



Using linear programming to solve clustered oversubscription planning problems for designing e-courses

Susana Fernández*, Daniel Borrajo

Departamento de Informática, Universidad Carlos III de Madrid, Avda. de la Universidad, 30 Leganés, Madrid, Spain

ARTICLE INFO

Keywords:
Automated planning
Application
e-Learning
Linear programming
Oversubscription

ABSTRACT

The automatic generation of individualized plans in specific domains is an open problem that combines aspects related to automated planning, machine learning or recommendation systems technology. In this paper, we focus on a specific instance of that task; that of generating e-learning courses adapted to students' profiles, within the automated planning paradigm. One of the open problems in this type of automated planning application relates to what is known as oversubscription: given a set of goals, each one with a utility, obtain a plan that achieves some (or all) the goals, maximizing the utility, as well as minimizing the cost of achieving those goals. In the generation of e-learning designs there is only one goal: generating a course design for a given student. However, in order to achieve the goal, the course design can include many different kinds of activities, each one with a utility (that depend on the student profile) and cost. Furthermore, these activities are usually grouped into clusters, so that at least one of the activities in each cluster is needed, though many more can be used. Finally, there is also an overall cost threshold (usually in terms of student time). In this paper, we present our work on building an individualized e-learning design. We pose each course design as a variation of the oversubscription problem, that we call the *clustered-oversubscription* problem, and we use linear programming for assisting a planner to generate the design that better adapts to the student.

© 2010 Elsevier Ltd. All rights reserved.

1. Introduction

Automated planning is the branch of artificial intelligence that studies the computational synthesis of ordered sets of actions that perform a given task (Ghallab, Nau, & Traverso, 2004). A planner receives as input a collection of actions (that indicate how to modify the current state), a collection of goals to achieve, and a state. It then outputs a sequence of actions that achieve the goals from the initial state. Given that each action transforms the current state, planners may be viewed as searching for paths through the state space defined by the given actions. However, the search spaces can quickly become intractably large, such that the general problem of automated planning is PSpace-complete (Bylander, 1994).

The evolution of automated planning can be seen from three different interrelated perspectives: planning techniques, expressiveness of planning languages, and applications. In relation to the first one, it is one of the most active areas, since very powerful domain-independent techniques have been devised in the last two decades. However, we can see that there is no such counterpart in the other two. The International Planning Competition (IPC¹) has

been one of the major drivers of this advancement, and the standard language, PDDL (Gerevini & Long, 2005), one of its bases. Although, PDDL is gaining in expressiveness, very few planners can handle the full set of features that it defines. This is due mainly to the competition: if competition does not focus on specific aspects of the planning task, then it is hard to see which technique is better for solving each aspect. Unfortunately, most real world applications require the use of some features of PDDL that are not handled by most planners, as well as the use of some representation features that are not present in PDDL. So, in relation to the third perspective, when trying to solve different real world applications, then one is very restricted to the planners that can be used. Another relation between current planners and their ability to directly be used in applications is that most planners are still in an advanced prototype status. So, as we will see later, they either crash or fail on solving problems in domains that do not belong to the competition. This is probably due to small implementation errors, but this makes it unfeasible to use them “as is” in current applications. Thus, there is still a gap between state of the art planners and their direct application. This can be easily explained given the two different emphasis on both communities: either generating powerful search techniques that can handle some expressiveness, and the need of polished implementations to be used as “off-the-shelf” components for applications.

Despite these problems, the application of planning techniques to real problems sometimes requires solving interesting associated

* Corresponding author.

E-mail addresses: susana.fernandez@uc3m.es (S. Fernández), daniel.borrajo@uc3m.es (D. Borrajo).

¹ <http://www.icaps-conference.org/index.php/Main/Competitions>.

problems that can be useful in more general contexts. This is the case of the work presented in this paper. The application area is the generation of learning designs adapted to students' profiles. And, the associated problem is a variation of the oversubscription problem (OSP) that we have called clustered oversubscription. OSP was first introduced by Smith (2004) and the objective of the planning process in the presence of goals oversubscription is not to find a feasible plan accomplishing all the goals, but to find a plan which reaches some of them, maximizing the sum of utilities of achieved goals. As Smith pointed out, many NASA planning problems are over-subscription problems. For example, space and airborne telescopes, such as Hubble, SIRTf, and SOFIA, receive many more observation requests than can be accommodated. As a result, only a small subset of the desirable requests can be accomplished during any given planning horizon. For a Mars rover mission, there are many science targets that the planetary geologists would like to visit. However, the rover can only visit a few such targets in any given command cycle because of time and energy limitations, and limitations on the rover's ability to track targets.

