The Academic Advising Planning Domain

Joshua T. Guerin, Josiah P. Hanna, Libby Ferland, Nicholas Mattei, and Judy Goldsmith
Department of Computer Science
University of Kentucky
Lexington, KY 40506-0633
jtguer2@uky.edu, jpha226@g.uky.edu, libby.knouse@uky.edu, nick.mattei@uky.edu, goldsmit@cs.uky.edu

Abstract

The International Probabilistic Planning Competition is a leading showcase for fast stochastic planners. The current domains used in the competition have raised challenges that the leading deterministic-planner-based MDP solvers have been able to meet. We argue that in order to continue to raise challenges and match real world applications, domains must be generated that exhibit true stochasticity, multi-valued domain variables, and concurrent actions. In this paper we propose the academic advising domain as a planning competition domain that exhibits these characteristics. We believe that this domain can build upon the success of previous contests in pushing the limits of MDP planning research.

Introduction

Generating new problem sets to challenge state-of-the-art probabilistic planners is often difficult. Planners have evolved in leaps and bounds in a very short time, opening the door to whole new classes of problems that can realistically be solved by computer programs. As a driver in planning research, the International Probabilistic Planning Competition (IPPC) has presented a number of domains meant to continue pushing the field in new and exciting directions.

As the IPPC has continued, domains have become increasingly complex and stochastic. In recent contests, strong planners that solve deterministic instances of stochastic domains (or "replanners" (Little and Thiébaux 2007)) have continued to outstrip their inherently probabilistic counterparts. This indicates to us that the domains that have been used are still not stochastic enough to closely model the complexities of real world planning problems. We suggest that current real world domains can be modified to present more difficult problems, and that new domains requiring different problem solving strategies can help provide more challenges for planners.

Current competition domains have been designed to represent real world scenarios for probabilistic planners. In the Elevator domain, for example, a controller coordinates a bank of elevators to pick up passengers when requests are received, and delivers them to the destination floor. This domain assumes that each elevator is acting separately and actions are performed sequentially. This makes the state space less complex, allowing deterministic replanners to produce strong solutions. In most real world problems, however, the most comprehensive models are highly stochastic and consider actions taken concurrently. This domain can be modified by assuming that the controller is directing elevators in concurrent pairs or groups, instead of individually and sequentially. This introduces joint actions that greatly increase the size and complexity of the state space. As the state space grows more complex, it becomes more difficult for deterministic planners to produce a good solution, and probabilistic planners using concurrent action planning become better choices due to a more complete consideration the domain (Sanner 2008). We believe that many other competition domains could be represented in this fashion, and that continuing to introduce domains requiring different planning methods provides interesting and valuable challenges to competitive planners.

Our domain, the academic advising domain, represents a real world model in which concurrent actions must be considered in order to reach an optimal solution. Academic advising, when treated as a probabilistic planning domain, is rich with the qualities used to classify domains as probabilistically interesting (Little and Thiébaux 2007). The presence of avoidable dead-ends, near-identical trajectories with distinct outcomes, multiple distinct trajectories, and mutually exclusive actions, as well as a high degree of unpredictable outside influences, makes for a challenging environment for state-of-the-art planners today. Planners in this domain must consider factors such as past performance history, courses completed, and above all else the student's ability to take multiple courses concurrently — all with the aim of maximizing the student's GPA as well as satisfying the requirements for graduation with a specific area of study.

This domain presents a large, complex, and easily scalable state space, ideal for a challenging competition. Most importantly, while deterministic planning may eventually produce a solution, optimal solutions can only be found using stochastic, concurrent-action planning. We believe that this quality could make the academic domain a good addition to the domains already used in competition, and help introduce even more "spice" into an already challenging problem set (Sanner 2008). Competitive domains have always been used to test the very best in planners, and it is our hope that the academic domain might be able to continue in this tradition.

In the section "The Academic Domain", we describe the real-world advising domain, and in the section "The Academic Domain Model Generator Problem Structure", we briefly sketch the proposed domain generator. Note that others have considered MDP-based advising domains. Both Khan et al. and Dodson et al. have used hand-constructed models of advising to motivate work on explanation generation for MDP policies (Khan, Poupart, and Black 2009; Dodson, Mattei, and Goldsmith 2011).

The Academic Domain

Good academic advising in the American educational system involves many decisions. In most degree programs, students choose electives to satisfy multiple, sometimes overlapping requirements. Their choice of electives, the order in which courses are taken, and the choices of which courses to take together, all have enormous effects on the student's success and their enjoyment of their academic career.

