Problem-Solving Knowledge Mining from Users' Actions in an Intelligent Tutoring System

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

[1] Université du Québec à Montréal (Canada)
[2] CRIL-CNRS, IUT de Lens (France)
nkambou.roger@uqam.ca, mephu@cril.univ-artois.fr

Abstract. In an intelligent tutoring system (ITS), the domain expert should provide relevant domain knowledge to the tutor so that it will be able to guide the learner during problem solving. However, in several domains, this knowledge is not predetermined and should be captured or learned from expert users as well as intermediate and novice users. Our hypothesis is that, knowledge discovery (KD) techniques can help to build this domain intelligence in ITS. This paper proposes a framework to capture problem-solving knowledge using a promising approach of data and knowledge discovery based on a combination of sequential pattern mining and association rules discovery techniques. The framework has been implemented and is used to discover new meta knowledge and rules in a given domain which then extend domain knowledge and serve as problem space allowing the intelligent tutoring system to guide learners in problem-solving situations. Preliminary experiments have been conducted using the framework as an alternative to a path-planning problem solver in CanadarmTutor.

1 Introduction

In an intelligent tutoring system (ITS), the domain expert should provide relevant domain knowledge to the tutor so that it will be able to guide the learner during problem solving. However, in several domains, this knowledge is not predetermined and should be captured or learned from expert users as well as intermediate and novice users. Our hypothesis is that, knowledge discovery (KD) techniques can help to build this domain intelligence in ITS.

This paper proposes an approach to support new domain knowledge discovery in domain where it is difficult to set up a clear problem space or task models. In such a domain, we need to capture new procedures (correct or incorrect), new problem spaces and new problem-solving strategies from users' actions. Cognitive task analysis that aims at producing effective problem space or task model (to support model and knowledge tracing, coaching, errors detection and plan recognition) is a very time consuming process [8]. How can we build this complex structure by learning from users' interactions with an ITS ?

The approach presented in this paper is based on a combination of sequential pattern recognition and association rule discovery. We show how the proposed approach

Z. Kobti and D. Wu (Eds.): Canadian AI 2007, LNAI 4509, pp. 393 – 404, 2007.

is used to discover new knowledge in a given domain, which then extends domain knowledge and serves as a problem space allowing the intelligent tutor to track learners' actions and give relevant hints when needed.

The paper is organized as follows. First, we will present the context of this research work by stating the need of KD to enhance tutoring agent knowledge. Then we will describe the tutoring context and show how data can be transformed for KD. We will also briefly describe algorithms that take this context as input, to extract significant sequences of patterns and relationships between them, which will constitute relevant partial or complete plans reusable in a given problem-solving activity to track student cognitive behavior. Finally, we show how this knowledge is used by a tutoring system (CanadarmTutor) aimed at training astronauts during procedural tasks on the ISS (International Space Station) using a robot manipulator called CanadarmII.

2 Problem Statement and Related Works

Educational data mining is becoming a very important area in the Artificial Intelligence in Education community [1]. Several techniques are used to extract relevant data, information or knowledge mainly from databases and log files of learning sessions. However, most work focuses on learner or group classification, clustering or sorting [2, 3]. Very few studies address procedural knowledge learning and none attempts to find and learn relations between actions, sequences of actions, and patterns among them, which may provide useful information regarding the procedure.

Kay *et al*. [4] describe student group interaction data mining that seeks to identify significant sequences of activity. Their goal is to flag interaction sequences which indicate problems and successes, so that tutors can help students recognize problematic situations in the early stages of the learning sessions. Their goal is not related to learning procedures nor does it aim to find links between significant interaction sequences or patterns.

Very little AIED research investigated ITS automatic procedural knowledge learning [5, 6, 7]. Yet, such a capability could facilitate the development of problem spaces (task model, procedural knowledge, etc.) and reduce the need for domain experts. For example, [6] attempted to induce simple production rules using a single example and the analogy mechanism in ACT-R; [7] looks up a set of marked examples, trying to generalize them and generate production rules. None of them have explored sequential patterns and rules discovery, which can help determine problem-solving steps and rules.

