

Automatic Generation of Learning Designs: Two Planning Approaches Comparison

Lluvia Morales

Advisors: Luis Castillo and Juan Fernández-Olivares

Department of Computer Science and Artificial Intelligence,
ETSIIT - University of Granada, 18071, Granada, Spain,
{lluviamorales, L.Castillo, faro}@decsai.ugr.es

Abstract. The application of Planning and Scheduling techniques in e-learning problems is not a recent subject. However, the automatic generation of planning domains in this area is a new and interesting topic of research. In this paper we compare two *automatic domain generation techniques* for two planner paradigms and its resulting plans called Learning Designs; its quality(compared with a real domain), practical application and expressivity are some of the criteria used for the comparison.

Key words: Planning and Scheduling, e-learning, Knowledge Engineering, IMS standards, Automatic Generation of Planning Domains

1 Introduction

The application of Planning and Scheduling techniques in e-learning problems is not a recent subject. Since 1986 Peachy and McCalla[14] proposed the integration of planning and scheduling techniques to improve the learning path of Intelligent Tutoring Systems. However, the automatic generation of planning domains in this area is a new and interesting topic of research. This automatic generation, made through a complicated knowledge engineering process, saves to the e-learning community money and time that planning experts spend in the domain generation for each e-learning course. It must be interesting to point that a little professional training enterprise can offer at least 100 of courses with a minimum of 10 students, and an open university can offer hundreds of courses.

Recently Kontopoulos, et al.[9] and Camacho, et al.[2] have applied the idea of automatic generation of planning domains taking advantage of an ontology based knowledge recovery to generate the domain for a state-based planner. On the other hand, K. Erol have said in [4] that a HTN representation is more flexible and expressive than STRIPS-style planning. However, last cited authors have no used a HTN planner for the plan generation because of the lack of information that ontologies can provide at this time.

In this paper we have proposed two knowledge recovery procedures in order to automatically generate a planning domain for two different planners, an state-based one called LPG-td and an HTN planner named SIADEX. This

knowledge recovery procedure is based on a knowledge engineering process over an exhaustive IMS-metadata[1] labeling of the learning objects (teaching material like exercises, lectures, simulations, etc.) in an e-learning course. And, these knowledge is used by two different algorithms that automatically generate the planning domain for each planner.

After the domain generation, we have compared the quality and expressivity in representing the e-learning domain of each planning domain paradigm. The generation of a valid plan, that is called Learning Design(LD)[8] by the e-learning community, is compared too as a key point of this paper.

It is important to remark that the IMS-Learning Information Package[8] standard, related to the student profile, is used to create the pddl problem file definition. The pddl domain and problem are used by planners to generate a consistent plan and both are generated taking advantage of well known standards. The resulting plan is transformed into a key standard too.

Through this paper, first a description of the e-learning domain we are working on is given. Secondly the knowledge engineering and domain generation algorithms for two kinds of planning paradigms are explained. Subsequently a comparison between them is made taking into account the quality, practical application and theoretical expressive implications from each one. And finally, the conclusions of this research and future work are addressed in Concluding Remarks section.

2 E-Learning Domain

In order to be able to start with the knowledge engineering and automated planning domain generation process is essential to extract mandatory information about learning objects of the subject using their metadata and the user models of each one of the students registered in the course.

To make easy the planning metadata abstraction of learning objects, we have divided them in two kinds: compound objects and primitive objects (that have no objects as part of them), which are translated to compound tasks and primitive actions in a HTN planning domain, respectively. From learning objects we can extract two kinds of metadata.

Structural Metadata. **Is-Part-Of.** It describes a hierarchical compositional structure between learning objects through the course as is shown in figure 1. **Is-Based-On,** provides ordering relations between tasks and actions. **Requires.** Reports content dependencies between tasks.

