A Category-based Framework of a Self-improving Instructional Planner

Roberto Legaspi

The Institute of Scientific and Industrial Research, Osaka University, Japan
roberto@ai.sanken.osaka-u.ac.jp, <http://www.ai.sanken.osaka-u.ac.jp>

Raymund Sison

College of Computer Studies, De La Salle University - Manila, Philippines
sisonr@dlsu.edu.ph, <http://www.dlsu.edu.ph/ccs>

Masayuki Numao

The Institute of Scientific and Industrial Research, Osaka University, Japan
numao@sanken.osaka-u.ac.jp, <http://www.ai.sanken.osaka-u.ac.jp>

keywords: intelligent tutoring system, self-improvement, learner categorization, machine learning

Summary

To have an instructional plan guide the learning process is significant to various teaching styles and an important task in an ITS. Though various approaches have been used to tackle this task, the compelling need is for an ITS to improve on its own the plans established in a dynamic way. We hypothesize that the use of knowledge derived from student categories can significantly support the improvement of plans on the part of the ITS. This means that category knowledge can become effectors of effective plans. We have conceived a Category-based Self-improving Planning Module (CSPM) for an ITS tutor agent that utilizes the knowledge learned from learner categories to support self-improvement. The learning framework of CSPM employs unsupervised machine learning and knowledge acquisition heuristics for learning from experience. We have experimented on the feasibility of CSPM using recorded teaching scenarios.

1. Introduction

Instructional planning is the process of choosing a combination of teaching activities to achieve a pedagogical goal. Its use in tutoring, coaching, cognitive apprenticeship, or Socratic dialogue can provide consistency, coherence, and continuity to the teaching process [Vassileva 96], in addition to achieving selected teaching goals [MacMillan 87].

Even though the diagnostic capability of an intelligent tutoring system (ITS) enables it to adapt its instruction or remediation to the needs of individuals [Sison 00], it is possible that students are not at all learning due to deficiency in instructional plans. This may be attributed to inflexible rule-based plans that degenerate and fail in unforeseen situations and to the ITS's inability to utilize experience as background knowledge for subsequent plan improvement. Hence, the compelling need is for an ITS to self-improve its plans in a dynamic way.

Despite the identification and reiteration of several researchers of the need for self-improvement, only a few computerized tutors possess such capability (refer to [Dillenbourg 89]). A *self-improving* tutor is ca-

pable of revising existing and/or learning new plans in response to any perceived deficiency in existing plans. This capability has been addressed by incorporating information (e.g., student-provided solutions that were proved better than the system's solutions) to extend the system's domain knowledge (e.g., [Kimball 82, Duchastel 86]) and/or by refining the rule composition and rule-selection mechanism that implement the plans to improve tutoring knowledge (e.g., [O'Shea 79, Dillenbourg 89]).

Our research aims to provide self-improvement capabilities by infusing category modeling and learning from experience to instructional planning. Plenty of research effort has been devoted to adapting instruction so as to work for *all* students. However, evidence show that when students are grouped based on certain common characteristics and adaptation is made distinct on the level of these groups, the effectiveness of the tutoring system increases [Arroyo 00]. We hypothesize that category knowledge can support an ITS tutor's self-improvement. In this sense, the student categories can be both contributors and recipients of effective plans. A plan is *effective* if at the end of it, the set pedagogical goal is achieved. Learning

Fig. 1 The functional view of CSPM as configured in a tutoring system.

from experience suggests that plans, and the learning process that derives these, can be realized from the circumstances or scenarios that relate one plan to other plans located in different times.

The crux of this work is a learning framework that is embodied by a *Category-based Self-improving Planning Module* (CSPM) within an ITS tutor agent that performs knowledge integration and discovery. Knowledge integration includes constructing models of different categories of students and unifying relevant plans into a single search space. Knowledge discovery innovatively utilizes an unsupervised machine learning (ML) technique and knowledge acquisition heuristics for learning from experience to find the effective plan. ML paradigms have been successfully applied in computerized tutors in various ways: to infer student models (as reviewed in [Sison 98]), to optimize teaching responses to students (e.g., [Beck 02, Mayo 01]), and to evaluate a tutor and understand how learning proceeds through simulated students (e.g., [VanLehn 94, Beck 02]). We utilize the cohesion information-theoretic metric and matching and sampling heuristics to guide a Q-learning algorithm in finding and improving plans for various learner categories.