In this paper, we describe our work on a task that also presents a kind of oversubscription: automatic generation of e-learning courses. An e-learning task can be posed as: given a specific student, and a course defined by a pool of diverse learning activities, automatically generate a learning design for that student using the given learning activities that better suit the student profile. This domain has some interesting characteristics that resembles OSP: (1) each activity has a utility, and a time (cost) that it takes to complete it; (2) the utility that a given learning activity will provide a student depends on some student features; (3) as a hard constraint, the time to execute all learning activities in the final plan must be bound by a threshold (the total amount a student can dedicate to that course); (4) and, as with usual planning tasks, the activities present causal relationships among them (the student should follow some learning activities before others). Besides, it has an extra characteristic that comes up with the clustered oversubscription problem: (5) each learning activity belongs to a given cluster, and the final design should have at least one activity of each cluster, potentially more. In clustered oversubscription, we perform some analysis before planning that selects a subset of the learning activities that is worth pursuing, given that they maximize utility (or have a high utility), and fulfill the constraint that the sum of their cost (time) is less than the cost limit (time threshold). The approach for performing this selection consists of posing the problem as a clustered-knapsack problem and using linear programming. Then, we translate this subset in two different ways, as preferences or as a metric.

Our work is based on previous pedagogical research where they have tested a valid way for measuring the utility each learning activity reports to the students (Baldiris et al., 2008). So, we focus on the problem of sequencing learning activities in a course adapted to every student's profile by trying to maximize the total utility. Other relevant work concerning the use of AI planning and scheduling techniques for sequencing learning activities, according to different student's profiles and pedagogical theories, is reported in Castillo et al. (2009), Ullrich and Melis (2009). But, they did not focus on the goal of finding a feasible plan that maximizes some measure of utility by reaching a subset of the goals.

Our aim is that the proposed solution would be general enough so it can be extrapolated to other domains. An example of domain that have similar clustered-oversubscription problems is a variation of the Rovers domain. The rovers need to take several samples, each sample of a specific kind (cluster) of rock, and there are several rocks of each kind. Also, taking those samples takes

some time, and they have a specific time threshold to perform the activities. Other examples are earth observing satellites (clusters are areas with the same information, as big forests, mountains, or oceans) or the planning and optimization of resources in bus transportation networks (clusters are the different lines that compose the transportation network and the goal is optimizing their resources and minimizing costs).

The contribution of the current paper is twofold: modelling e-learning tasks as clustered-oversubscription planning problems; and providing a generic approach for solving the resulting problems, applicable in other contexts than e-learning applications, as well. The remainder of the paper proceeds as follows. Next section introduces automated planning and its representation language PDDL. Section 3 describes how we have modelled the e-learning application in PDDL. Section 4 explains the approach that we have devised for solving the clustered-oversubscription problem by performing an action selection pre-processing to help the planning task using linear programming. Section 5 reports the experiments performed to test the validity of the approach. Section 6 describes the related work. Finally, last section draws the conclusions and our future work.

2. Automated planning

In general, two elements typically define a planning task: a domain definition comprised of a set of states and a set of actions; and a problem description composed of the initial state and a set of goals that the planner must achieve. Here, a state description includes a collection of grounded predicates and functions, while an action includes the parameters or typed elements involved in its execution, a set of preconditions (a list of predicates describing the facts that must hold for the action to apply), and a set of effects (a list of changes to the state that follow from the action). The planning task consists of obtaining a partially ordered sequence of actions that achieve all of the goals when executed from the initial state. Each action modifies the current state by adding and deleting the predicates represented in the domain-action effects. For example, the Rovers domain (inspired by planetary rovers problems) requires that a collection of rovers navigate a planet surface, finding samples and communicating them back to a lander. There are three types of samples, named soil, rock and image. The domain includes nine actions:

- (navigate ?x – rover ?y – waypoint ?z – waypoint) navigates a rover from one point to another. Both points must be visible and traversable.
- (sample_soil ?x – rover ?s – store ?p – waypoint) stores a soil sample in a rover that is located in a specified point.
- (sample_rock ?x – rover ?s – store ?p – waypoint) stores a rock sample in a rover that is located in a specified point.
- (drop ?x – rover ?y – store) empties a rover so that it can collect a new sample.
- (calibrate ?r – rover ?i – camera ?t – objective ?w – waypoint) calibrates a rover's camera using a specified calibration target which is visible from the current rover position.
- (take_image ?r – rover ?p – waypoint ?o – objective ?i – camera ?m – mode) takes an image of an objective in a specified mode. The objective is located in a point visible from the current rover position.
- (communicate_soil_data ?r – rover ?l – lander ?p – waypoint ?x – waypoint ?y – waypoint) communicates a soil data to the lander.
- (communicate_rock_data ?r – rover ?l – lander ?p – waypoint ?x – waypoint ?y – waypoint) communicates a rock data to the lander.

متن کامل مقاله

دریافت فوری ←

ISIArticles

مرجع مقالات تخصصی ایران

- ✓ امکان دانلود نسخه تمام متن مقالات انگلیسی
- ✓ امکان دانلود نسخه ترجمه شده مقالات
- ✓ پذیرش سفارش ترجمه تخصصی
- ✓ امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
- ✓ امکان دانلود رایگان ۲ صفحه اول هر مقاله
- ✓ امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
- ✓ دانلود فوری مقاله پس از پرداخت آنلاین
- ✓ پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات