International Journal of Fuzzy Logic and Intelligent Systems, vol. 12, no. 1, March 2012, pp. 77-85
http://dx.doi.org/10.5391/IJFIS.2012.12.1.77 pISSN 1598-2645 eISSN 2093-744X

Pedagogically-Driven Courseware Content Generation for Intelligent Tutoring Systems

Hend Ben Hadji [1], Ho-Jin Choi [1*] and Mohamed Jemni [2]

[1] Dept. of Computer Science, KAIST, Daejeon 305-701, KOREA (South)
[2] Dept. of Computer Science, High School of Sciences and Techniques of Tunis, Bab Menara 1008, Tunisia

Abstract

This paper describes a novel approach to adaptive courseware generation. This approach adopts its structure from existing intelligent tutoring systems and introduces a new component called pedagogical scenario model to support pedagogical flexibility in the adaptation process of courseware generation system. The adaptation is carried out using Dynamic Constraint Satisfaction Problem framework, which is a variant of classical Constraint Satisfaction Problem, to deliver courseware tailored to individual learner. Such a framework provides a high level of expressiveness to deal with the particular characteristics of courseware generation problem. Further, it automatically designs a sound courseware satisfying the design constraints imposed by the domain, the pedagogical scenario and learner models.

Keywords: Courseware adaptation, pedagogy, dynamic constraint satisfaction.

1. Introduction

The generation of adaptive courseware has been the subject of many research efforts resulting in several courseware generations systems such as APeLS [1], INSPIRE [2], and DCG+GTE [3], providing varying degrees of adaptation. Ullrich and Melis [4] and Brusilovsky and Vassileva [5] identify adaptive courseware generation as a way to generate a structured sequence of learning resources that is adapted to the learner.

Typically, an adaptive courseware generation system borrows its structure from the architecture of intelligent tutoring and adaptive hypermedia systems where four well-differentiated building blocks can be found. The learner model holds information about the learner's characteristics and preferences. The domain model is composed of the educational domain describing how the educational content is structured and linked together, educational resources and their metadata. The adaptation model contains rules which define how the domain model relates to the learner model in order to ensure adaptation. Finally, the adaptation engine decides what adaptive technique to apply and manages the entire process of adaptation according to the learner needs and preferences.

Several systems have been quoted in the literature for courseware generation. Most of these systems support the adaptation into one or two forms of adaptation [6]: *adaptive presentation* (i.e., the system adapts the content to the learner) and *adaptive navigation* (i.e., the system alters a number of visible links to support adaptive hyperspace navigation), but often they do not support adaptive pedagogy form. Adaptive pedagogy refers to the selection of the appropriate pedagogy enabling the system to deliver adaptive courseware that, while dealing with the same subject matter and learning goals can be designed and structured in a way that best fits the learner's preferences [1]. For instance, a learner which tends to be active may prefer a project-based learning method to be followed by the delivered courseware; while reflective learner may follow the same courseware using a simple presentation method. Therefore, it would be of interest to consider also the pedagogical dimension in the courseware adaptation process.

In this paper we propose a novel conceptual model of courseware generation system which adopts its structure from existing systems and introduce a new component called pedagogical scenario model to support pedagogical flexibility in the adaptive courseware generation system. We formalize the pedagogical knowledge using Dynamic Constraint Satisfaction Problem (DCSP) framework, which is a variant of classical Constraint Satisfaction Problem (CSP). The advantage of such a framework is twofold. First, it automatically designs a pedagogically sound courseware satisfying the design constraints imposed by the domain, learner and pedagogical models, thus avoiding tedious manual controls of these constraints. Second, it exhibits a high level of expressiveness to deal with the particular characteristics of courseware generation problem.

This paper is organized as follows. Section 2 reviews the existing courseware generation proposals and outlines their limitations. Section 3 proposes a new conceptual model of courseware generation systems and describes the role played by each of its components as well as the adaptation process. Section 4 shows how courseware generation problem can be mapped to DCSP framework and discusses the implementation of the proposed courseware generation. Section 5 concludes the paper.

Manuscript received Feb. 5, 2012; revised Mar. 7, 2012; accepted Mar. 15, 2012
* Corresponding author: Ho-Jin Choi(hojinc@kaist.ac.kr)
This work was supported by the National Research Foundation (NRF) grant (No. 2011-0018264) of Ministry of Education, Science and Technology (MEST) of Korea.

2. Related Work

Several systems have been quoted in the literature for courseware generation. The DCG+GTE [3] is a courseware generation system based on the modelling of the educational domain and learner's characteristics; however, it does not take into account the pedagogical perspective in performing the adaptation, only a limited selection of exercises and obviously do not allow the reuse of resources into different pedagogical environments. The adaptation technique used in these systems is based on the AI planner.

APeLS [1] is an adaptive educational system that uses the multi-model, metadata driven approach to maintain a set of models describing the necessary learner, content and pedagogical information which the system can then reconcile, at runtime, in order to generate an adaptive course for an individual learner. The key advantage of APeLS is the separation of pedagogy from other models. This allows APeLS to adapt courseware to different pedagogical approaches; however, the model is not flexible enough to provide mediating tools supporting the implementation of different pedagogical methods.