In the rest of this paper, we first provide an overview of CSPM and describe the methodology used to experiment on its feasibility using real-world data. We then discuss the learning approaches of CSPM, and for each approach, we report experimental results that demonstrate its usefulness. Finally, we state our future direction and concluding remark.

2. Functional Design of CSPM

Figure 1 shows the functional design of CSPM, as well as its external relationships, to configure a complete tutoring system. CSPM reasons using high level of abstraction, i.e., it only decides the sequence of activities based on its category knowledge, and leaves the fine-grained interaction (e.g., delivery, assessment, diagnosis, etc.) to the Teaching Module (TM). It assumes that the TM is the one that directly interacts

with the learner and is responsible for determining the context of the tutorial session.

Category modeling, i.e., the process of building and updating models of different student categories, allows CSPM to plan based on the needs of the members of the category. A category model is an incremental interpretation of its members' characteristics whose main purpose is to reproduce the current state of the members' knowledge, their learning profile with its strengths and weaknesses, and the plans that are likely to work best for the category. CSPM updates the set of models in memory by adding a new category or modifying an existing one.

Using category knowledge, CSPM constructs a *map* of alternative plans, which is a directed graph of related teaching activities that need to be carried out in sequence. The map becomes a tool for CSPM to explore different plans. CSPM relays its plan's activities to the TM. The TM is assumed to map each teaching activity to a procedural knowledge that selects the appropriate domain content and executes the activity. Both procedural and content knowledge may be stored in one domain knowledge base. The TM feedbacks the outcome to CSPM, and based on this feedback, CSPM self-improves its plan accordingly.

3. Description of Experiments

3.1 Experimentation Methodology

From the architectural point of view, by encapsulating pedagogic decision making aspect in CSPM, CSPM's learning capabilities can be tested while isolating it from the other modules. This testing is based on two premises validated by the difficulty of testing the individual modules in an ITS [Mark 93]: (1) In order to lessen the influence of other ITS modules, the experimentation is focused solely on CSPM. (2) Moreover, in order to perform significant experiments under the same initial conditions, CSPM is tested using teaching scenarios of actual human tutorial sessions. We used recorded scenarios since CSPM needs only to do coarse-grained pedagogic reasoning while

Fig. 2 An example of a case in [Reyes 02a] that was adapted to become a teaching scenario for CSPM.

leaving the fine-grained interaction to the TM. This testing is similar to a layered evaluation framework [Brusilovsky 01, Karagiannidis 00] and opens CSPM to the benefits of an ablative assessment [Beck 02].

Experimentation is performed in two stages. In the first stage, an initial set of student category models is constructed and their usefulness is observed. The second stage is a two-fold phase: (1) category knowledge is utilized to construct a map of candidate plans, hence the term *category-effected* map formation task, and (2) the development of the same teaching scenario is simulated using different plans in the map and the results are calculated in terms of the changes in the effectiveness level of the derived plans. The details are discussed in their corresponding sections.

3.2 Experimentation Data

For our data, we borrowed the case library used in [Reyes 02a], which was populated from verbal protocols of interactions between two expert instructors and 24 qualified peer tutors on the one hand and 120 students on the other. The tutees were Computer Science freshmen in a C Programming course. The instructors and tutors were asked to record their interactions with their tutees and the cases were identified from these verbal protocols. An average of three tutorial sessions were conducted for each student. Out of these sessions, 105 unique cases were identified. We adapted these cases to form our initial set of scenarios (an example is shown in Figure 2). A detailed discussion of the case structure can be found in [Reyes 02b], while the next section discusses the scenario's data representation. We only selected the attributes relevant to, and sufficient for, our experimentation.

A *teaching scenario* defines a previously used instructional plan and contains the circumstance by which it was applied successfully. Each scenario consists of (1) *student* attributes: cognitive ability (low,

moderate, or high), learning style (visual, auditory, or kinesthetic), knowledge scope, and list of committed errors; (2) *session* attributes: session goal and topic tackled; and (3) the corresponding successful plan. Cognitive ability was measured in terms of the student's performance in problem-solving tasks conducted in class prior to the tutorial session. Learning style was assessed using a learning style inventory. The topics were classed and sequenced in a taxonomy based on similarity in syntax, semantics, and/or purpose of their constructs. The knowledge scope indicates the last topic taken by the student following the sequence in the taxonomy. The student's errors in solutions to particular problems were recorded. Identifying the student's learning and knowledge states in the protocols is outside the scope of, and is abstracted from, our research. All these were already covered and performed beforehand in [Reyes 02a], and the cases were only made available to us.

