Learning Procedural Knowledge from User Solutions to Ill-Defined Tasks in a Simulated Robotic Manipulator

Philippe Fournier-Viger[1], Roger Nkambou[1] and Engelbert Mephu Nguifo[2]

[1]Department of Computer Science, University of Quebec in Montreal, Canada

[2] Department of Computer Sciences, Université Blaise-Pascal Clermont 2, Clermont-Ferrand, France

fournier_viger.philippe@courrier.uqam.ca, nkambou.roger@uqam.ca, mephu@isima.fr

INTRODUCTION

Domain experts should provide relevant domain knowledge to an Intelligent Tutoring System (ITS) so that it can assist a learner during problem-solving activities. There are three main approaches for providing such knowledge. The first one is cognitive task analysis that aims at producing effective problem spaces or task models by observing expert and novice users [1] to capture different ways of solving problems. However, cognitive task analysis is a very time-consuming process [1] and it is not always possible to define a complete or partial task model, in particular when a problem is ill-structured. According to Simon [2], an ill-structured problem is one that is complex, with indefinite starting points, multiple and arguable solutions, or unclear strategies for finding solutions. Domains that include such problems and in which tutoring targets the development of problem-solving skills are said to be ill-defined (within the meaning of Ashley et al. [3]). Constraint-based modeling (CBM) was proposed as an alternative [4]. It consists of specifying sets of constraints on what is a correct behavior, instead of providing a complete task description. Though this approach was shown to be effective for some ill-defined domains, a domain expert has to design and select the constraints carefully. The third approach consists in integrating an expert system into an ITS [8]. However, developing an expert system can be difficult and costly, especially for ill-defined domains.

Contrarily to these approaches where domain experts have to provide the domain knowledge, a promising approach is to use knowledge discovery techniques to automatically learn a problem space from logged user interactions in an ITS, and to use this knowledge base to offer tutoring services. A few efforts have been made in this direction in the field of ITS [5, 6, 17, 21, 22, 23]. But they have all been applied in well-defined domains. As an alternative, we propose in this chapter an approach that is specially designed for ill-defined domains. The approach has the advantage of taking into account learner profiles and does not require any form of background knowledge. The reader should note that a detailed comparison with related work is provided in section IV of this chapter.

This chapter is organized as follows. First, we describe the CanadarmTutor tutoring system and previous attempts for incorporating domain expertise into it. Next, we present our approach for extracting partial problem spaces from logged user interactions and describe how it enables CanadarmTutor to provide more realistic tutoring services. Finally, we compare our approach with related work, discuss avenues of research, and present conclusions.

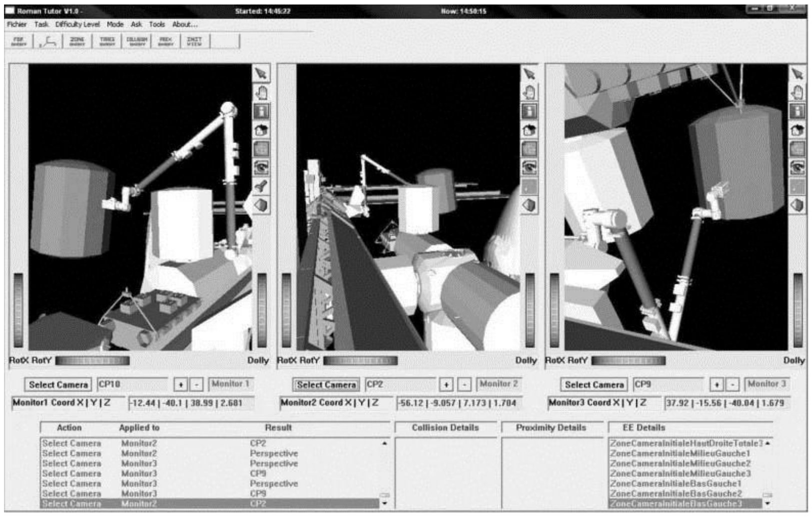
I. The CanadarmTutor Tutoring System

CanadarmTutor [8] (depicted in Figure 1) is a simulation-based tutoring system to teach astronauts how to operate Canadarm2, a 7 degrees of freedom robotic arm deployed on the International Space Station (ISS). The main learning activity in CanadarmTutor is to move the arm from a given configuration to a goal configuration. During the robot manipulation, operators do not have a direct view of the scene of operation on the ISS and must rely on cameras mounted on the manipulator and at strategic places in the environment where it operates. To move the arm, an operator must select at every moment the best cameras for viewing the scene of operation among several cameras mounted on the manipulator and on the space station.

To provide domain expertise to CanadarmTutor, we initially applied the expert system approach by integrating a special path-planner into CanadarmTutor [8]. The path-planner can generate a path avoiding obstacles between any two arm configurations. The path-planner makes it possible to track a learner solution step by step, and generate demonstrations when necessary. However, the generated paths are not always realistic or easy to follow, as they are not based on human experience, and they do not cover other aspects of the manipulation task such as selecting cameras and adjusting their parameters. Also, it cannot support important tutoring services such as estimating knowledge gaps of learners as there is no knowledge or skills representation.

In subsequent work, we attempted to describe how to manipulate Canadarm2 as a set of rules with a rule-based knowledge representation model by applying the cognitive task analysis approach [9]. Although we described high-level rules such as how to select cameras and set their parameters in the correct order, it was not possible to go into finer detail to model how to rotate the arm joint(s) to attain a goal configuration. The reason is that for a given robot manipulation problem, there is a huge number of possibilities for moving the robot to a goal configuration and because one must also consider the safety of the manoeuvres and their easiness, it is very difficult to define a "legal move generator" for generating the moves that a human would execute. In fact, some joint manipulations are preferable to others depending on several criteria that are hard to formalize such as the view of the arm given by the chosen cameras, the relative position of obstacles to the arm, the arm configuration (e.g. avoiding singularities) and the familiarity of the user with certain joint manipulations over others. It is thus not possible to define a complete and explicit task model for this task. Hence, CanadarmTutor operates in an ill-defined domain as defined by Simon.