Creating cognitive tutors usually rests on the implicit assumption that one should predefine a task model describing correct and incorrect solution paths. Similarly, CTAT (Cognitive Tutor Authoring Tool) [8] offers a set of tools that allows ITS designers to specify the behavioural graph (BG) of a task, presenting correct and buggy paths. BGs (sometimes transformed into production rules) are used to track student actions. The behaviour recorder can automate the translation of user actions into a BG. This concept was improved by the BND (Bootstrapping Novice Data) approach proposed by McLaren *et al.* [5]. BND records the actions of many students in a log file which is then used to create a common BG that can be improved by designers. Instead of having authors build problem-solving expertise from scratch, tap into only their own experience or incorporate student data manually as in traditional ITS

development, this tool semi-automatically leverages the empirical data of actual problem-solving activities. However, the BND approach is devoid of data mining and learning, reducing the approach to a simple way of storing or integrating raw user solutions into a structure, as in [6] and [7]. In fact, student data are incorporated into the BG regardless of possible links between problem steps or actions. This is very limiting because the system does not try to extract useful knowledge from those solutions, which could enrich the problem space.

Contrary to these approaches, we are proposing a solution to create a more general, flexible and powerful (albeit sometimes partial) BG-like structure by inferring association rules between actions or action sequences. In fact, domain users (both expert and novice) can provide primitive action sequences required to achieve typical tasks in the application domain. These sequences (good and buggy) may then be used to teach procedural knowledge associated with the task, thereby continually enhancing the system's intelligence.

3 Modelling Procedural Knowledge in CanadarmTutor

One of the main goals of an intelligent tutoring system is to actively provide relevant feedback to the student in problem-solving situations [9]. This kind of support becomes very difficult when an explicit representation of the training task is not available. This is the case in the ISS environment where the problem space associated with a given task consists of an infinite number of paths. Moreover, there is a need to generate new tasks on the fly without any cognitive structure. *Roman Tutor* brings a solution to these issues by using FADPRM, a path planner, as main resource for the tutoring feedback.

FADPRM [10] is a flexible and efficient approach for robot path planning in constrained environments. In addition to the obstacles that the robot must avoid, our approach holds account of desired and non-desired (or dangerous) zones. This will make it possible to take into account the disposition of cameras on the station. Thus, our planner will try to bring the robot in zones offering the best possible visibility of the progression while trying to avoid zones with reduced visibility.

FADPRM allows us to put in the environment different zones with arbitrary geometrical forms. A degree of desirability *dd*, a real in [0 1] is assigned to each zone. The *dd* of a desired zone is then near 1, and the more it approaches 1, the more the zone is desired; the same for a non-desired zone where the *dd* is in [0 0.5]. On the international Space Station, the number, the form and the placement of zones reflect the disposition of cameras on the station. A zone covering the field of vision of a camera will be assigned a high *dd* (near 1) and will take a shape which resembles that of a cone; whereas a zone that is not visible by any camera from those present on the station will be considered as an non-desired zone with a *dd* near to 0 and will take an arbitrary polygonal shape.

The ISS environment is then preprocessed into a roadmap of collision-free robot motions in regions with highest desirability degree. More precisely, the roadmap is a graph such that every node *n* is labeled with its corresponding robot configuration *n.q* and its degree of desirability *n.dd*, which is the average of *dd*s of zones overlapping with *n.q*. An edge *(n,n')* connecting two nodes is also assigned a *dd* equal to the

average of *dd* of configurations in the path-segment *(n.q,n'.q)*. The *dd* of a path (i.e., a sequence of nodes) is an average of *dd* of its edges.