Objects Attributes Metadata. **Language.** Object required language i.e. spanish, english, etc. **Learning Resource Type.** Describes what kind of learning resource we are working on, i.e. lecture, simulation, exercise, etc. **Other Platform Requirements.** Describes if there are special hardware or software requirements to use the learning object. **Difficulty.** Defines the performance level required by the student for this object to be in his plan. **Coverage.** If this

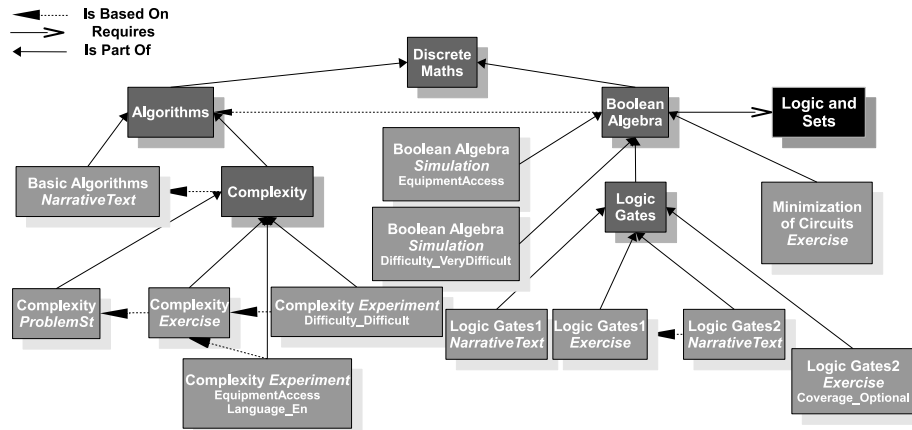


Fig. 1. Structural Relations and Objects Attributes in Discrete Maths Framework: Every primitive object in light gray which is part of a compound object in gray, has a **Learning Resource Type** and could have attributes as **English Level** or **Coverage**. Each compound object has an **Is-Based-On** or **Requires** order relation and **Is-Part-Of** relations from one or several actions.

metadata takes the “Optional” value it means that the learning object could be omitted in the plan if necessary (whenever there the student has no time left, or if the object is an extra-one). On the other hand it takes the “Mandatory” value. **Typical Learning Time**. Temporal constraints imposed by instructor for every object in minutes.

In order to personalize the LD, our algorithm use the next *student profile* options provided by the Learning Management Systems(LMS) we are working on, called ILIAS[7] and Moodle[10], and based on the IMS-LIP. **English Level**: to determine if he could take a high level english object. **Equipment**: defines software and hardware availability of the student like java environment, bandwidth, etc. **Previous Courses Level**: score in a related course or subject. **Performance Level**: performance level of the student. **Available Time**: special time student needs. **Learning Style**: to offer each user a set of learning objects with the temporal sequence that best fits his/her learning style. This style is defined by a psychological test, in this case, the Honey-Alonso[6] test, that is answered by the student in his/her first visit to the LMS.

The learning objects metadata of the IMS-MD standard are supported by the LMS ILIAS or, through a metadata editor called RELOAD[15], by Moodle. The work to introduce those metadata is given to an expert in the curricular domain, usually the tutor of the course. The student profile characteristics of IMS-LIP are options that most of the Learning Managements Systems support and can be required fields to the student.

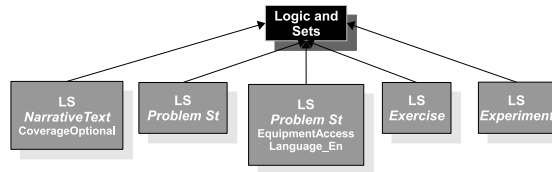


Fig. 2. Required Object from an external course, in this case from *Logic and Maths*. It could be any compound object from a different course or a complete course itself

3 Planning Domain and Problem Generation

Metadatas and students profiles are used in both translation processes, to HTN and state-based domains. This processes have been carried out through an algorithm for HTN planners, specifically SIADEX[3], and a different one for the state-based paradigm with LPG-td planner[11] as an example of it.

Once both algorithms have been implemented and their efficiency proved, significant differences have been found. Those differences are discussed at next section. In the meantime, a brief description of the referenced algorithms is shown in the upcoming subsections.