PERSO [7] uses a case-based reasoning (CBR) approach to determine which course to propose to the learners based on their characteristics, namely the levels of knowledge and the media preferences. They put less focus on pedagogical knowledge and provide only pre-defined templates of courseware upon which the adaptation engine can reason what courseware to generate to the learners. Another disadvantage is the difficulty of reuse of courseware solutions for future needs.

Karampiperis and Sampson [8] propose a sequencing method that selects the best path that matches the learning goal based on the use of a suitability function that estimates the suitability of educational resources for a targeted learner. A major drawback is that the suitability function does not take into account that the same learner can have different learning needs regarding the same concepts or even the same courseware can presented in different pedagogical approaches.

Mohan et al. [9] and Sicilia et al. [10] use AI planning technique to generate the "IMS Learning Design" instances. Their descriptions are on a very abstract level and sketches some basic operators and methods. Their work provides evidence that AI planning techniques can be used for the generation of learning designs; however they do not provide any detailed formalization.

PAIGOS [11] describes a course generator used in the Web-based learning environment, called ActiveMath. A significant amount of pedagogical knowledge is present enabling the generation of courses for different pedagogical scenarios. Their work show how courseware generation problem can be represented and managed by automated planning technique. The adaptation engine uses a hierarchical task network(HTN)-planner. Their work is similar to our work in the sense that supports different pedagogical scenarios but they are restricted to constructivist theory.

To sum up, most of courseware generation systems are interested in the adaptation of courseware from content and/or presentation perspectives; however, the adaptation of courseware from pedagogical perspective is often ignored. This is mainly due to the fact that pedagogic information, if they exist, is encoded as part of the domain model in courseware generation systems and is limited to the semantics encoded in the domain model, so-called pedagogical relationships (e.g., 'prerequisite', 'part-of', etc.) and/or the pedagogical annotation of learning resources. This issue has led to a lack of pedagogical flexibility in the existing courseware generation systems.

3. Conceptual Model of Courseware Generation

While there are several conceptual models of courseware generation, the generic conceptual model emerges from their common building blocks: domain model, learner model, adaptation model and adaptation engine. These building blocks are represented by Vassileva and Bontchev [12] as a triangular structure where the adaptation engine is at the core of the model (see Figure 1). The main benefits of this structure are it provides a strong independence of any of the building blocks; however, it does not consider the pedagogical perspective in the adaptation of courseware. Furthermore, the adaptation model is not represented at the core of the conceptual model although it contains all necessary rules for adaptation engine.

To address these issues we presented in this paper a new conceptual model of courseware generation which adopts the architecture proposed by Vassileva and Bontchev [12], where four components are used: the domain model, learner model, adaptation model and adaptation engine, introduces a new component: the pedagogical scenario model to stress the importance of pedagogical perspective in courseware generation and regroups the adaptation model and the adaptation into one module called adaptation module and represent it at the core of the model (see Figure 1).

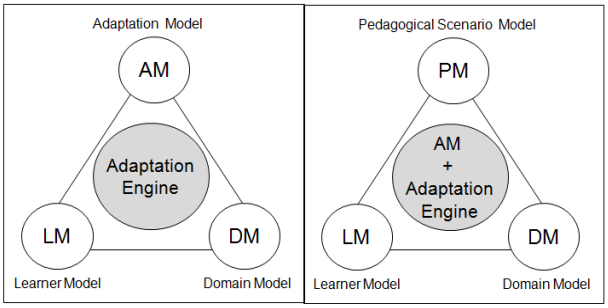

Figure 1. Vassileva and Bontchev model [12] vs. our proposed model

A clean separation between the pedagogical and domain models provides the system with the following advantages:

- *Modularity*: A strong independence of the building models of the courseware generator systems allows easy maintenance of the system. For example, changes in a particular model do not impact other models.
- *Pedagogical flexibility*: The system can easily reuse the

courseware content into different pedagogic environments without supplementary efforts of content adaptation.

- *Reuse*: The potential for the reuse of learning resources into different pedagogic environments is increased, and so is the reuse of different pedagogical strategies.

3.1. Pedagogical Scenario Model

Pedagogy is a fundamental information source used in improving the quality of learning because it influences the organization of courseware content and provides guidance and learner support. According to Franzoni and Assar [1] pedagogy refers to an organized and systematized sequence of activities and resources that teachers use while teaching. The main objective is to facilitate the students' learning. An appropriate selection of the pedagogy enables the system to deliver an adaptive courseware that, while dealing with the same subject matter and learning goals, can be designed and structured in a way that best fits the learner's preferences [13]. For instance, a learner which tends to be active may prefer a project-based learning method to be followed by the courseware; while reflective learner may follow the same courseware using a simple presentation method.

Therefore, it is essential that pedagogical information should be incorporated into a comprehensive pedagogical model. In this context, IMS consortium has developed the IMS Learning Design (LD) specification [14] offering an abstract model for learning designs. Even though this specification answers the need to formalize the representation of learning designs and to provide a set of machine interpretable metadata to assure interoperability, it has the limitation that it follows a traditional objectivist (i.e. behaviorist) theory of learning that asserts that knowledge must be objectified in order to be transmitted to learners. It is clear, however, that recent teaching and learning approaches have moved away from simply transmitting knowledge from teacher/book to the learner to more diverse approaches that engage the learner in the generation and evaluation of knowledge. Furthermore, in IMS LD learning resources are bound to learning designs at design time regardless of learner profiles, resulting in a limited flexibility in the adaptation of learning designs.