Following probabilistic roadmap methods (PRM) [11], we build the roadmap by picking robot configurations probabilistically, with a probability that is biased by the density of obstacles. A path is then a sequence of collision free edges in the roadmap, connecting the initial and goal configurations.

Following the Anytime Dynamic A* (AD*) approach [12], to get new paths when the conditions defining safe zones have dynamically changed, we can quickly re-plan by exploiting the previous roadmap. Moreover, paths are computed through incremental improvements so that the planner can be called at anytime to provide a collision-free path and the more time it is given, the better the path optimizes moves through desirable zones. Therefore, our planner is a combination of the traditional PRM approach [11] and AD* [12] and it is flexible in that it takes into account zones with degrees of desirability. This explains why we called it Flexible Anytime Dynamic PRM (FADPRM).

We implemented FADPRM as an extension to the Motion Planning Kit (MPK)[11] by changing the definition of PRM to include zones with degrees of desirability and changing the algorithm for searching the PRM with FADPRM. The calculation of a configuration's *dd* and a path's *dd* is a straightforward extension of collision checking for configurations and path segments.

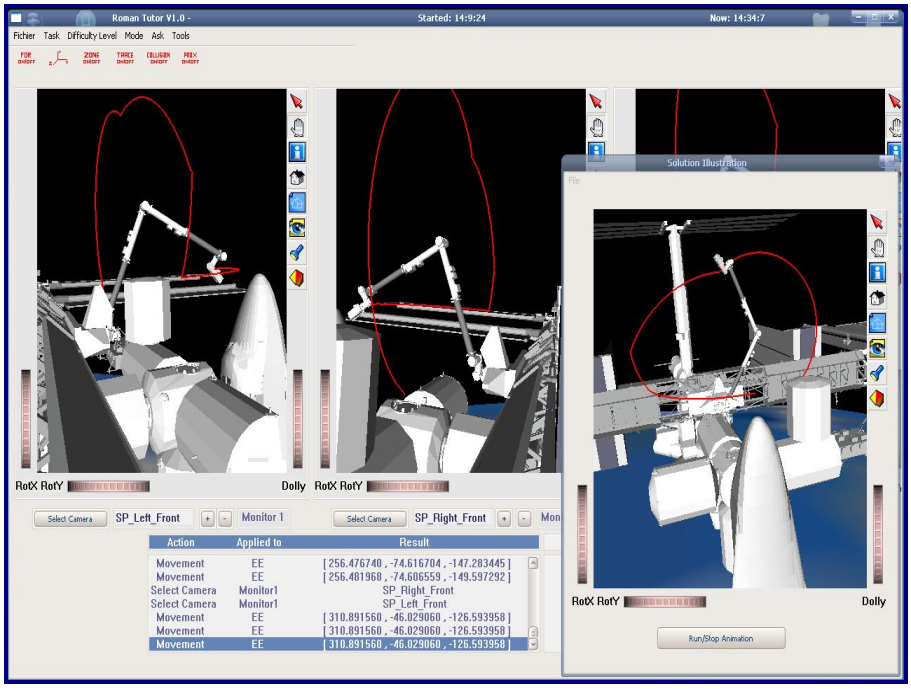


Fig. 1. Path planning and task demonstration using FADPRM

FADPRM computes a collision and soft constraints (camera views, etc.) free path that serves as expert solution for the tutor. FADPRM is also called when the tutor needs to validate student actions, to demonstrate a given task or to suggest a solution path. However, the resulting path is sometimes too much complex to be followed by a human user and far from the procedure that a human would execute in a real-world situation. In figure 1, FADPRM generates a path that the user should follow and can demonstrate it in another window.

A good tutor in procedural tasks should fulfill at least the following important properties: 1) guide the user through expert users' solution; 2) and recognize the student profile (novice, intermediate or expert) to offer tailored help.