Each case contained a plan, effective or otherwise. For ineffective plans, however, repairs which can render them effective were indicated (last row of the case structure). The effectiveness of the plans depended on the feedback of the students when the plans were used on them by the experts. If the students interrupted the session and made a request (e.g., give another example), then the plan is said to be ineffective. The scenarios contain only the effective states of the plans, i.e., the result when the *Repair Strategy* is performed.

4. Category Modeling and Planning

Should we expect categorization to be relevant to the selection of instructional plans? This is so if plans can be made significant on the level of the category models. To show this empirically as the first stage of our experimentation, we let CSPM form the category models by grouping together teaching scenarios with

Fig. 3 Two of the initial category models formed by CSPM.

identical student attribute values. As far as CSPM is concerned, the values of the attributes were already determined a priori and would just need to be input by the TM. In the absence of the TM, since we are testing CSPM in isolation, we do the inputting for now. Once the scenarios are grouped, CSPM subsequently creates the category model structure.

The category model structure is a tree of depth four (as shown in Figure 3). The root node contains the vector of student attribute values: $\langle \text{cognitive_ability}, \text{learning_style}, \text{knowledge_scope}, (\text{error_list}) \rangle$, and category membership is conjunctive on these values (i.e., identical features). The session attribute values in the intermediary level nodes distinguish plans in terms of the goal (2nd level) and topic (3rd level). The *goal* is a set of ordered subgoals, $G = \langle sg_1, \dots, sg_n \rangle$, and the *topic* is a set of one or more topics, $T = (to_1, \dots, to_m)$, that normally belong to the same class in the topic taxonomy. The leaf nodes contain the plans implemented as ordered lists of teaching activity representations, $P = \langle A_1(\cdot), \dots, A_p(\cdot) \rangle$. The representation can be in the form $A_i(\text{TO}, \text{LS}, \text{DL}, [\text{LE}])$ where A_i refers to a kind of teaching activity belonging to a class of instructional strategies specified in [Jonassen 99], TO specifies what the *activity object* (e.g., the next difficult exercise the student will work on) is about, LS is the learning style, DL is the activity object’s difficulty level, and LE points to the student’s error list. [Note: Only A_i and the DL are specified in Figure 3’s plans due to space constraints.] Again, the TM is assumed to take each A_i and maps it to a procedural knowledge that selects the appropriate activity object and executes A_i . Any path from the root to one of the leaf nodes specifies the plan that is supposed to work best for the context specified by the path.

The model representation and category formation are quite straightforward. This is sufficient since we only wanted to acquire a comprehensible explanation of why a category possesses certain plans. Our primary focus is how to utilize effectively the knowledge

provided by the categories. When CSPM conjunctively grouped the 105 teaching scenarios, 78 initial category models were formed. Figure 3 shows two of the initial models. With low-level learners in category A, support comes through easy to grasp activity objects and with the tutor providing sufficient guidance through feedback, review, advice, and motivation. With B’s moderate-level learners, however, the tutor can afford to scaffold while increasing the difficulty level of the activity objects. Introducing a new topic (in A2 and A3) or comparing it to a previous topic (in A4), for example when moving from FOR to WHILE or DO-WHILE, is characterized by activities that preteach vocabulary (e.g., provide definition, compare concepts), contextualize instruction (e.g., explain relevance of topic, motivate the learner, review previous topic), and integrate new knowledge using examples, exercises and practices, and feedback about the new topic; while reference to a previous topic (in B1) may call for summarization or further internalization (e.g., through simulated practices).

5. Machine-learning Capabilities

What CSPM has at this point is an incomplete and imperfect knowledge of its category models. Hence, incremental learning is necessary to complete the category modeling process. In both building new models and updating existing ones, the primary challenge lies in deriving and self-improving the instructional plans in the models. CSPM addresses this challenge by utilizing unsupervised ML techniques and heuristics for learning from experience.