3.1 HTN

The algorithm for a hierarchical paradigm domain generation take place in two steps:

Defining Primitive Actions. HTN primitive actions are extracted from those learning objects that have no objects related through the **Is-Part-Of** relation. These actions will take into account the following issues. The duration of the action will be its **Typical Learning Time**. If there are some especial **Equipment** or **Performance Level** needed, or it has been written in a different **language** to the common language of the course, then the list of preconditions will include this conditions linked for an **and** operator for the action to be included. An example of these preconditions is given in the definition of the Complexity-Experiment primitive action showed on the following lines:

```

(:durative-action Complexity-Experiment
 :parameters(?who - student)
 :duration (= ?duration 38)
 :condition(and
  (>=(language-level english ?who)50)
  (equipment ?who multimedia))
 :effect ()
)
  
```

Defining Compound Tasks. The previous domain extraction procedure allow us to generate the primitive actions of an HTN domain, but there are some actions which have special metadata values that require to generate new compound

tasks. That is the case of every atomic learning object labeled as “optional”, for this kind of objects a new task is created with two different methods, one of them includes its corresponding action and the other does not.

When more than one atomic object exists with the same name it means that there are different ways of performing the same learning action and, probably, under different conditions. This is encoded as an additional compound task that includes a unique method containing a single action. There will be a primitive action for each atomic object, so that, the compound task forces the introduction of one of these actions that will be found by planner through search and backtracking in the case that the conditions of the actions are not met.

These new tasks, optionals and multiples, inherits every structural metadata from the primitive action which they come from.

After to add the additional compound tasks to encode part of the adaptation scheme. There may be compound actions, like *Complexity* in Figure 1 whose constituent parts are fully ordered according to the **Is-Based-On** relation and, therefore, there will always be included in the same order in any learning path.

However, other compound tasks like *Logic and Sets* in Figure 2 do not have its constituent objects fully ordered.

Implicit order relations are defined in our approach to encode different orderings for every compound task. For example, the next rules are used to encode the order that must be followed, according to the **Learning Resource Type** of the objects included in the compound task:

```
((Learning Style ?student Theoretical)
 (Problem-statement Simulation Experiment Exercise Lecture))
((Learning Style ?student Pragmatic)
 (Simulation Experiment Exercise Problem-statement Lecture))
```

They mean that a different order is given if the student has a theoretical **Learning Style** or if he/she has a pragmatic one.

And finally, there is the case when a compound object **requires** another object that belongs to any other course. An example of it is *Logic and Sets* object which is required by *Boolean Algebra* in Figure 1. The *Boolean Algebra* task includes two different decompositions, one of them for the case when the student has successfully passed the required object, and the other one for the case when the student has not passed it and, thus, *Logic and Sets* will have to be included in his/her learning path.

In the problem generation process each student characteristics from his/her profile are translated into predicates of the initial state. The goals of the problem are defined by the instructor amongst the list of compound tasks available in the domain and they will appear totally ordered in the goal section besides the time restrictions given by the **Available Time** characteristic from every student.

This subsection has shown that a valid HTN domain and problem may be extracted from a well structured learning objects repository just by processing the knowledge of its standard metadata. Although the procedure has not been fully presented, more details can be found at [3].

3.2 State Based

Unlike HTN planners, a state-based planner has not compound tasks in its PDDL domain definition. That is the reason why we have to use a different algorithm to traduce the metadatas into the planning domain where structural relations have to be encoded as preconditions of primitive actions.

The algorithm first analyze attributes and relations from each action. The duration of the action will be its **Typical Learning Time**. Whether there are some especial **Equipment** or **Performance Level** needed, or it has been written in a **language** different than the common language of the course, then the list of preconditions will include this conditions for the action to be included.

Since a non HTN planner does not have a compositional hierarchy of tasks and actions to encode is-part-of relationships, the algorithm defines “order preconditions” for each action according to the requirements given by them structural relations. This second step is more tricky because it implies to check the course graph showed in Figure 1 in two steps¹:

1. Forming groups linearly arranged with those primitive actions (even in parallel because there are objects with the same name, but different attributes like *Complexity Experiment* in Figure 1). These groups (called primitive groups) are ordered according to the **Is-Based-On** relation or the **Learning Style** rule mentioned in Subsection 3.1. After the ordering process, a precondition like **actionTitle_done** is assigned to every action that has a precedent action in the primitive group. This is a landmarking strategy to detect when a given learning object has been passed and is also used in other approaches like in [5].
2. Subsequently, we explore each one of the hierarchical levels occupied by compound objects and inherit its ordering relations into the first action of the “primitive group” previously formed that is part of this compound object. With this inheritance relations we can reorder those groups as in the first step, to create new ones, and to obtain the same hierarchical structure than with an HTN paradigm.