The constraint-based modeling approach [4] may represent a good alternative to the cognitive task analysis approach. However, in the CanadarmTutor context, it would be very difficult for domain experts to describe a set of relevance and satisfaction conditions that apply to all situations. In fact, there would be too many conditions and still a large number of solutions would fit the conditions for each problem. Moreover, the CBM approach is useful for validating solutions. But it cannot support tutoring services such as suggesting next problem-solving steps to learners, which is a required feature in CanadarmTutor.



CP10 — Monitor1 Coord X | Y | Z: -12.44 | -40.1 | 38.99 | 2.681

CP2 — Monitor2 Coord X | Y | Z: -56.12 | -9.057 | 7.173 | 1.704

CP9 — Monitor3 Coord X | Y | Z: 37.92 | -15.56 | -40.04 | 1.679

Action	Applied to	Result	Collision Details	Proximity Details	EE Details
Select Camera	Monitor2	CP2			ZoneCameraInitialeHautDroiteTotale3
Select Camera	Monitor2	Perspective			ZoneCameraInitialeMilieuGauche1
Select Camera	Monitor3	Perspective			ZoneCameraInitialeMilieuGauche2
Select Camera	Monitor3	CP9			ZoneCameraInitialeMilieuGauche3
Select Camera	Monitor3	Perspective			ZoneCameraInitialeBasGauche1
Select Camera	Monitor3	CP9			ZoneCameraInitialeBasGauche2
Select Camera	Monitor2	CP2			ZoneCameraInitialeBasGauche3

Figure 1. The CanadarmTutor User Interface

II. A Domain Knowledge Discovery Approach for the Acquisition of Domain Expertise

Our hypothesis is that a data mining approach for learning a partial problem space from recorded user interactions can overcome some problems related to the ill-defined nature of a procedural learning domain. Our approach consists of the following steps. Each time a user attempts to solve a problem, the tutoring system records the attempt as a *plan* in a database of user solutions. Then humans can display, edit or annotate plans with contextual information (e.g. required human skills to execute the plan and expertise level of those skills). Thereafter, a data mining algorithm is applied to extract a partial task model from the user plans. The resulting task model can be displayed, edited, and annotated before being taken as input by the tutoring system to provide tutoring services. This whole process of recording plans to extract a problem space can be performed periodically to make the system constantly improve its domain knowledge. Moreover, staff members can intervene in the process or it can be fully automated. The next subsections present in detail each step of this process and how they were applied in CanadarmTutor to support relevant tutoring services.

A. Step 1: Recording Users' plans

In the first phase, the tutoring system records the solutions of users who attempt an exercise. In CanadarmTutor, an exercise is to move the arm from an initial configuration to a goal configuration. For each attempt, a *sequence of events* (or *plan*) is created in a sequence database. We define an event $X = (i_1, i_2, \ldots, i_n)$ as a set of one or more actions $i_1, i_2, \ldots, i_n$ done by a learner that are considered simultaneous, and where each action can specify an integer parameter value (this could be extended to several parameters or other types of values: see [15] for more details). In CanadarmTutor, we defined 112 such actions that can be recorded including (1) selecting a camera, (2) performing an increase or decrease of the pan/tilt/zoom of a camera and (3) applying a rotation value to an arm joint. Formally, we define a sequence of events (based on the definition of Hirate & Yamana [10]) as $s = \langle(t_1,X_1), (t_2, X_2), \ldots, (t_v, X_v)\rangle$ where each event X_k is associated with a timestamp t_k indicating the time of the event. In CanadarmTutor, timestamps of successive events are successive integers (0, 1, 2…). An example of a partial sequence recorded for a user in CanadarmTutor is <(0, $rotateSP\{2\}$), (1, $selectCP3$), (2, $panCP2\{4\}$), (3, $zoomCP2\{2\}$)> which represents decreasing the rotation value of joint SP by two units, selecting camera CP3, increasing the pan of camera CP2 by four units, and then its zoom by two units.

To annotate sequences with contextual information, we have extended the notion of sequence database with dimensional information as suggested by [11]. A sequence database having a set of dimensions $D=D_1, D_2,..., D_p$ is called a multidimensional database (MD-Database*)*. Each sequence of an MD-Database is called an *MD-Sequence* and consists of a plan and annotations. Annotations are a list of values for each dimension. Values can be symbols or the value "*", which subsumes all other values. Such a list of dimension values is called an MD-Pattern and is denoted $d_1,d_2,...d_p$. Table 1 shows an example of a toy MD-Database containing six learner plans annotated with five dimensions. In this table, the single letters a, b, c, and d denote actions. The first dimension “Solution state” indicates if the learner plan is a successful or buggy solution. In the case of CanadarmTutor, values for this dimension are produced by the tutoring system (see section IV). The four other dimensions of Table 2 are example of dimensions that can be added manually. Whereas the dimension “Expertise” denotes the expertise level of the learner who performed a sequence, “Skill_1”, “Skill_2” and “Skill_3” indicate wether any of these three specific skills were demonstrated by the learner when solving the problem. This example illustrates a five dimensions MD-database of three main types (skills, expertise level, and solution state). However, our framework can accept any kind of learner information or contextual information encoded as dimensions. For example, learner information could include age, educational background, learning styles, cognitive traits and emotional state, assuming that the data is available. In CanadarmTutor, we had 10 skills, and used the “solution state” and “expertise level” dimensions to annotate sequences (see section III for more details).