5.1 Provision for an Initial Modifiable Plan

An essential knowledge that CSPM provides is the initial *local plan*, i.e., a plan in the category model to which the student is classified. Though it is logical for CSPM to expect that a local plan is the best to apply to the student, it is not guaranteed to immediately

Fig. 4 Descriptions of the (a)new category, and (b)nearest category and its selected branch and plan.

work. A more accurate behavior is for CSPM to acquire this plan, but then slowly adapt and improve it to fit the student. However, if the student is classified to a new category, which implies the absence of a local plan, where and how can CSPM find an initial plan to work with? A *new category* is formed when none of the existing models have student attribute values that exactly match those of the current student.

In the absence of a local plan, it is plausible to find the alternative plan in the *nearest category*, i.e., the category least distant to the new one in terms of learner features. To find this category, CSPM employs the *cohesion* information theoretic metric. Unlike a Euclidean distance metric that sums attribute values independently, cohesion is a distance measure in terms of relations between learner attributes. [Talmon 93] provides an elaborate discussion of this metric. Briefly, cohesion (C_C) is defined as $C_C = W_C / O_C$ where W_C is the average distance between the learners of category C , and O_C is the average distance between C and all other categories. The most cohesive category is the one that best maximizes the similarity among its members while concurrently minimizing its similarity with other categories. CSPM pairs the new category to one of the existing categories and treats this pair as one category, say NE. Cohesion score can now be computed for NE and the rest of the existing categories ($C_{NE} = W_{NE} / O_{NE}$). The computation is repeated, pairing each time the new category with another existing one, until the cohesion score has been computed for all possible pairs. The existing category in the pair that yields the lowest C_{NE} is the nearest.

Once CSPM learns the nearest category R for the new category S , it then proceeds to the goal and topic nodes in R that are identical to that of S . If no identical nodes exists, CSPM will select the best matching i^{th} goal node in R specified by g , and the best matching j^{th} topic node under g specified by t :

- $g = \arg \max_i \{ \eta(G_S \cap G_{R,i}) + \eta(\varphi(G_S) \cap \varphi(G_{R,i})) \}$
- $t = \arg \max_j \{ \eta(T_S \cap T_{g,j}) \}$

$\eta(\zeta)$ returns the number of elements in ζ and $\varphi(G)$ returns the set of paired adjacent subgoals in G . For example, if $G_S = \langle a, b, c \rangle$, then $\eta(\varphi(G_S))$ will return $\eta(\langle \langle a, b \rangle, \langle b, c \rangle \rangle) = 2$. The second part of the computation for g places importance on how much the subgoals in G_S and $G_{R,i}$ are similar in order. For example, if $G_{R,1} = \langle a, b, d \rangle$ and $G_{R,2} = \langle a, d, c \rangle$, then the first goal node in R becomes g . The t is obviously maximized when topics in T_S and $T_{g,j}$ belong to the same class in the taxonomy since they are similar in syntax, semantics, and/or purpose. In case many goal nodes satisfy g , CSPM can look ahead to see which node will yield the best t result, and then selects that one as g . Figure 4b illustrates the R learned using cohesion for the S in Figure 4a (this was derived from a scenario that is not in the initial dataset that CSPM categorized), as well as the selected g and t nodes. Once the best path is selected, CSPM acquires its plan and forwards the plan to the next learning task where it can be utilized.

5.2 Category-effected Map Formation Task

For CSPM to provide one plan instead of another implies adaptivity and sensitivity to the student. This is demonstrated by means of a *map* of alternative plans. Table 1 outlines CSPM's algorithm for constructing this map. Initially, the map is formed as a union of plans that worked for existing categories whose goals are identical to, or is one of the subgoals of, the current session goal G_C (step 1). G_C may be from a new (e.g., Figure 4a) or existing category that becomes the target category for an effective plan. For example, if $G_C = \langle a, b, c \rangle$ then only the plans under any G_j with subgoals $\langle a, b, c \rangle$, $\langle a \rangle$, $\langle b \rangle$, or $\langle c \rangle$ can be included. CSPM considers only the actual subgoals in G_C since the manner by which relevant activities should be sequenced can be seen in the goal. At this stage of looking into the feasibility of CSPM, we considered only the minimum number of goal types (i.e., all-subgoals and one-subgoal G_j 's) that can still