Tutoring services based on FADPRM fails to satisfy these properties. We believe that a way to solve this problem is to base coaching on knowledge that comes from users themselves. In this way, the system can 1) capture data from the system usage by users of all possible profiles and 2) learn rules and constraints that can contribute to a knowledge base to support adapted tutoring services. Our hypothesis is that, tutoring services based on such a knowledge base will guarantee high quality assistance to the learners.

The next sections of this paper present a way to implement this solution.

4 Problem-Solving Data Representation

In cognitive tutors, problem-solving knowledge is represented as procedures each corresponding to a possible path to a successful or unsuccessful solution to the problem. A procedure (or a plan) is a sequence of atomic and non-atomic actions. Non-atomic actions are actions containing at least two atomic actions. Actions are events that occur at a given time. Table 1 shows an example dataset of 8 successful plans where each entry corresponds to a plan's events. From this dataset, it is possible to easily compute frequent sequences of actions using a minimal support (*minsup*) defined by the user. A sequence is said to be frequent if it occurs more than *minsup* times.

Table 1. A data set of 8 successful plans

PlanID	**Sequences of actions**
P1	1 2 25 46 48 {9 10 11 31}
P2	1 25 46 54 79 {10 11 25 27}
P3	1 2 3 {9 10 11 31} 48
P4	2 3 25 46 11 {14 15 16 48} 74
P5	2 25 46 47 48 49 {8 9 10}
P6	1 2 3 4 5 6 7
P7	25 26 27 28 30 {32 33 34 35 36}
P8	46 54 76 {10 27} {48 74}

From the frequent sequences set, the next step consists in finding rules that connect them using a simple algorithm that considers sub-sequences of each sequence and derives a relationship between them given their number of occurrences in the dataset.

5 The Proposed Framework

The system that we propose goes through different stages or processes to learn rules. At each stage, we adapt and integrate specific algorithms. The main scheme is as follow (process in bold):

Log files containing users plans → ***Automatic coding of data*** → *Formated Binary-File* → ***Sequential Patterns Finding (PrefixSpan)*** → *Frequent patterns* → ***Building of the Meta-Context*** → *Meta-Context* → ***Association Rules Finding (IGB)*** → *New procedural task knowledge (PTK)* → ***Integration within the Tutoring System.***

5.1 Sequential Patterns Mining

The problem of mining sequential patterns was originally proposed by Agrawal and Srikant [11]. Let $I = \{i_1, i_2,\ldots, i_n\}$ be a set of items or actions. We call a subset $X \subseteq I$ an *itemset or an actionset* and we call $|X|$ the *size* of X. A *sequence* $s = (s_1, s_2, \ldots, s_m)$ is an ordered list of actionsets, where $s_i \subseteq I$, $i \in \{1,\ldots,m\}$. The size, m, of a sequence is the number of actionsets in the sequence, i.e. $|s|$. The length of a sequence $s = (s_1, s_2, \ldots, s_m)$ is defined as : $l = \sum |s_i|$, *for i =1 to m.*

A sequence with length l is called an l-sequence. A sequence $s_a = (a_1,a_2,\ldots, a_n)$ is contained in another sequence $s_b = (b_1, b_2,\ldots, b_m)$ if there exists integers $1 \leq i_1 < i_2 < \ldots < i_n \leq m$ such that $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, . . . , $a_n \subseteq b_{in}$. If sequence s_a is contained in sequence s_b, then we call s_a a *subsequence* of s_b and s_b a *supersequence* of s_a.

The *relative support* is defined as the percentage of sequences $s \in D$ that contains s_a. The support of s_a in D is denoted by $supD(s_a)$.

Given a support threshold *minsup*, a sequence s_a is called a *frequent sequential pattern on D* if $supD(s_a) \geq minsup$. The problem of mining sequential patterns is to find all frequent sequential patterns for a database D, given a support threshold *minsup*.