B. Step 2: Mining a Partial Task Model from Users’ plans

In the second phase, the tutoring system applies the data mining framework to extract a partial problem space from users’ plans. To perform this extraction an appropriate method is needed that considers many factors associated with the specific conditions in which a tutoring system such as CanadarmTutor operates. These factors include the temporal dimension of events, actions with parameters, the user's profile, etc. All these factors suggest that we need a temporal pattern mining technique. According to [12], there are four kinds of patterns that can be mined from time-series data. These are trends, similar sequences, sequential patterns, and periodic patterns. In this work we chose to mine sequential patterns [13], as we are interested in finding relationships between occurrences of events in users' solutions. To mine sequential patterns, several efficient algorithms have been proposed. These have been previously applied, for example, to analyze earthquake data [10] and source code in software engineering [24]. While traditional sequential pattern mining (SPM) algorithms have as their only goal to discover sequential patterns that occur frequently in several transactions of a database [13], other algorithms have proposed numerous extensions to the problem of sequential pattern mining such as mining patterns respecting time-constraints [10], mining compact representations of patterns [14, 16, 24], and incremental mining of patterns [24]. For this work, we developed a custom sequential pattern mining algorithm [15] that combines several features from other algorithms such as accepting time constraints [10], processing databases with dimensional information [11], and mining a compact representation of all patterns [14, 16], and that also adds some original features such as accepting symbols with parameter values. We have built this algorithm to address the type of data to be recorded in a tutoring system offering procedural exercises such as CanadarmTutor. The main idea of that algorithm will be presented next. For a technical description of the algorithm the reader can refer to [15].

The algorithm takes as input an MD-Database and some parameters and find all MD-sequences occurring frequently in the MD-Database. Here, sequences are action sequences (not necessarily contiguous in time) with timestamps as defined in section II.A. A sequence $sa = <(ta_1, A_1), (ta_2, A_2), \ldots, (ta_n, A_n)>$ is said to be contained in another sequence $sb = <(tb_1, B_1), (tb_2, B_2), \ldots, (tb_n, B_m)>$, if there exists integers $1 \leq k1 < k2 < \ldots < kn \leq m$ such that $A_1 \subseteq B_{k1}, A_2 \subseteq B_{k2}, \ldots, A_n \subseteq B_{kn}$ and that $tb_{kj} - tb_{k1}$ is equal to $ta_{kj} - ta_{k1}$ for each $j \in 1..n$ (recall that in this work timestamps of successive events are successive integers. e.g. 0, 1, 2…). Similarly for MD-Patterns, an MD-Pattern $Px = dx_1, dx_2, \ldots, dx_p$is said to be contained in another MD-Pattern $Py = dy_1, dy_2, \ldots, dy_p$ if for each $i \in 1..p$, dy_i =“*” or $dy_i = dx_i$ [11]. The relative support of a sequence (or MD-Pattern) in a sequence database D is defined as the percentage of sequences (or MD-Patterns) from D which contain it. The problem of mining frequent MD-sequences is to find all the MD-sequences such that their support is frequent, defined as greater or equal than $minsup$ for an MD-database D, given a support threshold minsup . As an example, Table 2 shows some patterns that can be extracted from the MD-Database of Table 1, with a *minsup* of two sequences (33 %). Consider pattern P3. This pattern represents doing action *b* one time unit (immediately) after action *a*. The pattern P3 appears in MD-sequences S1 and S3. It has thus a support of 33 % or two MD-sequences. Because this support is higher or equal to *minsup*, P3 is frequent. Moreover, the annotations for P3 tell us that this pattern was performed by novice users who possess skills "Skill_1", "Skill_2" and "Skill_3" and that P3 was found in plan(s) that failed, as well as plan(s) that succeeded.

In addition, as in [10] we have incorporated in our algorithm the possibility of specifying time constraints on mined sequences such as minimum and maximum time intervals required between the head and tail of a sequence and minimum and maximum time intervals required between two successive events of a sequence. In CanadarmTutor, we mine only sequences of length two or greater, as shorter sequences would not be useful in a tutoring context. Furthermore, we chose to mine sequences with a maximum time interval between two successive events of two time units. The benefits of accepting a gap of two is that it eliminates some "noisy" (non-frequent) learners' actions, but at the same time it does not allow a larger gap size that could make patterns less useful for tracking a learner's actions.

Table 1. An MD database containing 6 user solutions

ID	Dimensions					Sequence
	Solution state	Expertise	Skill_1	Skill_2	Skill_3	
S1	successful	novice	yes	yes	yes	<(0,a),(1,b c)>
S2	successful	expert	no	yes	no	<(0,d) >
S3	buggy	novice	yes	yes	yes	<(0,a),(1,b c)>
S4	buggy	intermediate	no	yes	yes	<(0,a),(1,c), (2,d)>
S5	successful	novice	no	no	yes	<(0,d), (1,c)>
S6	successful	expert	no	no	yes	<(0,c), (1,d)

Table 2. Some frequent patterns extracted from the dataset of Table 1 with minsup = 33 %

Id	Dimensions					Sequence	Supp.
	Solution State	Expertise	Skill_1	Skill_2	Skill_3		
P1	*	novice	yes	yes	yes	<(0,a)>	33 %
P2	*	*	*	yes	yes	<(0,a)>	50 %
P3	*	novice	yes	yes	yes	<(0,a), (1,b)>	33 %
P4	successful	*	no	*	*	<(0,d)>	50 %
P5	successful	novice	*	*	yes	<(0,c)>	33 %
P6	successful	expert	no	*	no	<(0,d)>	33 %