To cope with these issues, we propose an ontological model for pedagogical scenarios that accommodate a wide variety of pedagogical methods. The main elements composing a pedagogical scenario are:

- *Pedagogical Identifier Code (PIC),* given to reference each pedagogical scenario stored in a repository. The use of PIC facilitates the search and retrieval of pedagogical scenarios from a repository.
- *Pedagogical objective,* which identifies the objective to achieve through the pedagogical scenario being modeled. Pedagogical objectives are defined here based on Bloom's taxonomy [15] which defines six levels of learning: knowledge, understanding, application, analysis, synthesis, and evaluation.
- *Pedagogical method,* which represents the teaching approach followed by a particular pedagogical scenario. There have been many pedagogical methods in education, but the following ten types of pedagogical methods are commonly considered in pedagogical design [16]: presentation, problem solving, discussion, brainstorming, games, simulation, role playing, case study, project design method, question and answer method.
- *Activity-sequencing,* which refers to an organized sequence of didactic activities (used for applying a particular pedagogical scenario). A didactic activity can be introduction, definition, problem, example, solution, etc. It is worth noting that the sequencing of didactic activities is the responsibility of instructional designers and pedagogical experts.
- *Candidate concepts,* which provide the potential concepts that might be associated with the particular pedagogical scenario. This helps the course author to select the appropriate pedagogical scenario in one hand and supports the reuse of pedagogical scenarios for different learning goals in the other hand.
- *Mediating tool,* which identifies the teaching tool or technique to implement a pedagogical method. For example, a set of PowerPoint slides can be a mediating tool for presentation. E-mails can be a mediating tool for discussion or project design method.

3.2. Learner Model

Learner is essential for an adaptive system to provide the adaptation effect, i.e., to behave differently for different learner or users [17]. Therefore, the information represented in the learner model should be able to represent in a convenient way the characteristics of the learner which are source of adaptation.

We envisage to represent our learner model into three parts, where the first part, called the *general part*, holds general information about the learner such as his/her unique identifier, forename, surname, date of birth, etc. The second part, called the *knowledge part*, represents the primary elements being modeled in the learner model, i.e., the cognitive state. This can be expressed by the following information:

- *Prior knowledge:* A set of domain-concepts already possessed by the learner
- *Target knowledge:* A set of domain-concepts that the learner lacks and must acquire to complete the courseware
- *Level of knowledge:* Novice, intermediate, or expert

The third part, called the *preferences part*, contains information pertaining to the learner`s preferences. This includes:

- *Felder and Silverman`s learning style* [18]*:* Felder and Silverman propose to categorize learners in four learning categories. Each category is characterized by two opposite

attributes (e.g., Visual vs. Verbal) which represent the range of possibilities for that category (see Table 1).

- *Media type:* Text, image, video, audio, illustration, or animation

Table 1. Felder and Silverman's learning styles dimensions [18]

Learning style	Attributes	Description
Perception	Sensitive	Deal with facts, raw data and experiments; they are patient with details, but do not like complications.
	Intuitive	Deal with principles and theories, are easily bored when presented with details and tend to accept complications.
Input	Visual	Easy for them to remember what they see: images, diagrams, time Tables, films, etc.
	Verbal	Remember what they have heard, read or said.
Processing	Active	Learn by working in groups and handling stuff
	Reflective	Learn better when they can think and reflect about the information presented to them. Work better alone or with one more person at most.
Understanding	Sequential	Follow a lineal reasoning process when solving problems and can work with a specific material once they've comprehended it partially or superficially.
	Global	Take big intuitive leaps with the information, may have a difficulty when explaining how they got to a certain result, need an integral vision.

3.3. Domain Model

The domain model is composed of the domain knowledge describing how the knowledge space is structured and linked together, resources and their metadata.

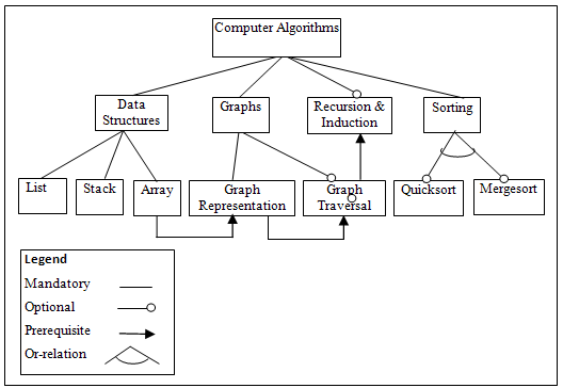

Figure 2. Partial view of computer algorithm domain model

We define the domain knowledge in a way that represents all possible courseware for a given subject in a single model using concepts. A concept can be a coarse or fine grained element of knowledge about a given subject. The relations between concepts, adapted from [19], take five different forms:

- *Mandatory:* A sub-concept is said to be mandatory when it is required to appear when the parent concept appears.
- *Optional:* A sub-concept is said to be optional when it can or not appear when the parent concept appears.
- *Composed-of:* A set of sub-concepts are said to have a composed–of relation with their parent concept when the later can be decomposed into fine grained concepts.
- *Alternative:* A set of sub-concepts are said to be alternative when only one sub-concept can be selected when the parent concept appears.
- *OR:* A set of sub-concepts are said to have an OR relation with their parent when one or more sub-concepts can be selected when the parent concept appears.