Table 1 The category-effected map formation algorithm

1. Construct the initial map M as $(p_1 \cup p_2 \cup \dots \cup p_n)$ where p_i is a plan under goal G_j of an existing category
 - Include p_i in M if and only if $G_j = G_C \vee (G_j = \langle sg_x \rangle \wedge sg_x \in G_C)$ where G_C is the current session goal
2. Minimize M by retaining an activity A and its edges if and only if:
 - A is in the set of activities $S_{candidates} = \bigcup_k f(learner_attribute_k, value_k) \cap \bigcup_m f(topic, value_m)$.
The m topics include the current topic and all topics in its class as specified in the topic taxonomy.
 - A follows specified tutorial norms.
 - A and its edges belong to a path that constitutes all the ordered subgoals of G_C .
3. Apply the *plan-map matching* heuristic over M

Fig. 5 The results of the first two steps of the map formation algorithm: (a) Step 1 and (b) Step 2.

provide, and lead to, effective plans. Though the algorithm can be easily revised to include paired adjacent subgoals (e.g., G_j with subgoal $\langle a, b \rangle$ or $\langle b, c \rangle$), given our initial results, the exclusion of paired subgoals seems to be a reasonable simplification at this stage. Definitely, as we continue to harness the capability of CSPM, paired subgoals should be considered. Certain activities in the initial map may still not apply to the student since this map was formed with only the goal in mind, not yet considering the learner and topic attributes. Hence, CSPM minimizes the map and retains in it an activity A based on specific membership conditions (step 2).

Function $f(attribute_name, value)$ returns the set of activities that worked for existing categories that have the specified attribute and value parameters. For example, with $f(learning_style, VISUAL)$, CSPM will seek all categories having visual learners and includes in $S_{candidates}$ all activities found in these categories. If the current topic is FOR, CSPM will search through the topic taxonomy and will find that FOR belongs to the iterative loop class, and with it are WHILE and DO-WHILE. CSPM will then execute $f(topic, FOR) \cup f(topic, WHILE) \cup f(topic, DO_WHILE)$. As for following certain tutorial norms, an example is that giving feedback should be among the activities to succeed an end-point pre-test in case this results to poor student performance. Lastly, if A belongs to a path that passes through all the ordered subgoals, then it contributed to a coherent integration of the subgoals.

Figure 5 shows the results of the first two steps of the map formation task for the new category in Fig-

ure 4a. Each edge indicates the *transition* A_i to A_j , and its value indicates its *frequency*, which is simply the number of times a transition occurred for a particular subgoal. [Note: The different transition shadings in Figure 5 indicate the subgoals to which the transitions belong, and a two-character code corresponds to an activity.] An activity may be present in various plans thereby becoming an intersection point for these plans. In Figure 5b, an activity is removed from the initial map if CSPM does not find it in $S_{candidates}$, and if it has activities before and after, it is replaced by a transition edge (e.g., $d8$ and $b4$). Activities and/or edges are also removed if they failed to integrate through all subgoals. For example, though dm is in $S_{candidates}$, it did not have a preceding activity that connects it to the first subgoal. Hence, dm and its edges were removed. Lastly, multiple edges between A_i and A_j are merged and their frequencies are summed (e.g., $c5 \rightarrow c6$). Figure 6 shows the actual map corresponding to Figure 5b.

To avoid inefficient random selection of activities, CSPM uses a *plan-map matching* heuristic (step 3) to find the i_{th} path in map M that preserves most the sequence of activities in the selected plan P_R in the nearest category R . This path is specified by p :

- $p = \arg \max_i \{ \eta(P_R \cap P_{M,i}) + \eta(\varphi(P_R) \cap \varphi(P_{M,i})) \}$

$\varphi(P)$ will return the set of paired adjacent activities in plan P . Again, the second part of the equation weights the sequencing of activities in P . Once path p is selected, CSPM finds the minimal subpath p' in p that contains all activities included in $sbp = P_R \cap P_{M,p}$ with the first and last activities in p' being in sbp . All

Fig. 6 The final map for the new category model in Figure 4(a).