Another important consideration is that when applying sequential pattern mining, there can be many redundant frequent sequences found. For example, in Table 2, the pattern "*, *novice, yes, yes, yes <(0,a)>*" is redundant as it is included in the pattern "*, *novice, yes, yes yes <(0,a), (1,b)>*" and it has exactly the same support. To eliminate this type of redundancy, we have adapted our algorithm to mine only *frequent closed sequences*. "Closed sequences" [14, 16, 24] are sequences that are not contained in another sequence having the same support. Mining frequent closed sequences has the advantage of greatly reducing the number of patterns found, without information loss (the set of closed frequent sequences allows reconstituting the set of all frequent sequences and their support) [14]. To mine only frequent closed sequences, our sequential pattern mining algorithm was extended based on [14] and [16] to mine closed MD-Sequences (see [15]).

Once patterns have been mined by our sequential pattern mining algorithm, they form a partial problem-space that can be used by a tutoring agent to provide assistance to learners as the next subsection will describe. However, we also provide a simple software program for displaying patterns, editing them, or adding annotations.

C. Step 3: Exploiting the Partial Task Model to Provide Relevant Tutoring Services

In the third phase, the tutoring system provides assistance to the learner by using the knowledge learned in the second phase. The basic operation that is used for providing assistance is to recognize a learner's plan. In CanadarmTutor, this is achieved by the plan recognition algorithm, presented next.

```
RecognizePlan(Student_trace, Patterns)
    Result := Ø.
    FOR each pattern P of Patterns
        IF Student_trace is included in P
            Result = Result ∪ {P}.
    IF Result := Ø AND length(Student_trace) ≥ 2
        Remove last action of Student_trace.
        Result:= RecognizePlan(Student_trace, Patterns).
    Return Result.
```

The plan recognition algorithm *RecognizePlan* is executed after each student action. It takes as input the sequence of actions performed by the student (*Student_trace*) for the current problem and a set of frequent action sequences (*Patterns*). When the plan recognition algorithm is called for the first time, the variable *Patterns* is initialized with the whole set of patterns found during the learning phase. The algorithm first iterates on the set of patterns *Patterns* to note all the patterns that include *Student_trace*. If no pattern is found, the algorithm removes the last action performed by the learner from *Student_trace* and searches again for matching patterns. This is repeated until the set of matching patterns is not empty or the length of *Student_trace* is smaller than 2. In our tests, removing user actions has improved the capability of the plan recognition algorithm to track learner's patterns significantly, as it makes the algorithm more flexible. The next time *RecognizePlan* is called, it will be called with the new *Student_trace* sequence and just the set of matching patterns found by the last execution of *RecognizePlan*, or the whole sets of patterns if no pattern matched.

After performing preliminary tests with the plan recognition algorithm, we noticed that in general, after more than 6 actions performed by a learner, it becomes hard for *RecognizePlan* to tell which pattern the learner is following. For that reason, we made improvements to how the CanadarmTutor applies the sequential pattern mining algorithm to extract a knowledge base. Originally, it mined frequent patterns from sequences of user actions for a whole problem-solving exercise. We modified our approach to add the notion of "problem states". In the context of CanadarmTutor, where an exercise consists of moving a robotic arm to attain a specific arm configuration, the 3D space was divided into 3D cubes, and the problem state at a given moment is defined as the set of 3D cubes containing the arm joints. An exercise is then viewed as going from a problem state P_1 to a problem state P_f. For each attempt at solving the exercise, CanadarmTutor logs (1) the sequence of problem states visited by the learner $A = P_1, P_2, \ldots P_n$ and (2) the list of actions performed by the learner to go from each problem state to the next visited problem state (P_1 to P_2, P_2 to P_3, … P_{n-1} to P_n). After many users perform the same exercise, CanadarmTutor extracts sequential patterns from (1) sequences of problem states visited, and from (2) sequences of actions performed for going from a problem state to another. To take advantage of the added notion of problem states, we modified *RecognizePlan* so that at every moment only the patterns performed in the current problem state are considered. To do so, every time the problem-state changes, *RecognizePlan* will be called with the set of patterns associated with the new problem state.

Moreover, at a coarser grain level tracking the problem states visited by the learners is also achieved by calling *RecognizePlan*. This allows connecting patterns for different problem states. We describe next the main tutoring services that a tutoring agent can provide based on the plan recognition algorithm.

1. Assessing the profile of a learner

First, a tutoring agent can assess the profile of the learner (expertise level, skills, etc.) by looking at the patterns applied. If for example 80 % of the time a learner applies patterns with value "intermediate" for dimension “expertise”, then CanadarmTutor can assert with confidence that the learner expertise level is "intermediate". In the same way, CanadarmTutor can diagnose mastered and missing/buggy skills for users who demonstrated a pattern by looking at the “skills” dimensions of patterns applied (e.g. “Skill_1” in Table 2). In CanadarmTutor, assessing the profile of a learner is done with a simple student model that is updated each time *RecognizePlan* finds that a student followed a pattern. The estimations contained in such a student model could be used in various ways by a tutoring system. The tutoring service described in the next subsection presents an example of how to use this information.