Table 1 shows the dataset consisting of tuples in its sequence representation. Consider the sequence of plan 2; the size of this sequence is 6, and the length of this sequence is 9. Suppose we want to find the support of the sequence $s_a = (1\ \{9\ 31\})$. From Table 1, we know that s_a is a subsequence of the sequences for plan 1 and plan 3 but is not a subsequence of the sequence for plan 2. Hence, the support of s_a is 2 (out of a possible 8), or 0.25. If the user-defined minimum support value is less than 0.25, then s_a is deemed frequent.

A subsequence or pattern, P, is *closed* if there exists no superset of P with the same support in the database. A closed pattern induces an equivalence class of pattern sharing the same closure. The *minimal generators* and the unique *closed pattern* of an equivalence class of actionsets share a common set of plans. The minimal generators are the minimal ones among the equivalent actionsets, while the closed pattern is the maximum one. The closed pattern is unique.

5.2 Finding Frequent Sequential Patterns Using PrefixSpan

Many algorithms have been proposed to efficiently mine sequential patterns or other time-related data [13, 14, 15, 16]. We choose here the PrefixSpan approach [15] as it is

one of the most promising ones for mining large sequence databases having numerous patterns and/or long patterns, and also because it can be extended to mine sequential patterns with user-specified constraints. PrefixSpan is a projection-based, sequential pattern-growth approach that recursively projects a sequence database into a set of smaller projected databases. Sequential patterns are grown in each projected database by exploring only locally frequent fragments. Table 2 shows examples of sequential patterns extracted by PrefixSpan from data in table 1 using a minimum support equals 25%.

Links between sequential patterns can lead to the tutor goal. Thus PrefixSPan can find long frequent sequence patterns, and those patterns will be linked by generating associations among them. In our case, we are interested by minimal and non redundant association rules, also called generic bases.

Table 2. Examples of sequential patterns extracted by PrefixSpan with their associated labels

Sequential patterns	Sequence patterns' labels
1 25 46 48	S1
1 25 46 {10 11}	S2
1 {9 10 31}	S6
1 {9 11 31}	S7
1 {9 10 11 31}	S8
1 46 {10 11}	S13

Among previous studies on mining of generic bases, we choose IGB [17] as it efficiently extracts more compact generic bases without information loss, i.e. all association rules can be derived from these generic bases with their exact support.

5.3 Extracting Generic Rules Between Patterns Using IGB

IGB [17] is a new informative generic basis. It has a valid and complete axiomatic system allowing the derivation of all the association rules. Rules of *IGB* are correlations between minimal premise and maximal conclusion (in term of items number). Indeed, it was shown that this kind of rules is the most general (*i.e.,* conveying the maximum of information). The premise of some generic rules of *IGB* can be empty such that they are two types of generic rules: (1). *factual rules* having an empty premise; and (2). *implicative rules* having a non empty premise.

IGB basis is generated by a dedicated algorithm which takes as input the meta-context of initial plans, and two thresholds which are the minimum support, *minsup* (already defined in PrefixSpan), and the minimum confidence, *minconf*. The meta-context of initial plans (see example in Table 3) is the set of plans redefined with the frequent sequential patterns obtained with PrefixSpan.

IGB algorithm checks for each non empty closed pattern, P, if its support is greater or equal to *minconf*. If it is the case, then the generic rule $\emptyset \rightarrow P$ is added to *IGB base*. Else, it iterates on all frequent closed actionsets P_0 subsumed by P. For those having support at least equal to *support(P)/minconf*, the algorithm iterates on the list of minimal generators associated to P_0. During this iteration, we look for the smallest minimal generator g_s, such that there does not exist a generator g_0 subsumed by g_s which is already inserted in the list L of smallest premises. Then, IGB algorithm iterates on all elements of the list L in order to generate rules of *IGB* which have the following form: $g_s \rightarrow (P - g_s)$.