In addition to the structural relations, dependencies are also allowed. They refer to additional cross–tree constraints to restrict concept combinations and take here two different forms:

- *Prerequisite:* A concept, C1, is a prerequisite of C2 means that if C1 appears then C2 should appear but not backwards.
- *Exclude:* A concept, C1, excludes C2 means that if C1 appears then C2 should not appear and backwards.

Each concept in the domain knowledge is associated with a set of resources that focus on this concept. This association is established through metadata. Our resources model has the following descriptive metadata scheme, extracted from Learning Object Metadata (LOM) specification [20]:

- *Resource_id,* which is the unique identifier of the resource.
- *Category,* which defines the category to which belongs the resource.
- *Pedagogic role,* which describes the didactic role developed by a resource in reference to didactic activities embodied in the pedagogical model.
- *Media type,* which indicates the form in which a resource is presented. The most used types of media are: text/image, video, audio, and simulation.
- *Address,* which refers to the physical or logical address where an educational resource is stored.

It is worth noting that resources should be sufficiently small grained, in the sense that it describes a single concept and is presented by one type of data. Indeed with small grained resources, the course designer has greater flexibility in the reuse of existing resources to assemble new courseware. For example, if the granularity level of resources is of a description, theorem, or demonstration level then the course designer can add/remove content at this finer level to produce several versions of the same courseware.

3.4. Adaptation Model

In order to introduce the pedagogy perspective in the adaptation process of courseware generation, we take into account three hierarchical levels (see Figure 3): domain concepts selection, pedagogical scenario selection, and resources selection.

- *Domain concept selection level.* The first level constitutes the classical level of any adaptation process in courseware generation systems. It consists of selecting concepts related to a learning goal as well as associated concepts (sub-concepts and prerequisites) based on the domain model relationships. This formulates a conceptual structure that represents all the concepts of a learning goal and their relationships.
- *Pedagogical scenario selection level.* At this level pedagogical scenario selection takes place. The course author might impose a particular pedagogical scenario to use for designing and organizing the courseware or let the system decides which pedagogical scenario to select with a respect to the attributes of learner model. The connection between the pedagogical scenario model and the learner model is established based on the adaptive teaching taxonomy proposed by [1]. This taxonomy consists on matching the different learning styles with teaching strategies. It also suggests the suitable mediating tool for implementing teaching strategies. Table 2 depicts this connection.
- *Resources selection level.* The main functionality of this level is to associate to both selected concepts and pedagogical scenario the corresponding learning resources based on the connection of the resources model with both learner and pedagogical scenario models.

The advantage of our approach is that it allows a complete independence between the domain model, learner model and pedagogical scenario model. Such independence enables various forms of adaptation (content adaptation, presentation adaptation, and pedagogy adaptation). Furthermore, we could identify the appropriate pedagogical scenario using some of the identified learning preferences to improve the efficiency of the learning process through matching leaner and pedagogical scenario models.

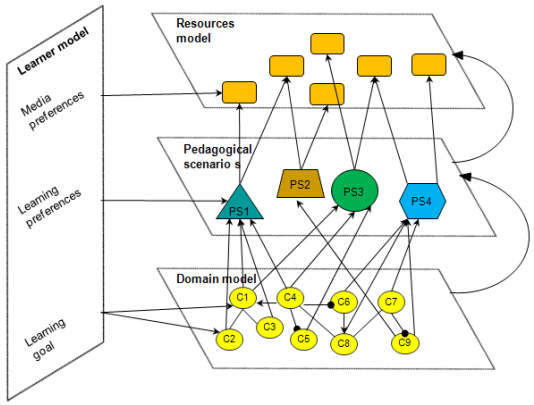

Figure 3. Adaptation process in courseware generation system

Table 2. Relating learner model to pedagogical scenario model

Learner model	Pedagogical scenario model
Learner level: - Novice - Average - Expert	*Pedagogical objective:* - Knowledge (Novice) - Understanding (Novice) - Application (Average) - Analysis (Average) - Synthesis (Average) - Evaluation (Expert)
Processing category of learning style: - Active - Reflective *Perception category of learning style:* - Sensitive - Intuitive *Understanding category of learning style:* - Sequential - Global *Input category of learning style:* - Visual - Verbal	*Learning method:* -Presentation (Sensitive, Visual, Reflective, Sequential) - Brainstorming (Verbal, Active, Global) - Problem solving (Sensitive, Active) - Discussion (Intuitive, Verbal, Active) - Games (Intuitive, Visual, Active) - Simulation (Intuitive, Visual, Active) - Role playing (Intuitive, Active, Global) - Case study (Intuitive, Reflective, Global) - Project design method (Intuitive, Active, Global) - Question and answer method (Sensitive, Verbal, Reflective, Sequential) *Mediating tool:* - Audio recording, audio conference (Verbal, Sequential) - Forums, Wikis (Sensitive, Active, Visual, Global) - Blog (Sensitive, Active, Global) - E-mail, chat (Active, Global) - Animations, graphics, pictures (Sensitive, Visual) - Digital newspaper (Reflective, Sequential) - Digital magazine (Reflective) - Web pages, e-books, Slides shows (Visual, Reflective, Sequential) - Internet research (Intuitive, Active, Reflective, Global) - Course Legacy System (Intuitive) - Tutorial system, WebQuest (Intuitive, Reflective) - Podcast (Verbal) - Recorded Live events, videoconference, videos (Visual, Verbal)

4. Modeling Courseware Generation Using DCSP

Dynamic Constraint Satisfaction Problem (DCSP) consists of a classical Constraint Satisfaction Problem (CSP) together with constraints describing how variables are added (or removed) from a problem based on other particular variable assignments [21], called activity constraints.

