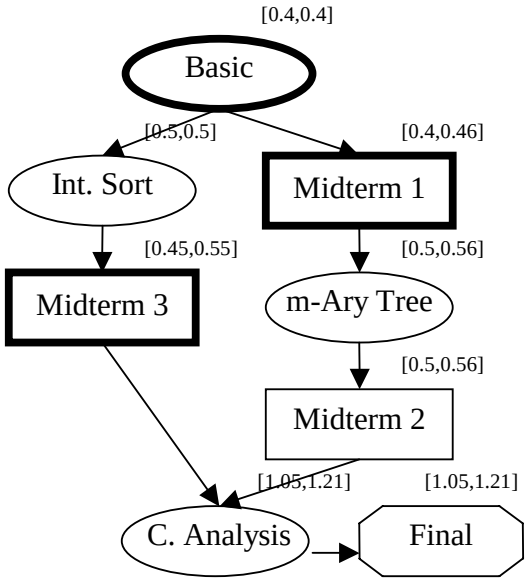

Figure 2: A Reduced Courseware Diagram as a DAG

We use "=" as a comparison operator, and "set" to represent an assignment statement. The algorithm needs to deal with four types of objects: atomic course units, atomic test units, compound course units, and compound test units. For an atomic course unit, its LIV is increased by its CW, and ***CD-Eval*** is called recursively. For an atomic test unit, the accumulated LIV is passed to its descendents. For a compound course unit, the revised breadth first search function is used to evaluate all parent nodes of the exit node (i.e., ENode). Then, the LIV of ENode is summarized from all of its parent nodes (within the Basic DAG). For a compound test unit, there are three alternative paths after the test unit. Assuming that they are Nx, Ny, and Nz, with the percentages of average scores equal to x, y, and z, respectively. Breadth first search is

```
CD-Eval(Node: Pointer, [LIV_min, LIV_max]: Pair of Real) is defined as
        {
        if Node is an atomic node then
                if Node is a course unit then {
                        set LIV of Node to [LIV_min + CW, LIV_max + CW]
                        CD-Eval(Node-->child, [LIV_min + CW, LIV_max + CW])
                        }
                else { /* a test unit should pass LIV to its descendant nodes */
                        set LIV of Node to [LIV_min, LIV_max]
                        CD-Eval(Node-->child, [LIV_min, LIV_max])
                        }
        else /* compound node */
                if Node is a course unit then {
                        set ENode be the exit node in the compound node
                        BFS-Eval(Node, [LIV_min + CW, LIV_max + CW], ENode)
                        set LIV of ENode to the sum of LIVs of all ENode's parent nodes
                        CD-Eval(ENode-->child, [LIV_min of ENode, LIV_max of ENode])
                        }
                else /* Node points to a test unit */
                        if PF of Node is omitted then { /* pass LIV to its descendant nodes */
                                set LIV of Node to [LIV_min, LIV_max]
                                CD-Eval(Node-->child, [LIV_min, LIV_max])
                                }
                        else { /* differentiated instruction */
                                set LIV of Node to [LIV_min, LIV_max]
                                set ENode be the exit node in the compound node
                                set x, y, z be the percentages of average scores of the 3 groups
                                set Nx, Ny, Nz be pointers to the first node in the 3 groups
                                BFS-Eval(Nx, [LIV_min * x + CW, LIV_max * x + CW], ENode)
                                BFS-Eval(Ny, [LIV_min * y + CW, LIV_max * y + CW], ENode)
                                BFS-Eval(Nz, [LIV_min * z + CW, LIV_max * z + CW], ENode)
                                set min to the minimal LIV_min values of parent nodes of EDode
                                set max to the maximal LIV_max valuse of parent nodes of ENode
                                set LIV of Node to [min, max]
                                CD-Eval(ENode-->child, [min, max])
                                }
        } /* end of CD-Eval */

BFS-Eval(Node: Pointer, [LIV_min, LIV_max]: Pair of Real, ENode: Pointer) is defined as
        {
        if Node = ENode then
            return
        else {
                set LIV of Node to [LIV_min + CW, LIV_max + CW]
                for each child node N of Node
                        CD-Eval(N, [LIV_min + CW, LIV_max + CW])
                }
        } /* end of BFS-Eval */
```

An Evaluation Algorithm of Courseware Diagram

applied to the three alternatives again. And, finally, the minimal and maximal values of ENode are computed from all of its parent nodes (within the Basic DAG). Based on the method discussed, we designed an authoring system, which allows the instructor to design distance learning courseware using the courseware diagram. Due to space limitation, we are not able to present our system here.

4. CONCLUSIONS

Distance learning seems to be a trend of future education. Or, at least distance learning tools will be able to help high level education to move toward another dimension of teaching. Distance learning tools focus on communication and multimedia presentation technologies. These new improvements enable Web-based course materials, video conferencing, video-on-demand lectures, and others. Several standard formats, including platform format and content format were proposed. Yet, assessment of distance learning is still weak. In this paper, we think of courseware design as a decision problem. We studied conceptual graphs and influence diagrams, and proposed a courseware diagram evaluation algorithm. The algorithm can be used in a software system, which allows an instructor to design a courseware as making a decision, which can then be computed to justify the maximal efficiency. Assessment of distance learning did not get much attention in the past, especially on the systematic mechanisms to evaluate the quality of a courseware. We hope that, the assessment criteria, or standard, can be realized by educators, engineers, and policy makers. Thus, future distance learning will provide better courseware and a more accurate control of education quality.

REFERENCES

[1] Timothy K. Shih and Robin Hung, "Multimedia Courseware Development Using Influence Diagram," in Proceedings of the 2002 IEEE International Conference on Multimedia and Expo (ICME'2002), Lausanne, Switzerland, August 26-29, 2002.

[2] http://www.webct.com/

[3] http://www.blackboard.com/

[4] http://www.lotus.com/home.nsf/welcome/learnspace