Table 3. Part of the crisp meta-context of frequent sequences built from dataset in table1

PlanID	Frequent sequential patterns
P1	S1 S2 S4 S5 S6 S7 S8 S9 S10 S95 S97 S98 S113 S116 S118
P2	S1 S5 S6 S7 S9 S98
P3	S1 S2 S3 S4 S5 S6 S8 S10 S95 S97 S98 S113 S116 S118
P4	S2 S3 S6 S7 S9 S10
P5	S2 S4 S5 S7 S9 S10 S95
P6	S1 S2 S3
P7	S7
P8	S5 S9 S10

By dividing the sub-sequence occurrence by the plans' occurrence, we obtain the relative support associated to the sub-sequence. Let consider a *minsup* of 2 (25%), meaning that a valid sequence should occur in at least 2 input-plans, we can obtain the meta-context which part is shown in table 3. Each sub-sequence can appear in a plan with a certain confidence which is its relative support (in table 3, we consider a crisp context where dichotomic values (0 or 1) are assigned when a subsequence appears or not in a plan). Using this meta-context as input, IGB computes a set of generic meta-rules, part of which is shown in table 4. These meta-rules combined with frequent sequential patterns will constitute the knowledge that will be used by the tutor to guide students and domain users to explore and learn the procedural task.

Table 4. Examples of generic meta-rules extracted by IGB

Meta-rules	Support	Confidence	Expanded meta-rules
S10 ===> S9	4	0.8	…
S9 ===> S7	4	0.8	1 {10 31} ===> 1 {9 11 31}
S9 ===> S5	4	0.8	…
S5 ===> S10	4	0.8	…

6 How the Learned Knowledge Base Is Used for Tutoring Services?

As said before, tutoring systems should provide useful tutoring services to assist the learner. These services include coaching, assisting, guiding, helping or tracking the student during problem-solving situations. To offer these services, a tutoring system needs some knowledge related to the context. The knowledge base namely procedural task knowledge (PTK) obtained from the previous knowledge mining process serves to that end. The next paragraphs present some examples of services that can be supported.

Assisting the User to Explore Possible Solutions of a Given Problem. Domain expert users can explore, validate or annotate the PTK. The validation can consist in removing all meta-rules with a low confidence, meaning that those rules can not significantly contribute to help the student. Annotation consists in connecting some useful information to meta-rules lattice depicting semantic steps of the problem as well as

hints or skills associated to a given step. A meta-rule lattice annotated in this way is equivalent to [8]'s BN or Sherlock's effective problem space (EPS), except that EPS and BN are explicitly built from scratch by domain experts.

For student users, exploring PTK will help them learn about possible ways of solving problem. They can be assisted in this exploration using an interactive dialog with the system which can prompt them on their goals and helps them go through the rules in order to achieve these goals. This kind of service can be used when the tutoring system wants to prepare students before involving them in real problem-solving situation.

Tracking the Learner Actions to Recognize the Plan S/He is Following. Plan recognition is very important in tutoring systems. PTK is a great resource to this process. Each student's action can be tracked by searching the space defined by meta-rules lattice so that we can see the path being followed. For this service, partitioning the space in terms of equivalent classes corresponding to maximal sequences as proposed in [14] can make plan recognition (and exploration) easier. In fact, when it is recognized the current plan is in a class, all other classes are pruned so that the exploration will continue only in a single class.

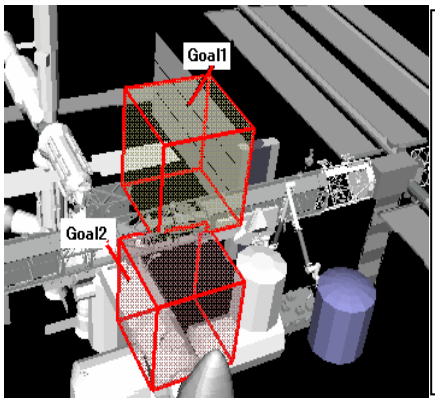
Guiding Learners. When solving a problem, an ITS should be able to help the student. A classic situation is when the student asks the tutor what to do next from the actual state. PTK can help the tutoring agent to produce the next most probable actions that the student should execute and prompt him on that, taking into account uncertainty related to rules' confidence. An example of a dialog can be as follow:

....
Student : What should I do now ?
Tutor : Oh! I don't quite know but I think you should try action B.
Student : Why ?
Tutor : Well, in 75% of the cases, people who tried that action achieved the final goal !
Student : Ok! Are there any other possibilities?
...