Formally, a DCSP is defined as $(V, V_{initial}, D, C_C, C_A)$, where $V = \{v_1, \ldots, v_i\}$ is the set of variables, $V_{initial} \subseteq V$ is the set of initial variables that must appear in all solutions, and $D = \{D_1, \ldots, D_i\}$ is the set of domains of the variables providing a set $D_i = \{d_{i1}, \ldots, d_{ij}\}$ of values of each variable. The compatibility constraints C_C correspond to the constraints found in classical CSPs, specifying allowed combinations of variables assignments and the activity constraints C_A specify the conditions under which a variable and its associated domain become relevant in

constructing a solution.

Following the DCSP formalism proposed in [22, 23], we propose to use the following form of activity constraint:

$$v_1, \ldots, v_i \xrightarrow{ACT} m \{ v_j \mid \ldots \mid v_k \} n$$

where $0 \leq m \leq n$. This activity constraint states that if variables $v_1, \ldots, v_i$ are active in and satisfied by an assignment, the subset of variables $v_j, \ldots, v_k$ are active in the assignment.

A solution to DCSP is an assignment A that has assignment to the set of initials variables and satisfies $C_C \cup C_A$.

4.1. Modeling Courseware Generation Problem as a DCSP

We define courseware generation problem as a DCSP such that each concept included in the domain model is represented by a variable, the domain associated with these variables is a set of pedagogical scenarios available in the repository. The root concept defined in the domain model and target concepts specified by the learner are represented as initial variables. The relations defined by the domain model are translated into activity constraints; whereas the conditions under which a concept is assigned a selected pedagogical scenario are specified in compatibility constraints. A solution to DCSP is a courseware that satisfies all compatibility and activity constraints and has assignments for each of the initial variables.

To illustrate how courseware generation problem can be mapped to a DCSP model, let us consider a simplified view of a computer algorithms model (see Figure. 2), which consists of 12 concepts and 13 relations, among which 7 are mandatory, 2 optional, 3 prerequisite and 1 OR relations. Consider now a learner who wants to take a computer algorithms courseware to learn about graph representation following a problem solving approach assuming he/she has no prior background on that domain.

Tables 3, 4 and 5 describe the mapping of courseware generation problem to DCSP model. Each concept is represented by a variable, denoted by V_{Ci}, and the domain of each variable includes one or more scenarios (see Table 3). A scenario is represented by S_{Meth}, where S stands for the Scenario and *Meth* a short name of the pedagogical method associated with it. For example S_{Pres}, $S_{Q\&A}$, S_{Sim}, S_{PS} represent the scenarios respectively associated with the presentation, question and answer, simulation, and problem solving methods. The target concepts which have to be present in every courseware solution are included in the set of initial variables as well as the root concept (i.e., computer algorithms). The activity constraints C_A specify the conditions under which a variable and its associated domain become relevant in constructing a courseware solution. For example, the constraint

$$V_{Ca} \xrightarrow{ACT} V_{Ds}$$

states if computer algorithms concept is active in a courseware solution, data structures concept is then active. This allows the selection of relevant concepts to a specific courseware generation problem and the removal of irrelevant ones. The compatibility constraints C_C correspond to the constraints found in classical CSPs, i.e., normal constraints specifying allowed combinations of variables assignments. The activity constraints C_A and compatibility constraints C_C are respectively given in Table 4 and Table 5.

Table 3. DCSP variables of the computer algorithms domain

Variable	Description	Domain
$V_{Ca} \in V_{initial}$	Computer algorithms	$D(V_{Ca}) = \{S_{Pres}, S_{Q\&A}\}$
V_{Ds}	Data structures	$D(V_{Ds}) = \{S_{pres}\}$
V_{Ls}	List	$D(V_{Ls}) = \{S_{pres}\}$
V_{Ar}	Array	$D(V_{Ar}) = \{S_{pres}\}$
V_{St}	Stack	$D(V_{St}) = \{S_{pres}\}$
V_{Ri}	Recursion and induction	$D(V_{Ri}) = \{S_{pres}\}$
V_{Sa}	Sorting algorithms	$D(V_{Sa}) = \{S_{pres}\}$
V_{Qs}	Quick sort	$D(V_{Qs}) = \{S_{pres}, S_{sim}\}$
V_{Ms}	Merge sort	$D(V_{Ms}) = \{S_{pres}, S_{sim}\}$
V_{Ga}	Graph algorithms	$D(V_{Ga}) = \{S_{Pres}, S_{Q\&A}\}$
$V_{Gr} \in V_{initial}$	Graph representation	$D(V_{Gr}) = \{S_{pres}, S_{PS}\}$
V_{Gt}	Graph traversal	$D(V_{Gt}) = \{S_{pres}, S_{sim}, S_{PS}\}$