2. Guiding the learner

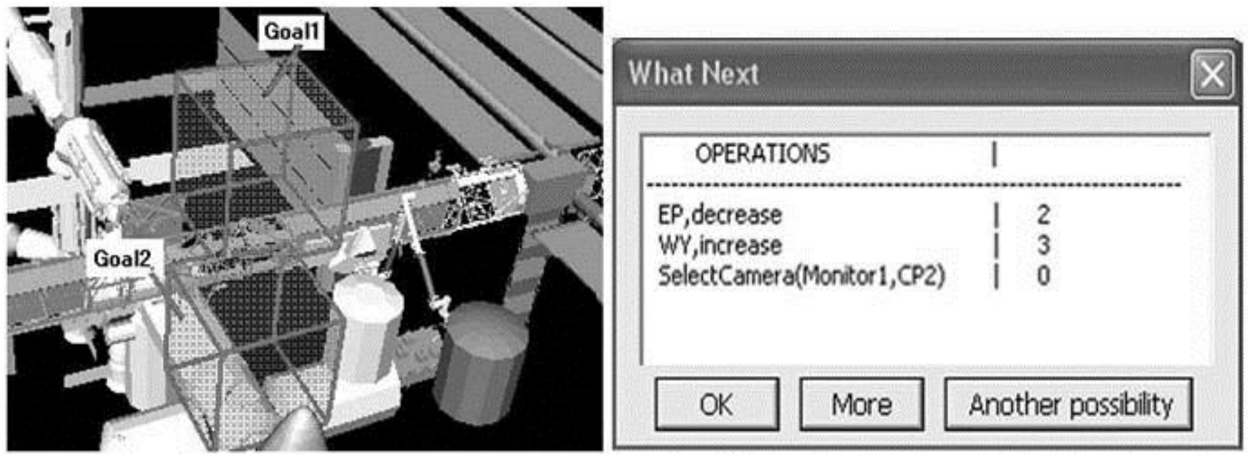
Second, a tutoring agent can guide the learner. This tutoring service consists in determining the possible actions from the current problem state and proposing one or more actions to the learner. In CanadarmTutor, this functionality is triggered when the student selects "What should I do next?" in the interface menu. CanadarmTutor then identifies the set of possible next actions according to the matching patterns found by *RecognizePlan*. The tutoring service then selects the action among this set that is associated with the pattern that has the highest relative support and that best matches the expertise level and skills of the learner. If the selected patterns contain skills that are not considered mastered by the learner, CanadarmTutor can use textual hints that are defined for each skill to explain the missing skills to the learner. In the case where no actions can be identified, CanadarmTutor can use the aforementioned path planner to generate solutions. In this current version, CanadarmTutor only interacts with the learner upon request. But it would be possible in future versions to program CanadarmTutor so that it can intervene if the learner is following an pattern that lead to failure or a pattern that is not appropriate for his or her expertise level. Different criteria for choosing a pattern could also be used for suggesting patterns to learners. Testing different tutorial strategies is part of our current work.

3. Letting learners explore different ways of solving problems

Finally, a tutoring service that has been implemented in CanadarmTutor is to let learners explore patterns to learn about possible ways to solve problems. Currently, the learners can explore a pattern with an interface that lists the patterns and their annotations, and provides sorting and filtering functions (for example to display only patterns leading to success). However, the learner could be assisted in this exploration by using an interactive dialog with the system which could prompt them on their goals and help them go through the patterns to achieve these goals. This tutoring service could be used when the tutoring system wants to prepare students before involving them in real problem-solving situations.

III. EVALUATING THE NEW VERSION OF CANADARMTUTOR

We conducted a preliminary experiment in CanadarmTutor with two exercises to qualitatively evaluate its capability to provide assistance. The two exercises consist of moving a load with the Canadarm2 robotic arm to one of the two cubes (figure 2.A). We asked 12 users to record plans for these exercises. The average length of plans was 20 actions. Each plan was annotated with “Solution state” and “Expertise level” information semi-automatically by CanadarmTutor: annotated solutions with skills information by hand. In our experiment, we included 10 skills that consist of familiarity with some important cameras and joints, and some more abstract skills of being good at evaluating distances, and at performing plans of manipulations requiring precise movements. From the set of annotated plans, CanadarmTutor extracted sequential patterns with the algorithm. In a subsequent work session, we asked the users to evaluate the tutoring services provided by the virtual agent. All the users agreed that the assistance provided was helpful. We also observed that CanadarmTutor often correctly inferred the estimated expertise level of learners, a capability not available in the previous version of CanadarmTutor, which relied solely on the path-planner.



OPERATIONS	
EP,decrease	2
WY,increase	3
SelectCamera(Monitor1,CP2)	0

Figure 2. (a) The two scenarios (b) A hint offered by CanadarmTutor

As an example of interaction with a learner, Figure 2.B illustrates a hint message given to a learner upon request during scenario 1. The guiding tutoring service selected the pattern that has the highest support value, matches the student actions for the current problem state, is marked "successful" and corresponds to the estimated expertise level and skills of the learner. The given hint is to decrease the rotation value of the joint "EP", increase the rotation value of joint "WY", and finally to select camera "CP2" on "Monitor1". The values on the right column indicate the parameter values for the action. In this context, the values "2" and "3" mean to rotate the joints 20 ^{o} and 30 ^{o}, respectively (1 unit equals 10º). By default, three steps are shown to the learner in the hint window depicted in figure 2.B. However, the learner can click on the "More" button to ask for more steps or click on the "another possibility" button to ask for an alternative.

It should be noted that although the sequential pattern algorithm was applied only one time in this experiment after recording each learner's plan, it would be possible to make CanadarmTutor apply it more often, in order to continuously update its knowledge base, while interacting with learners.

IV. RELATED WORK

This section compares our approach with related work.

1. Other automatic or semi-automatic approaches for learning domain knowledge in ITS

A few approaches have been described for learning domain knowledge from demonstrations in ITS [5, 6 17, 21, 22, 23].