7 Results and Discussion

We have set up two scenarios consisting each in moving the load to one of the two cubes (figure 2a). A total of 15 users (a mix of novices, intermediates and experts) have been invited to execute these scenarios using the CanadarmII robot simulator. A total of 119 primitive actions have been identified. Figure 2b shows part of an example log file from a user's execution of the first scenario. We obtain a database with 45 entries each corresponding to a given usage of the system. A value indicating the failure or success of the plan has been added at the end of each entry.

The framework presented in section 5 was applied. A unique number was assigned to each action. After coding each entry of the traces database using PrefixSpan data representation, we obtained a binary file containing plans' data for the two scenarios. This file was sent as input to the rest of the process.



…
EnterCorridor(CouloirZoneCameraInitiale)
{SelectCamera(Monitor1,CP8),SelectCamera(Monitor2,CP10),
SelectCamera(Monitor3,CP9)}
SelectJoint(WE)
bigMove(WE,decrease)
LeaveCorridor(CouloirZoneCameraInitiale)
smallMove(WE,decrease)
SelectJoint(SP)
bigMove(SP,decrease)
SelectCamera(Monitor1,CP2)
smallMove(SP,decrease)
EnterCorridor(CouloirZoneCameraInitialeBasDroitePresqueTotale)
mediumMove(SP,decrease)
LeaveCorridor(CouloirZoneCameraInitialeBasDroitePresqueTotale)
smallMove(SP,decrease)
…

Fig. 2. (a) Environment setup for the two experimental scenarios (b) An entry of the plans' database

The Results. After executing PrefixSpan, the first stage of the experiment consisting in finding sequential patterns from the input data, we obtained a total of 76 significant patterns (with a *support* greater that .5). At the second stage, we created a binary context where each row represents a plan data and each column stands for a set of patterns. The goal at this stage was to mine association rules between sequential patterns. Using IGB approach, we obtained a PTK of 37 significant meta-rules. These rules were then coded and integrated in a new version of CanadarmTutor that uses this knowledge base to support tutoring services. An empirical test with this version has been conducted with the same users of the system's version relying on the FADPRM. They have been asked to execute the two scenarios. We found that, the system behavior in terms of guiding the user was significantly improved compared to the behavior observed in the version relying on the path planner. The system can now recommend good and easy-to-follow actions sequences. The system can also recognize users' plans and anticipate failures or successes, thus proactively help them at the earlier stage. Using the learned knowledge base, the system can also infer user profiles by detecting (analyzing) the path they follow. The PTK produced by our framework is sometimes too large and contains non useful rules. However, this is not harmful for the tutor behavior but it may slow the performance as the system need to go through this huge knowledge base each time the user executes an action. We are now working to improve the quality of the PTK. We are also looking for a way of managing unsuccessful plans data. In fact in the actual version of the implemented framework, we do not consider plans that fail. We should find a way to integrate these data. We believe that this integration may lead to a more powerful behavior of the tutoring agent in the sense that it can easily identify sequence patterns that lead to failure or success, hence better guiding the learner.

8 Conclusion

In this paper, we proposed a KD framework that combines sequential pattern mining and association rules discovery techniques. We showed how the proposed framework can contribute to enhance an intelligent tutoring system's knowledge in procedural

domain. We used the framework to build a meta-knowledge base of plans from users' traces in CanadarmTutor. The resulting knowledge base enables CanadarmTutor to better help the learner. For future works, we plan to find some ways of filtering the resulting meta-rules and integrating unsuccessful paths. We will also carry out further tests to clearly measure the benefit of the approach in terms of tutoring assistance services.