We can model advising as a factored MDP. A student's course history and grades determine their current state. Based on this state, a student can select one or more courses to take during the next semester. More formally, states of the MDP are student transcripts. Each variable represents a possible course, with values (let us say) HIGH, LOW, FAIL and NOT TAKEN. An action specifies a course to take. The courses available for selection at any stage depend on the fulfillment of prerequisites. Long prerequisite chains mean a policy with a time bound on the number of semesters (or a discounted reward) must begin the long chains early enough to ensure the later courses can be taken.

Note that students who have taken the same courses can be in very different states if one has received high grades and the other has received low or failing grades. Two students with the same GPA may be in different states because they took different courses.

The effects of actions are stochastic. Grading scales are less than precise measures, and small exogenous events (something seen on the way to an exam, a girlfriend with tutoring skills, a fight with a friend) can have large effects on grades.

A policy specifies sets of concurrent actions at each time step, since students take sets of courses each semester. While the goal of the policy can vary, almost all academic domain policies will seek to avoid states that involve failed classes.

Probabilistic Outcomes

The effects of sets of actions in the academic domain are uncertain. Adding to the uncertainty, the number of possible next states is large for even a small number of concurrent actions. Consider a set of four classes that a student has taken. There are 3^4 possible outcomes, ranging from HIGH, HIGH, HIGH, HIGH to FAIL, FAIL, FAIL, FAIL. Each outcome, or at least each set of FAILs, requires different responses. Required courses must be repeated. Prerequisites and predictors of success *should* be repeated. In some cases, a LOW should be repeated, but not always. For instance, if it were shown that students who had taken first-semester calculus performed better in discrete mathematics then a policy that had a student take calculus before discrete math would be more likely to result in a higher grade for the discrete math class, and a LOW grade in calculus might skew likely grades in several later courses toward LOW or FAIL.

As a result of this combinatorial explosion, the underlying planning problem is not easy to determinize.

University curricula can lead to an enormous state space, and real-world weakly coupled MDPs. Courses that are required for Human Ecology majors may have little impact on Computer Science majors. There is a wealth of potential challenges in modeling actual curricula, and building planners that can recognize and leverage the compartmentalization of individual programs and majors. However, a realistic model of transition probabilities can be built using data mining techniques on grade data from students to have taken courses previously (Guerin and Goldsmith 2011).

Concurrent Actions

Within the academic domain, students take courses in sets rather than one at a time. The result of this is a much larger set of choices for each stage of the plan. The number of courses taken in a semester can vary so that the policy does not need to have a uniform number of courses each semester. Constraints can be specified for the domain so that a student must remain full time (establishing a minimum number of courses for each semester) and/or to establish a maximum number of courses. This can be further complicated by having the number of courses taken affect the probability of success in each course. For example, a student taking four courses is more likely to achieve higher grades than a student taking the same four as well as three additional courses. It should be noted that this planning domain does not have to involve concurrent action planning if the number of courses to be taken at a time is limited to one.

We can model the transition probabilities for each course, based on statistical predictors (CS II grades probabilistically depend, let us say, on the grade in CS I and on grades in prior attempts on CS II). However, actions taken concurrently can affect each other's outcome probabilities, either synergistically or destructively. For instance, taking a compilers course at the same time as models of computation tends to improve grades in both, because the compilers course motivates interest in regular and context-free grammars, while the theory course reinforces computational techniques. On the other hand, taking two courses with the same schedule of assignments, exams, and projects can be detrimental to grades. This means that optimal planning must consider actions concurrently rather than sequentially. This holds in a fully realized model that takes into account concurrency effects, but it also holds in simpler models. Taking multiple courses concurrently has a different effect than taking them in sequence, and thus concurrent actions should not be modeled sequentially.

Goal States

For each school and each program within that school, requirements define a set of goal states. Individual students have preferences on the types of electives, professors, semester schedules, grade point average, time to graduation, etc. The goal of the academic planner is to optimize the student's total expected utility and keep them moving toward graduation. For a small scale realistic version of the problem, the goal can be to earn an academic minor. A larger scale version is planning for completion of a degree. Besides varying in size, goals also depend on what is valued in the program.

A policy can be developed with a simple reward based on time to the goal or on maximizing grades. With the former, an optimal policy would only be concerned with the student passing courses, while a GPA-driven policy might recommend more time be taken if it ensured better grades. Other criteria can be considered for optimization such as enjoyment (different rewards for different courses or for certain distributions of courses in each semester) or uniformity of number of courses each semester.

Real-World Features of the Academic Domain

The features of the academic advising domain reflect aspects of many real world problems. The three features described above allow the academic domain to capture interesting planning problems. These are:

- planning under uncertainty,
- planning with concurrent actions, and
- planning with multiple reward criteria.