4.2. Implementation

There exist two approaches to solve DCSP. The first approach was proposed by Mittal and Falkenhainer [24]. It consists of using dynamic backtracking algorithm to solve DCSP. The second approach, which we use to solve courseware generation problem, was proposed by Kambhampati [25]. It consists of compiling DCSP into a standard CSP and using the standard CSP algorithms. The main idea is to introduce a new "null" value (denoted by "$\perp$") into the domains of each of the DCSP variables. We then model an active DCSP variable as a CSP variable which takes the value "$\neq \perp$". Table 6 demonstrates such a compilation.

Table 4. Activity constraints of the computer algorithms domain

Activity constraints	Description	Domain model relations
$V_{Ca} \xrightarrow{ACT} V_{Ds}$	If computer algorithms concept is active in a solution, data structures concept is then active	Mandatory relations
$V_{Ds} \xrightarrow{ACT} V_{Ls}$	If data structures concept is active in a solution, list concept is then active	
$V_{Ds} \xrightarrow{ACT} V_{Ar}$	If data structures concept is active in a solution, array concept is then active	
$V_{Ds} \xrightarrow{ACT} V_{St}$	If data structures concept is active in a solution, stack concept is then active	
$V_{Ca} \xrightarrow{ACT} V_{Ga}$	If computer algorithms concept is active in a solution, graph algorithms is then active	
$V_{Ga} \xrightarrow{ACT} V_{Gr}$	If graph algorithms concept is active in a solution, graph representation is then active	

$V_{Ca} \xrightarrow{ACT} V_{Sa}$	If computer algorithms concept is active in a solution, sorting algorithms is then active	
$V_{Ca} \xrightarrow{ACT} 1$ $\{V_{Qs} \mid V_{Ms}\}$ 2	If sorting algorithms concept is active in a solution, quick sort, merge sort, or both might be active	OR-relation
$V_{Ca} \xrightarrow{ACT} 0$ $\{V_{Ri}\}$ 1	If computer algorithms concept is active in a solution, recursion and induction might be active	Optional relations
$V_{Ga} \xrightarrow{ACT} 0$ $\{V_{Gt}\}$ 1	If graph algorithms concept is active in a solution, graph traversal might be active	
$V_{Gr} \xrightarrow{ACT} V_{Ar}$	If graph representation concept is active in a solution, array concept is then active	Prerequisite relations
$V_{Gt} \xrightarrow{ACT} V_{Gr}$	If graph traversal concept is active in a solution, graph representation is then active	
$V_{Ri} \xrightarrow{ACT} V_{Gr}$	If graph recursion and induction concept is active in a solution, graph traversal is then active	

Table 5. Compatibility constraints of the computer algorithms domain

Compatibility constraints	Description	Domain
$V_{Gt} = S_{PS}$	Graph traversal concept is assigned a scenario with a problem solving method	Learner Preferences
$V_{Ca} = S_{pres}$	Computer algorithms concept is assigned a scenario with a presentation method	Other constraints (imposed by the pedagogue or the course author)
$V_{Gr} = S_{PS} \rightarrow V_{Ga} \neq S_{Q\&A}$	If the graph presentation concept is assigned a scenario with a problem-solving method, the graph algorithms concept must be assigned a scenario different from problem-solving method.	

Table 6. Compiling DCSP to CSP

DCSP model	CSP model
Variables	*Variables*
$V_{Ca}, V_{Ds}, V_{Ls}, \ldots, V_{Gr}, V_{Gt}$	$V_{Ca}, V_{Ds}, V_{Ls}, \ldots, V_{Gr}, V_{Gt}$
Initial variables	*Initial variables*
$V_{Ca}, V_{Gt} \in V_{initial}$	$V_{Ca} \neq \perp \wedge V_{Gt} \neq \perp$
Domains	*Domains*
$V_{Ca} = \{S_{Pres}, S_{Q\&A}\}$, $V_{Ds} = \{S_{Pres}\}$, $V_{Ls} = \{S_{Pres}\}$, …, $V_{Gr} = \{S_{Pres}, S_{PS}\}$, $V_{Gr} = \{S_{Pres}, S_{Sim}, S_{PS}\}$.	$V_{Ca} = \{S_{Pres}, S_{Q\&A}, \perp\}$, $V_{Ds} = \{S_{Pres}, \perp\}$, $V_{Ls} = \{S_{Pres}, \perp\}$, …, $V_{Gr} = \{S_{Pres}, S_{PS}, \perp\}$, $V_{Gr} = \{S_{Pres}, S_{Sim}, P_{PS}, \perp\}$.
Activity constraints	*Constraints*
$V_{Ca} \xrightarrow{ACT} V_{Ds}$ $V_{Ds} \xrightarrow{ACT} V_{Ls}$ …	$V_{Ca} \neq \perp \xrightarrow{ACT} V_{Ds} \neq \perp$ $V_{Ds} \neq \perp \xrightarrow{ACT} V_{Ls} \neq \perp$ …
$V_{Gt} \xrightarrow{ACT} V_{Gr}$ $V_{Ri} \xrightarrow{ACT} V_{Gr}$	$V_{Gt} \neq \perp \xrightarrow{ACT} V_{Gr} \neq \perp$ $V_{Ri} \neq \perp \xrightarrow{ACT} V_{Gr} \neq \perp$
Compatibility constraints	*Constraints*
$V_{Gr} = S_{PS}$ $V_{Ca} = S_{pres}$ $V_{Gr} = S_{PS} \rightarrow V_{Ga} \neq S_{Q\&A}$	$V_{Gr} = S_{PS}$ $V_{Ca} = S_{pres}$ $V_{Gr} = S_{PS} \rightarrow V_{Ga} \neq S_{Q\&A}$