Demonstr8 by Blessing [23] can infer simple production rules from pairs of inputs and outputs. However, rules can only have two arguments, and each rule action is defined as a single function from a predefined set of LISP functions such as adding two numbers. Moreover, demonstrators have to constantly indicate to Demonstr8 their goal and sub-goals during demonstrations so that it can link rules together. Blessing [23] applied Demonstr8 for the task of multi-column subtraction, and admitted that the approach would be hard to apply for more complex domains, or domains where not all reasoning is done within the user interface of the tutoring system (e.g. CanadarmTutor).

A second example is the Behavior Recorder [5], which allows an author to create a problem space by demonstrating the different correct or erroneous ways of solving a problem. The problem spaces produced by the Behavior Recorder are graphs where nodes are problem states and links are learner actions for passing from one state to another. The major limitation of the Behavior Recorder is that it does not perform any generalization. For a given problem, it simply integrates raw solutions in a graph, and the graph is problem-specific.

Jarvis & Nuzzo-Jones [6] designed a system that can extract general production rules from a set of demonstrations, given that authors provide a minimal set of background knowledge. This background knowledge comprises functions and conditions to create respectively the “IF” and the “THEN” parts of the learned rules. During demonstrations, demonstrators have to label the values that they manipulate in the user interface (for example, define an integer as a carry in a multi-column subtraction). Some limitations of the approach are that (1) the search for rules is an exhaustive search with a complexity that increases quickly with the number of functions and conditions to consider, (2) rules produced are sometimes over general, and (3) the approach was applied only in well defined domains such as fraction addition [6].

Another example is the virtual agent SimStudent [21], which can learn production rules from demonstrations. Similarly to the work of Jarvis & Nuzzo-Jones and Demonstr8, SimStudent requires authors to define operators and predicates to be used to create rules. During demonstrations, demonstrators have to indicate the name of each rule to be learned and the elements in the user interface to which the rule is applied. Rules that are learned are general and can be applied to several problems. SimStudent is designed for well-defined domains where a task can be described as production rules, and providing background knowledge or information during demonstrations can be a demanding task.

For building constraint-based tutors, a few systems have been described to learn constraints from demonstrations. One, named CAS [22] takes as input a domain ontology, and problems and solutions both annotated with concept instances from the ontology. CAS then extracts constraints directly from the restrictions that are defined in the ontology. CAS was successfully applied for the domains of fraction addition and entity-relation diagram modeling. But the success of CAS for a domain depends directly on whether it is possible to build an ontology that is appropriate and detailed enough, and this is not necessarily the case for all domains. In fact, ontology modeling can be difficult. Moreover, as previously mentioned, constraint-based tutors can validate solutions. But they cannot suggest next problem-solving steps to learners.

A last method that we mention is the work of Barnes & Stamper [17], which was applied for the well-defined domain of logic proofs. The approach of Barnes & Stamper consists of building a Markov decision process containing learner solutions for a problem. This is a graph acquired from a corpus of student problem-solving traces where each state represents a correct or erroneous state and each link is an action to go from one state to another. Then, given a state an optimal path can be calculated to reach a goal state according to desiderata such as popularity, lowest probability of errors, or shortest number of actions. An optimal path found in this way is then used to suggest to a learner the next actions to perform. This approach does not require providing any kind of domain knowledge or extra information during demonstrations. However, since all traces are stored completely in a graph, the approach seems limited to domains where the number of possibilities is not large. Moreover, their approach does not support more elaborated tutoring services such as estimating the profile of a learner by looking at the actions that the learner applies (e.g. expertise level), and hints are not suggested based on the estimated profile of a learner. We believe this to be a major limitation of their approach, as in many cases an ITS should not consider the “optimal solution” (as previously defined) as being the best solution for a learner. An ITS should instead select successful solutions that are adapted to a learner profile, to make the learner progress along the continuum from novice to expert.

In summary, in addition to some specific limitations, all systems mentioned in this section have one or more of the following limitations: they (1) require defining a body of background knowledge [6, 21, 22, 23], (2) have been demonstrated for well-defined domains [5, 6, 17, 21, 22, 23], (3) rely on the strong assumption that tasks can be modeled as production rules [6, 21, 23], (4) do not take into account learner profiles [5, 6, 17, 21, 22, 23], (5) learn knowledge that is problem specific [5, 17], or (6) require demonstrators to provide extra information during demonstrations such as their intentions or labels for elements of their solutions [6, 22].

Our approach is clearly different from these approaches as it does not possess any of these limitations except for generating task models that are problem-specific. We believe that this limitation is an acceptable trade-off, as in many domains a particular collection of exercises can be setup to be administered to many students. Also, it is important to note that our approach allows manual annotation of sequences if needed.

The approach of Barnes et al. shares some similarities with our approach as it needs to be applied for each problem, and authors are not required to provide any kind of domain knowledge. But the approach of Barnes et al. also differs from ours in several ways. The first important difference is that our approach extracts partial problem spaces from user solutions. Therefore our framework ignores parts of learner solutions that are not frequent. This strategy of extracting similarities in learners' solutions allows coping with domains such as the manipulation of Canadarm2 where the number of possibilities is very large and user solutions do not share many actions. In fact, our framework builds abstractions of learners' solutions, where the frequency threshold *minsup* controls what will be excluded from these abstractions. A second important difference is that the abstractions created by our framework are generalizations as they consist of subsequences appearing in several learner solutions. This property of problem spaces produced by our framework is very useful as it allows finding patterns that are common to several profiles of learners or contexts (for example, patterns common to expert users who succeed in solving a problem, or common to users possessing or lacking one or more skills). Conversely, the previously mentioned approaches do not take into account the profile of learners who recorded solutions.

2. Other Applications of Sequential Pattern Mining in E-Learning

Finally, we mention related work about the application of sequential pattern mining in the field of AIED. This work can be divided into two categories.