Table 2 The Q-learner's algorithm

Initialize all $Q(s,a)$ to zero
Construct map M given the student and session attribute values of the teaching scenario (refer to Table 1)
Repeat (for x number of stages or sessions)
Initialize state s to <i>nil</i>
Repeat (for each transition in M until the end of M is reached)
If <i>greedy_time</i> select activity a , given s , that yields to the highest $Q(s,a)$ value
Else select a following the <i>sampling heuristic</i>
Observe reward r and next state s'
$Q(s,a) = Q(s,a) + r + \gamma \max_{a'} Q(s',a') - Q(s,a)$
$s = s'$

transitions that branch away from p' will be disregarded during exploration. To demonstrate using the P_R in Figure 4b-left and the map in Figure 6, matching will result to $p' = \langle a3, df, d5, b1, b3, c5, c6 \rangle$. Due to the sequencing of activities in p' , *reviewRepeat* is now in front, which is valid since it is effective to re-read/rehearse what was taught before comparing the concepts. *giveNonExample* in P_R was not found in p' since it was not selected during steps 1-2.

5.3 Plan Exploration and Exploitation

Instead of using adhoc exploration heuristics to learn an effective plan, CSPM employs *Q-learning* [Watkins 92]. As a reinforcement learner [Sutton 98], Q-learning can process an experience generated from interaction with minimal amount of computation, and becomes more efficient when given background knowledge.

Table 2 outlines the Q-learning algorithm. The Q-learner has 2 inputs and 2 outputs. The inputs are: (1) the state s , which refers to the set of activities the student has previously performed, and implicitly, the set of activity objects he/she has received, and (2) the reward r for executing a teaching activity (e.g., +1 if the student makes no request for other activities; otherwise, -1). The outputs are: (1) a teaching activity a , and (2) as a side effect, it also updates the $Q(s,a)$ function, which is actually a table. Any $Q(s,a)$ value in the table represents the “goodness” of taking activity a in state s . The values in the table when convergence has taken place would be a specification

of the effective plan, i.e., the effective plan can be derived from the table by selecting at every step for s the activity a that will yield the highest $Q(s,a)$ value. The Q-learner uses an internal ϵ -greedy policy. At a certain percentage of the time, ϵ , it explores the map for another version of the plan rather than the one it thought was best. This helps prevent it from getting stuck to a sub-optimal version. Over time, ϵ is gradually reduced. The Q-learner's goal is to maximize the reward it receives in the long run.

Let us illustrate using the map in Figure 6. The initial state is $s = \{ \}$. In the first iteration, say that the Q-learner selected *reviewRepeat* ($(to_1, to_2), visual, easy$) where to_i is a topic at hand. After which, the TM assessed the activity, which may be good (e.g., the student understood the review) or bad (e.g., the student skipped it). Consequently, $r = v$ (v denotes a positive or negative reinforcement), and the new state after executing a becomes $s' = \{ review_made(to_1, to_2) \}$. If at the next step the Q-learner selected *compareConcepts* ($(to_1, to_2), visual, easy$), then $s' = \{ review_made(to_1, to_2), comparison_received(to_1, to_2) \}$. The next steps are performed, and for each step s and $Q(s,a)$ are updated.

The Q-learner is assisted by a sampling heuristic. This heuristic *samples first* the most frequent transitions since they are more likely to be correct. Once sampled, the frequency becomes zero to give way to another transition in the next iteration. When all edges have been sampled, they are re-initialized with their original frequencies, and the sampling process is

repeated. In Figure 6, let us say that $c1$ has just been executed. The Q-learner will try $c1 \rightarrow \text{END}$ since its edge frequency is 5, and it will try $c1 \rightarrow c9$ in the next iteration. In case of ties, the Q-learner will choose arbitrarily. This heuristic is overridden by p' (in 5.2). For example, though $\text{START} \rightarrow d7$ has frequency 5, $\text{START} \rightarrow a3$ alone shall be performed since it is in p' .

5.4 Fold-two of the 2nd Experiment Stage

To simulate the Q-learner’s capability, a set of fourteen new teaching scenarios that vary in their level of required learning task was used in three different experiment set-ups: (a) the Q-learner utilizes both the plan-map matching and sampling heuristics, (b) only the sampling heuristic was used (CSPM did not compute p'), and (c) the Q-learner randomly selected from all possible transitions. Each experiment simulates the development of the same scenario for 50 successive stages where each *stage* is characterized by a modified version of the Q-learner’s plan. We want to know (i) if and when the Q-learner can learn the effective plans expected for these scenarios, and (ii) if plan-map matching significantly aided in efficiently deriving the effective plans. The Q-learner’s performance is measured in terms of the mean effectiveness of the plans in the i_{th} stage across all m scenarios:

- $\text{Mean Effectiveness}(\text{ME})_i = \Sigma_m \Sigma_n Q_{m,n}(s,a) / nm$

A path from START to END in map M is interpreted as a plan. In every stage, only one path in M is selected. n is the number of transitions in the selected path. M varies depending on the current scenario and n varies depending on the M used. n is used in the equation to normalize ME. An ME of 1.0 means the plan is 100% effective. On matters of implementation, the $Q(s,a)$ table and the map were implemented as array matrices, with the latter being an adjacency matrix. We empirically set the discount factor γ to 1 putting emphasis on the overall effect of the plan.