We use the open source solver CHOCO [26] to generate possible courseware solutions. Due to the length reasons, Figure 4 illustrates a part of the constraint programming code using CHOCO solver. The first two lines (lines 1-2) create an instance of *CPModel()* for constraint programming model. The second part (lines 4-8) define the domain concepts as CSP variables and the boundaries for their domains using the *makeIntVar()* method. The option *Options.V_NO_DECISION* added in the definition of variables specifies that the variables *Ca* and *Ds* can be excluded from the search strategy if they are not activated. This translates the activity constraints and active variables explained above. Once the variables are defined, the constraints are then added (lines 10-11) using the *m.addConstraint()* method. For example, the line 11 states if $V_{Gr} = S_{PS}$ then $V_{Ga} \neq S_{Q\&A}$. Finally, an instance of *CPSolver()* for constraint programming solver is created. Then, the solver reads (translates) the model and solves it (lines 13-14).

Given the above CSP model, CHOCO solver generates eight courseware solutions. The assignment $\mathcal{A}_1 = \{V_{Ca} = S_{Pres}, V_{Ds} = S_{Pres}, V_{Ls} = S_{Pres}, V_{Ar} = S_{Pres}, V_{St} = S_{Pres}, V_{Ga} = S_{Pres}, V_{Gr} = S_{PS}, V_{Sa} = S_{Pres}, V_{Qs} = S_{Sim}\}$ is a solution since it satisfies the compatibility and activity constraints and has assignment to the set of initials variables. However, $\mathcal{A}_2 = \{V_{Ca} = S_{Pres}, V_{Ds} = S_{Pres}, V_{Ls} = S_{Pres}, V_{Ar} = S_{Pres}, V_{St} = S_{Pres}, V_{Ga} = S_{Q\&A}, V_{Gr} = S_{PS}, V_{Sa} = S_{Pres}, V_{Qs} = S_{Sim}\}$ is not a solution since the assignment $V_{Ga} = S_{Q\&A}$ violates the compatibility constraints. It is worth noting that a satisfying courseware solution allows only the selection of relevant concepts satisfying both $C_C \cup C_A$, thus discarding the non potential concepts.

```
1.  CPModel m=new CPModel();
2.  CPSolver   s = new CPSolver();
3.  …
4.  Ca=Choco.makeIntVar("Ca",tab,
    Options.V_NO_DECISION);
5.  Ds=Choco.makeIntVar("Ds",tab1,
    Options.V_NO_DECISION);
6.  …
7.  Scenarios1=Choco.makeIntVar("scenarios1",A);
8.  scenarios2=Choco.makeIntVar("scenarios2",B);
9.   …
10. m.addConstraint(Choco.eq(Ca,pres));
11. m.addConstraint(Choco.implies(Choco.eq(Gr,PS),
    Choco.neq(Ga,Q&A)));
12. …
13. s.read(m);
14. s.solve();
```

Figure 4. Partial view of constraint programming using CHOCO

5. Conclusion

This paper has proposed a new conceptual model for courseware generation to support pedagogical flexibility in courseware generation, and formulated the courseware generation problem as a DCSP. This approach allows the translation of all forms of relations defined in the domain knowledge model to activity constraints, learning goals to initial variables, and other design constraints to compatibility constraints. We argue that DCSP is more expressive than CSP in terms of knowledge representation. In terms of computational overhead, however, combining different elements or generalized definitions requires further optimization. For the future work, we plan to extend our DCSP framework to support reusing previous courseware solutions for prospective learners with similar characteristics. Knowledge conversion techniques such as studied in [27] may help this process of reusing knowledge.

References

[1] A.L. Franzoni and S. Assar, "Student Learning Styles Adaptation Method Based on Teaching Strategies and Electronic Media," *International journal of Educational Technology & Society*, vol. 1, no. 4, pp. 15-29, 2009.

[2] K. A. Papanikolaou, M. Grigoriadou, H. Kornilakis and G. Magoulas, "Personalizing the Interaction in a Web-based Educational Hypermedia System: the case of INSPIRE". *International Journal of User Modeling and User-Adapted Interaction*, vol. 13, pp. 213-267, 2003.

[3] J. Vassileva, "DCG+GTE: Dynamic courseware generation with teaching expertise," *International Journal of Instructional Science*, vol. 26, pp. 317-332, 1998.

[4] C. Ullrich and E. Melis, "Pedagogically founded courseware generation based on HTN-planning," *International Journal of Expert Systems with Applications*, vol. 36, pp. 9319-9332, 2009.