First, the stochastic nature of human endeavors, particularly humans of college age, introduces uncertainty into any plan. For this reason a classical plan does not suffice; rather, a policy is needed.

Secondly, the academic domain models concurrent action planning, because most students take multiple courses each semester. Concurrent action planning has applications from space exploration to pharmaceuticals (Mausam and Weld 2004). An example application for this research could be designing drug regimens. For complex medical conditions, a doctor has a choice of different drug treatments. Different drugs may be taken simultaneously and there are uncertainties associated with the effects and side effects of each drug. For instance, a person might be taking medication for migraines, rheumatoid arthritis, allergies, and digestive disorders. Side effects may lead to the prescription of more drugs which will have their own uncertain effects. This scenario can be modeled as a concurrent action MDP. Solving it could provide huge benefits to doctors prescribing drug regimens and to patients trying to understand the utility of their medicines. But first we need to develop concurrent-action MDP solvers.

Finally, the academic domain has different criteria to optimize and could, therefore, be used for planning with multi-criteria optimization (Perny and Weng 2010). This is relevant to many scenarios in which there are multiple values to optimize, such as balancing risk and reward with investments or prescribing radiation to kill tumors and not harm healthy organs (Ehrgott 2000). The academic domain is structured in a way that allows planners to be built that must account for any or all of these features.

The Academic Domain Model Generator

Problem Structure

Although the common Bayes net representation of MDPs uses dynamic Bayes nets, we represent the temporal aspect of our models implicitly, and draw simple Bayes nets. The nodes of the network represent courses. The values of each course are Not Taken, High, Low, and Fail. Each node has a probability table for those values, conditioned on its parent nodes.

The domain generation module generates the structure and parameters of the network, a goal state, and additional constraints. The structure is based on a standard lattice. In order to create varied instances, we begin with a full lattice and then remove edges.

One feature of our domain instances, which is more typical of computer science programs than of many other programs at our university, is chains of prerequisites. These can be represented as explicit constraints, if the planners can handle constraints, or can be handled implicitly, through the construction of the conditional probability tables (CPTs): students are able to skip prerequisites or to go forward with a poor or failing grade in a prerequisite course, but the expectation of High or Low grades is significantly depressed. For now, we use the latter approach, so edges in the lattice are interpreted both as prerequisites and as parents for CPTs.

The instance generator will choose:

- the size of the lattice;
- where to randomly break the prerequisite chains;
- the conditional probability tables for each node;
- the utility/reward function, and/or goal state.

Prerequisite Hierarchy

Course prerequisites are specified by a prerequisite graph. The graphs we generate are based on lattices with complete connectivity between layers, which are then pruned to generate prerequisite models. A visual representation of the first three full lattices of this sequence are shown in Figures 1 and 2. Edges of the lattice will be pruned until the following conditions are met: (1) all courses have at least one prerequisite and (2) few courses have more than one prerequisite. The resulting graphs will bear strong resemblance to prerequisite hierarchies we have surveyed in actual academic departments.

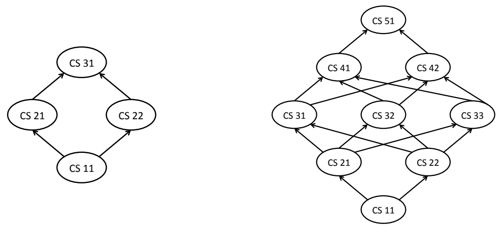

Figure 1: Full prerequisite lattices for small domains

The models we build, based on these lattices, have several interesting properties:

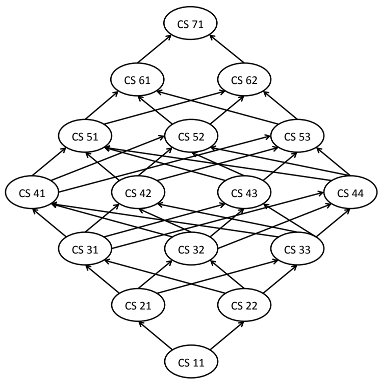

Figure 2: Full prerequisite lattice for a larger domain

1. there is an optimal policy, but even following that policy doesn't guarantee success;
2. the final prerequisite hierarchy specifies a partial ordering over action sequences which must be learned in order to find an optimal policy;
3. the number of *state variables* grows quadratically with the lattice size parameter;
4. the number of *states* grows exponentially in the number of state variables.