To simulate how the Q-learner can be rewarded, curriculum experts were consulted on what could be the effective plans given the new scenarios’ attributes. This seemed plausible given the experts’ years of experience that permit them to make reliable generalizations on how to teach effectively. Experts agree that, in this domain, there can be more than one way to teach effectively. Consequently, an expert’s recommended plan can be interpreted as an effective plan P_E in the space of effective plans. For the meantime, however, we assigned only one P_E to each test scenario and then tested the Q-learner if it can find this

Fig. 7 The Q-learner’s learning performance.

P_E . If the Q-learner came up with the correct sequence of activities as that of P_E , +2 is given as end reward; otherwise, -1. As intermediary rewards, +1 is given if the current activity and its place in the sequence are correct; otherwise, -1. Reward values were determined empirically as suitable.

Figure 7 shows the Q-learner’s plans becoming effective asymptotically over time. It would take a longer time to learn an effective plan using alone the sampling heuristics or random selection. This means that the plan from the nearest category used in plan-map matching significantly helped minimize learning time. However, it can be expected that even with this heuristic, the Q-learner may still even take a very long time to find P_E in certain scenarios, or even fail if no P_E exists in the map. This is natural given the imperfect knowledge in the category models that leads to insufficient transitions in the map. Consequently, performance will be less than good. Here, performance is good since the map was sufficient to account for most of the new test scenarios. But what if it is insufficient? There are at least two possible solutions to this. One is to make the initial set of category models nearly complete by providing much more initial teaching scenarios. As more category background knowledge are constructed prior to system deployment, a good asymptotic behavior can be achieved. The second is to provide other credible sources of effective activity transitions, such as appropriate learner’s feedback, which can be incorporated as new workable paths to be evaluated in the succeeding sessions. The latter becomes practical and useful when learning during on-line interactions.

6. Future Work and Conclusion

Most existing ITS planners assume that the plans they derive work for all users. Evidence show, however, that learners may need to be taught differently according to differences in their learning patterns. In

fact, we demonstrated that these differences, when utilized by unsupervised ML techniques and heuristics, powerfully cued CSPM to self-improve its plans.

This research in progress can be completed and improved in various ways. A very interesting study is to find a correlation between polymorphous categories and instructional planning. Another, as previously mentioned, is to find a mechanism to include learner feedback in the planning process that can be translated into additional paths which CSPM can subsequently utilize. Lastly, it would be very challenging to explore the idea of a pure bottom-up approach of discovering learning theories between categorization and planning by mining raw interaction data logs.