[5] P. Brusilovsky and J. Vassileva, "Course sequencing techniques for large-scale web-based education," *International Journal of Continuing Engineering Education and Lifelong Learning*, vol. 13, no.1/2, pp. 75-94, 2003.

[6] P. Brusilovsky, "Methods and techniques of adaptive hypermedia," *User Modeling and User-Adapted Interaction Journal*, vol. 6, no. 2-3, pp. 87-129, 1996.

[7] H. Chorfi and M. Jemni, "PERSO: Towards an adaptive e-learning system," *Journal of Interactive Learning Research*, vol. 15, no. 4, pp 433-447, 2004.

[8] P. Karampiperis and D. Sampson, "Adaptive Learning Resources Sequencing in Educational Hypermedia Systems," *International Journal of Educational Technology & Society*, vol. 8, no. 4, pp. 128-147, 2005.

[9] P. Mohan, J. Greer and G. McGalla, "Instructional Planning with Learning Objects," *In Proc. of the Workshop on Knowledge Representation and Automated Reasoning for E-Learning Systems,* Acapulco, 2003.

[10] M.-A. Sicilia, S. Sanchez-Alonso and E. Garcia-Barriocanal, "On supporting the process of learning design through planners," *In Virtual Campus 2006 Post-proceedings*, 2006.

[11] E. Melis, E. Andres, J. Budenberder, A. Frishaud, G. Goguadse, P. Libbrecht, M. Pollet and C. Ullrich, "ActiveMath: A generic and adaptive web-based learning environment," *International journal of AI in Education*, vol. 12, no. 4, 2001.

[12] D. Vassileva and B. Bontechev, "Self adaptive hypermedia navigation based on learner model characters," *In Proc. of IADAT*, Barcelona, Spain, pp. 46-52, 2006.

[13] O. Conlan, D. Lewis, S. Higel, D. O'Sullivan and V. Wade, "Applying adaptive hypermedia techniques to semantic web service decomposition," *In Proc. of AH2003: Workshop on adaptive hypermedia and adaptive web-based systems*, pp. 53-62, 2003.

[14] IMS Learning Design Information Model, http://www.imsglobal.org/learningdesign/ldv1p0/imsld_infov1p0.html

[15] B.S. Bloom, *Taxonomy of Educational Objectives*, New York: David Mckay, 1956.

[16] R. Heinich, M. Molenda, J. D. Russell and S. E. Smaldino, *Instructional Media and Technologies for Learning*, Prentice-Hall, 1999.

[17] P. Brusilovsky and E. Millan, "User models for adaptive hypermedia and adaptive educational systems," In *The Adaptive Web, Lecture Notes in Computer Science*, vol. 4321, pp. 3-53, Springer, 2007.

[18] R. Felder and L. Silverman, "Learning and Teaching Styles in Engineering Education," *Engineering Education*, vol. 78, no. 7, pp. 674-681, 1988.

[19] K. Kang, S. Cohen, J. Hess, W. Novak and S. Peterson, "Feature–Oriented Domain Analysis (FODA) Feasibility Study," *Technical Report CMU/SEI-90-TR-21*, Software Engineering Institute, Carnegie Mellon University, 1990.

[20] LOM, Draft Standard for Learning Object Metadata, http://Itsc.ieee.org/wg12/index.html

[21] I. Miguel, *Dynamic Flexible Constraint Satisfaction and Its Application to AI Planning*, PhD thesis, Springer, ISBN1-85233-764-8, 2004.

[22] T. Soininen and E. Gelle, "Dynamic constraint satisfaction in configuration," *AAI Technical Report WS-99-05*, pp. 419-434, 1999.

[23] T. Soininen, E. Gelle and I. Niemla, "A fixpoint definition of dynamic constraint satisfaction," *In Proc. of Constraint Programming*, pp. 419-434, 1999.

[24] S. Mittal and B. Falkenhainer, "Dynamic constraint satisfaction problems", *In Proc. of 8th National Conference on Artificial Intelligence*, pp. 25-32, 1990.

[25] S. Kambhampati, , "Planning Graph as a (Dynamic) CSP: Exploiting EBL, DDB and other CSP Search Techniques in Graphplan," *Journal of Artificial intelligence Research*, vol. 12, pp. 1-34, 2000.

[26] CHOCO CSP Solver, http://www.emn.fr/x-info/choco-

solver/
[27] J.S. Kim, "A Knowledge Conversion Tool for Expert Systems," *International Journal of Fuzzy Logic and Intelligent Systems*, vol.11, no.1, pp.1-7, March 2011.

Hend Ben Hadji
PhD Candidate, Dept. of Computer Science, KAIST, Korea
Research Area: E-learning, intelligent tutoring systems, component-based software engineering.
E-mail: hend@kaist.ac.kr

Ho-Jin Choi
Associate Professor, Dept. of Computer Science, KAIST, Korea
Research Area: Artificial intelligence, software engineering, data mining, biomedical informatics.
E-mail: hojinc@kaist.ac.kr

Mohamed Jemni
Professor, Dept. of Computer Science, High School of Sciences and Techniques of Tunis, Tunisia
Research Area: E-learning, grid computing
E-mail: mohamed.jemni@fst.rnu.tn