Acknowledgment

We would like to thank Kalhed Belghith and Froduald Kabanza for their contributions to the FADPRM design and implementation.

References

1. Heiner, C., Baker, R. and Yacef, K. (2006). *Proceedings of the Educational Data Mining Workshop*. ITS'2006.
2. Graf, S. and Bekele, R. (2006). "Forming Heterogeneous Groups for Intelligent Collaborative Learning Systems with Ant Colony Optimization". *Intelligent Tutoring Systems* 2006: 217-226. Springer-Verlag.
3. Amershi, S. and Conati, C. (2006). "Automatic Recognition of Learner Groups in Exploratory Learning Environments". *Intelligent Tutoring Systems* 2006: 463-472. Springer-Verlag.
4. Kay, J., Maisonneuve, N., Yacef, K, Zaïane, O. (2006). "Mining Patterns of Events in Students' Teamwork Data". *Proceedings of the Workshop on Educational Data Mining Workshop*. ITS'2006, pp. 45-52.
5. McLaren, B. M. et al. (2004). "Bootstrapping Novice Data: Semi-Automated Tutor Authoring Using Student Log Files". *Proceedings of the Workshop on Analyzing Student-Tutor Logs*. ITS'2004.
6. Blessing, S.B. (2003). "A Programming by Demonstration Authoring Tool for Model-Tracing Tutors". In, *Authoring Tools for Advanced Technology Learning Environments: Toward Cost-Effective Adaptive, Interactive and Intelligent Educational Software*, pp. 93-119. Kluwer Academic Publishers
7. Jarivs, M., Nuzzo-Jones, G. & Heffernan. N. T. (2004). "Applying Machine Learning Techniques to Rule Generation in Intelligent Tutoring Systems". *Intelligent Tutoring Systems* 2006: 541-553. Springer.
8. Aleven, V., McLaren, B. M., Sewall, J., & Koedinger, K. (2006). "The Cognitive Tutor Authoring Tools (CTAT): Preliminary evaluation of efficiency gains". *Intelligent Tutoring Systems* 2006: 61-70. Springer.
9. VanLehn, K. (2003). "The advantages of Explicity Representing Problem Spaces". *User Modeling*, Springer Verlag LNAI 2702:3.
10. Kabanza, F., Nkambou, R. and Belghith, K. (2005). "Path-Planning for Autonomous Training on Robot Manipulators in Space". *IJCAI 2005*: 1729-1731.
11. Sanchez, G., Latombe, J.C. (2001). "A single-query bi-directional probabilistic roadmap planner with lazy collision checking". *Int. Symposium on Robotics Research (ISRR'01).* Springer Tracts in Advanced Robotics, Springer, 403-417.

12. Likhachev, M., Ferguson, D., Stentz, A., Thrun, S. (2005). “Anytime Dynamic A*: An Anytime Replanning Algorithme”. *Proceedings of International Conference on Automated Planning and Scheduling*.
13. R. Agrawal and R. Srikant. Mining Sequential Patterns. *Proceedings of the 1995 Int. Conference on Data Engineering*, pp. 3-14, 1995
14. Zaki, M.J. (2001). SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning Journal*, 42(1-2): 31-60.
15. J. Pei, J. Han et al (2004). Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. *IEEE Transaction on Knowledge and Data Engineering*, 16(10), oct. 2004.
16. F. Masseglia, F. Cathala, and P. Poncelet (1998), “The PSP Approach for Mining Sequential Patterns”. *Proceedings European Symp. Principle of Data Mining and Knowledge Discovery (PKDD ’98*), pp. 176-184.
17. Gasmi G., Ben Yahia S., Mephu Nguifo E., Slimani Y. (2005). "A new informative generic base of association rules", *The Ninth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-05)*, LNCS, Springer Verlag, Hanoi, Vietnam.