The first category is work on discovering patterns that will be interpreted by humans. For example, Kay et al. [20] mined sequential patterns from group work logs to discover patterns that would indicate good or bad group behavior in collaborative learning, so that human tutors could recognize problematic behavior in early stages of learning sessions. Similarly, Antunes [25] extracted sequential patterns to identify the deviations of individual student behaviors from a curriculum comprising several courses in a university setting. In this case, a sequence is the list of courses followed by a student. From these sequences, several sequential patterns were found. For instance, one pattern is that whenever some students failed a course on the 4th semester, they chose a particular economy course next [25]. Romeo et al. [26] proposed an information visualization tool that has a similar purpose. It allows visualizing routes that learners follow within a curriculum. Different parameters such as frequency define the routes that are displayed. The approach was applied in a web-based ITS to visualize navigation patterns.

A second category of work uses sequential pattern mining to help build systems for recommending learning activities to learners. For instance, Su et al. [27] described a system based on the idea of suggesting learning activities to learners based on what “similar” learners did. Their approach consists of (1) extracting sequential patterns from the learning patterns of each student, (2) clustering students who exhibited similar learning patterns into groups and (4) building a decision tree for each group from the patterns found for learning activity recommendation. A system with a similar aim was developed by Kristofic & Bielikova [7]. This system uses patterns found in learner navigation sequences to recommend concepts that a student should study next.

All work mentioned in this subsection is different from ours, as it does not attempt to learn domain knowledge.

V. CONCLUSION

In this paper, we have presented an approach for domain knowledge acquisition in ITS, and shown that it can be a plausible alternative to classic domain knowledge acquisition approaches, particularly for a procedural and ill-defined domain where classical approaches fail. For discovering domain knowledge, we proposed to use a sequential pattern mining algorithm that we designed for addressing the type of data recorded in tutoring systems such as CanadarmTutor. Since the proposed data mining framework and its inputs and outputs are fairly domain independent, it can be potentially applied to other ill-defined procedural domains where solutions to problems can be expressed as sequences of actions as defined previously. With the case study of CanadarmTutor, we described how the approach can support relevant tutoring services. We have evaluated the capability of the new version of CanadarmTutor to exploit the learned knowledge to provide tutoring services. Results showed an improvement over the previous version of CanadarmTutor in terms of tracking learners' behavior and providing hints. In future work, we will perform further experiments to measure empirically how the tutoring services influence the learning of students.

Because the problem spaces extracted with our approach are incomplete, we suggest using our approach jointly with other domain knowledge acquisition approaches when possible. In CanadarmTutor, we do so by making CanadarmTutor use the path-planner for providing hints when no patterns are available. Also, we are currently working on combining our data mining approach with the rule-based model that we implemented in another version of CanadarmTutor using the cognitive task analysis approach [9]. In particular, we are exploring the possibility of using skills from the rule-based model to automatically annotate recorded sequences used by the data mining approach presented in this paper.

We also plan to use association rule mining as in a previous version of our approach, to find associations between sequential patterns over a whole problem-solving exercise [18]. Mining association rules could improve the effectiveness of the tutoring services, as it is complementary to dividing the problem into problem states. For example, if a learner followed a pattern *p*, an association rule could indicate that the learner has a higher probability of applying another pattern *q* later during the exercise than some other pattern *r* that is available for the same problem state. Finally, we are interested in mining other types of temporal patterns in intelligent tutoring systems. We have recently published work [19] that instead of mining sequential patterns from the behavior of learners, mines frequent sequences from the behavior of a tutoring agent. The tutoring agent then reuses sequences of tutorial interventions that were successful with learners. This second research project is also based on the same algorithm described here.

Acknowledgment. Our thanks go to FQRNT and NSERC for their logistic and financial support. Moreover, the authors would like to thank Severin Vigot, Mikael Watrelot and Lionel Tchamfong for integrating the framework in CanadarmTutor, the other members of the GDAC/PLANIART teams who participated in the development of CanadarmTutor, and the anonymous reviewers who provided many comments for improving this paper.

REFERENCES

1. Aleven, V., McLaren, B. M., Sewall, J. and Koedinger, K. 2006. The Cognitive Tutor Authoring Tools (CTAT): Preliminary evaluation of efficiency gains. In *Proc. of the 8th Intern. Conf. on Intelligent Tutoring Systems*, Jhongli, Taiwan, June 26-30, 61-70.
2. Simon, H. A. 1978. Information-processing theory of human problem solving. In *Handbook of learning and cognitive processes: Vol. 5. Human information*, ed. W.K. Estes, 271-295. New Jersey: John Wiley & Sons, Inc.
3. Lynch, C., Ashley, K., Aleven, V. and Pinkwart, N. 2006. Defining Ill-Defined Domains; A literature survey. In *Proc. of the Intelligent Tutoring Systems for Ill-Defined Domains Workshop*, Jhongli, Taiwan, June 27, 1-10.
4. Mitrovic, A., Mayo, M., Suraweera, P. and Martin, B. 2001. Contraint-based tutors: a success story. In *Proc. of the 14th Industrial & Engineering Application of Artificial Intelligence & Expert Systems*, Budapest, Hungary, June 4-7, 931-940.
5. McLaren, B., Koedinger, K.R., Schneider, M., Harrer, A. and Bollen, L. 2004. Bootstrapping Novice Data: Semi-Automated Tutor Authoring Using Student Log Files. In *Proc. of the Workshop on Analyzing Student-Tutor Logs to Improve Educational Outcomes*, Maceiò, Alagoas, Brazil, August 30, 1-13.
6. Jarvis, M., Nuzzo-Jones, G. and Heffernan, N.T. 2004. Applying Machine Learning Techniques to Rule Generation in Intelligent Tutoring Systems. *Proc. of 7th Intern. Conf. on Intelligent Tutoring Systems*, Maceiò, Brazil, August 30 - September 3, 541-553.
7. Kriatofic, A and Bielikova. 2005. Improving adaptation in web-based educational hypermedia by means of knowledge discovery. In *Proc. 16th ACM Conference on Hypertext and Hypermedia*, Bratislava, Slovakia, September 6-10, 184-192.
8. Kabanza, F., Nkambou, R. and Belghith, K. 2005. Path-planning for Autonomous Training on Robot Manipulators in Space, In *Proc. of Intern. Joint Conf. on Artificial Intelligence 2005*, Edinburgh, Scotland, July 30 - August 5, 35-38.
9. Fournier-Viger, P., Nkambou, R. and Mayers, A. 2008. Evaluating Spatial Representations