Previous process is done till it covers every order relation from compound objects which are part of a root compound object as *Discrete Maths* in figure 1.

Finally we have to connect this root objects according to **Requires** relations. For example, if *Boolean Algebra Simulation* is the first action in the primitive group of *Boolean Algebra*, then the last action of the primitive group from *Logic and Sets* has to be done as precondition of *Boolean Algebra Simulation*.

An example of the primitive action generated by this state-based approach is described next:

¹ However, it must be said that this encoding is only necessary when the learning objects repository is grouped into chapters, sections, subsections and so on. This is the case of most repositories although there may be repositories without any hierarchical structure.

```

( :action actionLS-ProblemSt
  :parameters (?who - Student)
  :precondition
    (and
      (>=(language-level english ?who)50)
      (equipment ?who multimedia)
      (or
        (and
          (LSTheoretical ?who) (actionLSNarrativeText_done ?who) )
        (and
          (LSPragmatical ?who) (actionLS-Exercise_done ?who) ) ) )
  :effect
    (actionLS-Exercise_done ?who)
)

```

In the process of the problem generation each student features from his/her profile is translated into predicates of the initial state and the goals are defined by the instructor amongst the list of tasks available in the real world domain. From every goal task selected by the instructor, a little algorithm have to find its final ordered action and, finally, those actions will appear totally ordered in the goal section.

Using this process is possible to generate a valid state-based planning domain in PDDL with the same expressive capability than the metadatas repository. The detailed process and problem generation can be seen in [12].

4 Models Comparison

After the implementation of both translation processes described in previous section, significant differences have been found which are the basis of the comparison made as the core of this research.

In following subsections we will explain those differences between the state-of-the-art planners that influence the automatic domain generation process and the quality of the final plan compared with a handmade LD.

4.1 Structural Processing

The hierarchical nature of an e-learning course domain is an advantage for HTN planners because their planning domain in PDDL represents in a clear way the real world domain studied in Figure 1.

For example, chapters, sections, subsections, and topics are represented as tasks that contain different sequences of “subtasks” and/or learning durative actions. Learning material, also called learning objects, are represented as durative actions with an associated duration and other attributes related.

The State-Based domain extraction is more complicated and less clear than HTN because it only allows the primitive action declaration and every structural relation must be inherited as preconditions through the complicated and expensive process tour through the e-learning course graph.

4.2 Temporal Knowledge Representation

The temporal representation of the knowledge for each planner is determinant in what we can represent about an e-learning domain.

Two limitation kinds over the temporal knowledge representation have been found in the LPG-td planner. Compared with SIADEX planner, those limitations restricts the planning domain expressivity of the e-learning domain requirements. This is not a rule for every state-based planner, but we have used LPG-td because it is one of the most used and robust planners to solve problems with temporal restrictions in its domain.

Those temporal knowledge representation differences are addressed in next subsections.

Deadlines Usually, e-learning courses have a limit date, like a deadline, to carry out the goals defined by the tutor. Also, in distance courses, students could have the opportunity to define their own deadlines before the global one; this property is represented as the **Available Time** of the student and it is used to determine a goal inside the HTN planning problem.

The state-based planner LPG-td does not support deadline goals in a natural way. Frequently, the method used by state-based planners, that support temporal constraints, is to use a minimization and/or maximization metric (depending of the planner) over the time required by durative actions to carry out a global goal.

In this case the LPG-td minimizes the required time to perform the course in a problem that includes every student particular goal, but the final student goal is only a little part of the global one. A pddl problem file for each student can be made too, but this implies to execute the planner as many times as students are addressed in the course in order to minimize every particular plan. Hence, dealing with global and individual deadlines at the same time is not possible.