◇ References ◇

- [Arroyo 00] Arroyo, I., Beck, J., Woolf, B., Beal, C., and Schultz, K.: Macro-adapting AnimalWatch to gender and cognitive differences with respect to hint interactivity and symbolism, *Lecture Notes in Computer Science*, Vol. 1839, pp. 574–583 (2000)
- [Beck 02] Beck, J.: Directing development effort with simulated students, *Lecture Notes in Computer Science*, Vol. 2363, pp. 851–860 (2002)
- [Brusilovsky 01] Brusilovsky, P., Karagiannidis, C., and Sampson, D.: The benefits of layered evaluation of adaptive applications and services, in *8th International Conference on User Modeling, Workshop on Empirical Evaluations of Adaptive Systems* (2001)
- [Dillenbourg 89] Dillenbourg, P.: The design of a self-improving tutor: PROTOTECH, *Instructional Science*, Vol. 18, No. 3, pp. 193–216 (1989)
- [Duchastel 86] Duchastel, P. and Imbeau, J.: Instructible ICAI, in *Doc. 86-07 du Laboratoire d'Intelligence Artificielle en Education* (1986)
- [Jonassen 99] Jonassen, D., Tessmer, M., and Hannum, W.: *Task analysis methods for instructional design*, Mahwah, NJ: Lawrence Erlbaum Associates, Inc. (1999)
- [Karagiannidis 00] Karagiannidis, C. and Sampson, D.: Layered evaluation of adaptive applications and services, *Lecture Notes in Computer Science*, Vol. 1892, pp. 343–246 (2000)
- [Kimball 82] Kimball, R.: A self-improving tutor for symbolic integration, in Sleeman, D. and Brown, J. eds., *Intelligent Tutoring Systems*, pp. 283–307, Academic Press, London (1982)
- [MacMillan 87] MacMillan, S. and Sleeman, D.: An architecture for a self-improving instructional planner for intelligent tutoring systems, *Computational Intelligence*, Vol. 3, pp. 17–27 (1987)
- [Mark 93] Mark, M. and Greer, J.: Evaluation methodologies for intelligent tutoring systems, *International Journal of Artificial Intelligence in Education*, Vol. 4, pp. 129–153 (1993)
- [Mayo 01] Mayo, M. and Mitrovic, A.: Optimising ITS behaviour with Bayesian networks and decision theory, *International Journal of Artificial Intelligence in Education*, Vol. 12, pp. 124–153 (2001)
- [O'Shea 79] O'Shea, T.: *Self-improving tutoring systems*, Birkhauser Verlag, Basel (1979)
- [Reyes 02a] Reyes, R.: *A case-based reasoning approach in designing explicit representation of pedagogical situations in an intelligent tutoring system*, PhD thesis, College of Computer Studies, De La Salle University-Manila (2002)
- [Reyes 02b] Reyes, R. and Sison, R.: Using case-based reasoning approach in planning instructional activities, *Lecture Notes in Computer Science*, Vol. 2417, pp. 502–511 (2002)
- [Sison 98] Sison, R. and Shimura, M.: Student modeling and machine learning, *International Journal of Artificial Intelligence in Education*, Vol. 9, pp. 128–158 (1998)
- [Sison 00] Sison, R., Numao, M., and Shimura, M.: Multi-strategy discovery and detection of novice programming errors, *Machine Learning*, Vol. 38, pp. 157–180 (2000)
- [Sutton 98] Sutton, B. and Barto, A.: *Reinforcement learning: An introduction*, MIT Press, Cambridge, MA (1998)
- [Talmon 93] Talmon, J., Fonteijn, H., and Braspenning, P.: An analysis of the WITT algorithm, *Machine Learning*, Vol. 11, pp. 91–104 (1993)
- [VanLehn 94] VanLehn, K., Ohlsson, S., and Nason, R.: Applications of simulated students: An exploration, *International Journal of Artificial Intelligence in Education*, Vol. 5, No. 2, pp. 135–175 (1994)
- [Vassileva 96] Vassileva, J. and Wasson, B.: Instructional planning approaches: From tutoring towards free learning, in *Proceedings of European Conference on Artificial Intelligence in Education*, pp. 1–8 (1996)
- [Watkins 92] Watkins, J. and Dayan, P.: Q-learning, *Machine Learning*, Vol. 8, pp. 279–292 (1992)

〔担当委員：柏原 昭博〕

Received April 19, 2005.

Author's Profile

Legaspi, Roberto

Mr. Legaspi is a doctoral student of the Department of Architecture for Intelligence of the Institute of Scientific and Industrial Research, Osaka University. He received his M.S. in Computer Science in 1995 from De La Salle University-Manila, Philippines.

Sison, Raymund

Dr. Sison is a professor of the College of Computer Studies of De La Salle University-Manila. His research interest lies in understanding how AI techniques can be used to improve ICT support of e-learning and knowledge management. His research group is working on a framework for intelligent e-learning support whose key features are learner agents with modeling and collaborative capabilities, and ontologies. He obtained his Ph.D. in Computer Science from Tokyo Institute of Technology.

Numao, Masayuki (Member)

Dr. Numao heads the Department of Architecture for Intelligence of the the Institute of Scientific and Industrial Research, Osaka University. He received his Ph.D. in Computer Science in 1987 from Tokyo Institute of Technology, and worked in its Department of Computer Science in 1987–2003. He was also a visiting scholar at CSLI, Stanford University in 1989–1990. His research interest lies in developing and implementing constructive and highly adaptive interfaces and data mining approaches using advanced machine learning paradigms.