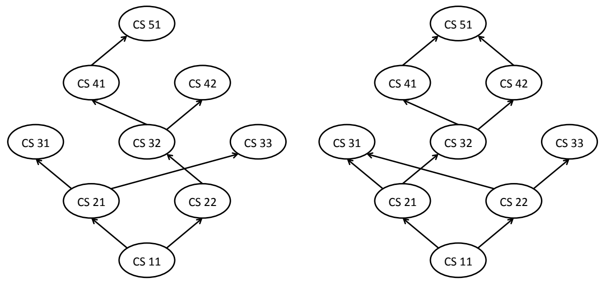

Figure 3: Pruned prerequisite lattices

Figure 3 shows how two lattices with the same nodes can produce two significantly different instances, based on the different edge prunings. A goal state can be specified by giving a list of courses that are required. This can be done either with an explicit set of courses that must be taken plus a total number, or by specifying the number of courses to be taken from each set of courses (i.e., "take 2 courses from the third level of the lattice").

Generating the Domain

The domains and instances can be generated from the specifications of the lattices described in the previous section. The code examples in this paper use the RDDL domain definition language (Sanner). The code describing these lattices will also contain the conditional probabilities of state transitions. This will include depressed probabilities when prerequisites are untaken. The following table shows possible conditional probabilities for the outcome of taking a course.

CS 32	CS 42 outcomes		
grades	H	L	F
H	0.7	0.2	0.1
L	0.3	0.4	0.3
F	0.1	0.15	0.75
NT	0.05	0.1	0.85

A reward function will be specified in the instance problem based upon the definition of the goal state. This function rewards success in required classes and creates penalties for being further from the goal state. Therefore, reaching the goal state is the way to maximize reward. For instance, consider the following reward description for a domain with four possible courses.

```
reward = 3 * [(CS11 == @High)
             + (CS21 == @High)
             + (CS22 == @High)
             + (CS31 == @High)]
        + 1 * [(CS11 == @Low)
             + (CS21 == @Low)
             + (CS22 == @Low)
             + (CS31 == @Low)]
        + 0 * [(CS11 == @Fail)
             + (CS21 == @Fail)
             + (CS22 == @Fail)
             + (CS31 == @Fail)]
        -5 * [(CS11 == @NotTaken)
             + (CS21 == @NotTaken)
             + (CS22 == @NotTaken)
             + (CS31 ==@NotTaken)].
```

Concurrency can be enforced as the means of reaching a solution by specifying a horizon that requires multiple courses taken at a time to reach the goal. The below example specifies a horizon of 8 and allows up to 5 concurrent actions. If the goal state for this domain required 35 courses to be taken, then concurrency is required to reach the goal.

```
instance advising {
        domain = advising;
        init-state {
            CS11 = @NotTaken;
            CS21 = @NotTaken;
            . . .
        };
        max-nondef-actions = 5;
        horizon = 8;
        discount = 0.99;
}
```

Conclusion

This paper has proposed a novel domain, the academic advising domain, for probabilistic planning competitions. This domain is interesting because it involves concurrent actions, true stochasticity, and multi-valued domain variables. It enables the use of constraints, both in terms of prerequisites and potential mutual exclusion constraints. In addition, it opens the possibility of explicitly considering the planning problem as a multi-criteria optimization problem. Introducing these elements to planning competitions will raise new challenges in the IPPC. It is our hope that this domain will build upon the success of previous competition domains to push the state of the art in planning research.

Acknowledgements This work is partially supported by by NSF EAGER grant CCF-1049360. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Dodson, T.; Mattei, N.; and Goldsmith, J. 2011. Natural language argumentation interface for explanation generation in Markov decision processes. In *Proc. Algorithmic Decision Theory*. also appeared in the EXaCT workshop at IJCAI 2011.

Ehrgott, M. 2000. *Multicriteria Optimization*. Berlin: Springer.

Guerin, J. T., and Goldsmith, J. 2011. Constructing a dynamic Bayes net model of academic advising. In *Proc. Bayesian Modelling Applications Workshop, UAI*.

Khan, O. Z.; Poupart, P.; and Black, J. 2009. Minimal sufficient explanations for factored Markov decision processes. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

Little, I., and Thiébaux, S. 2007. Probabilistic planning vs. replanning. In *ICAPS Workshop on IPC: Past, Present and Future*.

Mausam, and Weld, D. S. 2004. Solving concurrent Markov decision processes. In *National Conference on Artificial Intelligence*. AAAI.

Perny, P., and Weng, P. 2010. On finding compromise solutions in multiobjective markov decision processes. In *European Conference on Artificial Intelligence Multidisciplinary Workshop on Advances in Preference Handling*, 55–60.

Sanner, S. Relational dynamic influence diagram language (rddl): Language description.

Sanner, S. 2008. How to spice up your planning under uncertainty research life. In *Workshop on a Reality Check for Planning and Scheduling Under Uncertainty (ICAPS-08)*.