The following point about plan duration is another derived limitation.

Durations For an HTN planner the deadline for every student depends of the **Available Time** defined by he/she. Then the duration of the plan should be as large as the student wants to be or, if the students does not define any **Available Time**, as the global duration defined by the tutor.

In addition to this, there are optional actions, i.e. extra material for advanced students, that can or cannot be included in the plan according to the student characteristics and available time. Optional actions can be represented in the HTN domain as a method called by an “optional” task if there are enough time to do it according to the established deadline in the goal.

Moreover, when a student have no restrictions over the equipment, language or another issue that could be a precondition to realize a primitive action, he/she can take any of the multiple actions with the same name even the larger one if he/she has enough available time.

In a state-based domain, the inference engine should always try to minimize the time needed to fulfill the specified goal. Then, for LPG-td planner, the optional action declaration has no sense because it never includes the optional action in order to minimize the duration of the plan. The same applies in the multiple actions case where the planner does not take into account the existence of deadlines, and hence only include the shorter one.

Therefore the plan duration of the same student could be not the same for the two used approaches and as personalized as could be for the state-based approach.

4.3 Obtained Plans

Considering the aspects previously explained, now we are going to study two examples where clearly those differences are described.

CASE 1: A student with no time limitations and the availability of any action precondition like **English Level**, **Previous Courses Level**, **Equipment**, etc. Plan length is not the same for HTN and state based approaches because of the learning objects taken into account. Optional objects are not in the state-based plan example and only the multiple actions that requires less time are included in the plan, although the student can take the larger ones. In HTN planner, every action can be included in the plan because it can be than larger as the number of actions in the course deadline time the tutor have considered.

CASE 2: A student with **Available Time** restrictions and that have no enough level for some of the preconditions required by the actions. The actions duration on the student plan depends of the actions that he/she is able to take. Those actions are assigned to the student according to the capabilities where student's level fulfill the expected level by the action precondition. He could have plans with the same length for any paradigm, but the **Available Time** can be greater than the minimum time found by LPG-td. In this case, the deadline plays a crucial role.

In previous cases we can clearly note that generated plans have significant differences depending of the planner used to represent the e-learning domain. In next subsection we are going to explore those differences deeply.

4.4 Comparative Resume

In Table 1 we show three basic aspects that we are comparing to select the better approach that could deeply solve the detailed domain needed by a Learning Design construction.

Although both planners are deeply used in real domains, under temporal restrictions, we can observe that domain generation is easier for HTN domains because state-based has no the expressive capacity needed by an e-learning domain. At this point we note that the preconditions generation is expensive for LPG-td domains.

DOMAINS			
	<i>Hierarchical</i> (SIADEX)	<i>State-Based</i> (LPG-td)	<i>Real</i> (e-learning)
<i>Practical Application</i>	Similar representation to the real one. "Easy" domain generation.	Complexity in the domain generation algorithm. Too many preconditions in generated primitive actions.	Expensive generation. Impossible to be continuously following the course performance.
<i>Quality</i>	Adaptive Scheduling according to availability. Optional activities in consonance with capacity and time.	Yes: general adaptation No: extra material temporal adaptation	Quality based on the tutor experience. Adaptation, timing, colaboration, extra material.
<i>Expressive Features</i>	Most of the required capacities in this phase of the proposed solution.	No added value given earlier proposals.	Sets standards to follow for the two previous proposals. Every allowed capacities by IMS-LD .

Table 1. Comparative table between the real and planner domains, three aspects for the LD generation and resulting.

Moreover, deadline support and action optionality of SIADEX planner make it better than LPG-td for student adaptation support according to the requirements needed by the e-learning plan requirements.

5 Concluding Remarks

This paper has presented two approaches to extract automatically a course domain and problem in PDDL and made a comparative between them. Generated domains and problems are used by SIADEX and LPG-td planners, HTN and State-Based planner paradigms respectively, in order to obtain an adapted learning path of the encoded course that is known by e-learning community as a Learning Design. A broad expressiveness is hoped from a Learning Design representation, it implies that the expressiveness of the planning domain must be certainly wide too.