and Skills in a Simulator-Based Tutoring System. *IEEE Transactions on Learning Technologies*, 1(1): 63-74.

10. Hirate, Y., Yamana, H. 2006. Generalized Sequential Pattern Mining with Item Intervals, *Journal of Computers*, 1(3): 51-60.
11. Pinto, H. et al. 2001. Multi-Dimensional Sequential Pattern Mining, In *Proc. of the 10th Int. Conf. Information and Knowledge Management*, Atlanta, Georgia, November 5-10, 81-88.
12. Han, J. and Kamber, M., 2000. *Data mining: concepts and techniques*, San Francisco: Morgan Kaufmann Publisher.
13. Agrawal, R. and Srikant, R. 1995. Mining Sequential Patterns. In *Proc. Int. Conf. on Data Engineering*, Taipei, Taiwan, March 6-10, 3-14.
14. Wang, J., Han, J. and Li, C. 2007. Frequent Closed Sequence Mining without Candidate Maintenance, *IEEE Trans. on Knowledge and Data Engineering*, 19(8):1042-1056.
15. Fournier-Viger, P., Nkambou, R. and Mephu Nguifo, E. 2008. A Knowledge Discovery Framework for Learning Task Models from User Interactions in Intelligent Tutoring Systems. In *Proc. 6th Mexican International Conference on Artificial Intelligence,* Atizapán de Zaragoza, Mexico, October 27-31, 765-778.
16. Songram, P, Boonjing, V. and Intakosum, S. 2006. Closed Multidimensional Sequential Pattern Mining, In *Proc. 3rd Int. Conf. Information Technology: New Generations*, Las Vegas, USA, April 10-12, 512-517.
17. Barnes, T, Stamper, J. 2008. Toward Automatic Hint Generation for Logic Proof Tutoring Using Historical Student Data, In *Proc. 9th Intern. Conf. on Intelligent Tutoring Systems,* Montreal, Canada, June 23-27, 373-382.
18. Nkambou, R., Mephu Nguifo, E. and Fournier-Viger, P. 2008. Using Knowledge Discovery Techniques to Support Tutoring in an Ill-Defined Domain. In *Proc. 9th Intern. Conf. on Intelligent Tutoring Systems*, Montreal, Canada, June 23-27, 395-405.
19. Faghihi, U., Fournier-Viger, P. and Nkambou, R. 2009. How Emotional Mechanism Helps Episodic Learning in a Cognitive Agent. In *Proc. IEEE Symposium on Intelligent Agents,* Nashville, USA, March 30 - April 2, 2009, 23-30.
20. Perera, D., Kay, J., Koprinska, I., Yacef and K., Zaiane, O. 2008. Clustering and Sequential Pattern Mining of Online Collaborative Learning Data, *IEEE Transactions on Knowledge and Data Engineering*, 21(6), 759-772.

21. Matsuda, N., Cohen, W., Sewall, J., Lacerda, G. and Koedinger, K. 2007. Performance with SimStudent: Learning Cognitive Skills from Observation. In *Proc. of Artificial Intelligence in Education 2007*, July 9-13, 2007, Los Angeles, USA, 467-478.
22. Suraweera, P., Mitrovic, A. and Martin, B. 2007. Constraint Authoring System: An Empirical Evaluation. In *Proc. of Artificial Intelligence in Education 2007*, July 9-13, 2007, Los Angeles, USA, 467-478.
23. Blessing, S. B. 1997. A programming by demonstration authoring tool for model-tracing tutors. *International Journal of Artificial Intelligence in Education*, 8, 233-261.
24. Yuan, D., Lee, K., Cheng, H., Krishna, G., Li, Z., Ma., X., Zhou, Y. and Han, J. 2008. CISpan: Comprehensive Incremental Mining Algorithms of Closed Sequential Patterns for Multi-Versional Software Mining. In *Proc. of the 8th SIAM Intern. Conf. on Data Mining*, Atlanta, USA, April 24-26, 84-95.
25. Antunes, C. 2008. Acquiring Background Knowledge for Intelligent Tutoring Systems. In *Proc. of the 2nd International Conference on Educational Data Mining*, Montreal, Canada, June 20-21, 18-27.
26. Romero, C. and Gutiarrez, S., Freire, M. and Ventura, S. 2008. Mining and Visualizing Visited Trails in Web-Based Educational Systems. In *Proc. of the 2nd International Conference on Educational Data Mining*, Montreal, Canada, June 20-21, 182-186.
27. Su, J.-M., Tseng, S.-S., Wang. W., Weng, J.-F., Yang, J. T. D. and Tsai, W.-N. 2006. Learning Portfolio Analysis and Mining for SCORM Compliant Environment. *Educational Technology & Society*, 9(1): 262-275.