On the other hand, PDDL domains are generated by hand in most of the cases. In this paper we apply knowledge engineering techniques in order to generate the domains automatically according to a previous metadata labeling based on the IMS-MD standard that is well known by the e-learning community.

According to the situations explained in the previous paragraphs we appreciate that the automatic generation of the planning domain by using the IMS-metadata labeling of the learning objects that are part of a e-course is a novel approach that save work and time. And the comparison between the domains of these planning paradigms have sense, in order to select the better approach that could deeply solve the detailed domain needs by a Learning Design construction.

In conclusion, the HTN planner SIADEX have represented the e-learning domain in a better way than state-based LPG-td planner.

However, there are still issues that need further study as the adaptation of the learning path to run-time information like the intermediate evaluations results during the development of the course. It implies new precondition kinds and the inclusion of last viewed or reinforcing materials, besides the use of replanning or continual planning techniques. Collaborative activities there are another interesting subject to study because of time restrictions and parallelism that we have to handle with.

Acknowledgments. This work is supported by the Mexican National Council of Science and Technology besides of the project ADAPTAPLAN. Adaptation based on machine learning, user modelling and planning for complex user-oriented tasks(TIN2005-08945-C06-02) from the Spanish Ministry of Education and Science.

References

1. IEEE Standard for Learning Objects Metadata, <http://ltsc.ieee.org/wg12/>
2. Camacho, D., R-Moreno, M.D., Obieta, U.: CAMOU: A Simple Integrated e-Learning and Planning Techniques Tool. In: 4th International Workshop on Constraints and Language Processing. (2007)
3. Castillo, L., Morales, L., Gonzalez-Ferrer, A., Fdez-Olivares, J., García-Pérez, O.: Knowledge Engineering and Planning for the Automated Synthesis of Customized Learning Designs. In: Current Topics in Artificial Intelligence (CAEPIA 2007). Springer LNAI 4788 (2007)
4. Erol, K.: Hierarchical task network planning: Formalization, Analysis, and Implementation, Ph.D. Thesis, University of Maryland, College Park, MD, 1995.
5. Garrido, A., Onaindia, E., Sapena O.: Planning and Scheduling in an e-learning environment. A constraint-programming-based approach. In Press: Engineering Applications of Artificial Intelligence, Special issue on constraint satisfaction techniques for planning and scheduling problems, 2008.
6. Honey Alonso Learning Style, <http://www.estilosdeaprendizaje.es/chaea/chaea.htm>
7. ILIAS Learning Management System, <http://www.ilias.de/ios/index-e.html>
8. IMS-GLC IMS Global consortium, In: <http://www.imsglobal.org/>
9. Kontopoulos, E., Vrakas, D., et al.: An Ontology-based Planning System for e-course Generation. In: Expert Systems with Applications, Elsevier, 37 (1).(2007)
10. MOODLE, Modular Object-Oriented Dynamic Learning Environment. <http://moodle.org/>
11. Morales, L., Castillo, L., Fernández-Olivares, J., González-Ferrer, A.: Automatic Generation of User Adapted Learning Designs: An AI-Planning Proposal. In Press: Lecture Notes in Computer Science, Adaptive Hypermedia 2008.
12. Morales, L., Castillo, L., Fdez-Olivares, J., and Gonzalez-Ferrer, A.: Building Learning Designs by Using an Automatic Planning Domain Generation: A State-Based Approach. In: Frontiers in Artificial Intelligence and Applications (STAIRS2008). IOS Press 179 Ed. A. Cesta and N. Fakotakis.

13. Myers, K.L.: Towards a Framework for Continuous Planning and Execution. In: Proceedings of the AAAI Fall Symposium on Distributed Continual Planning. (1998)
14. Peachy, D.R., McCalla, G.I.: Using Planning Techniques in Intelligent Tutoring Systems. In: International Journal of Man-Machine Studies 24, 77-98. (1986)
15. RELOAD Reusable eLearning Object Authoring and Delivery. In: <http://www.reload.ac.uk>/<http://www.reload.ac.uk/>