Planning with graded fluents and actions

Marta Cialdea, Carla Limongelli, Andrea Orlandini, Valentina Poggioni
Dipartimento Informatica e Automazione
Università degli Studi Roma Tre
Via della Vasca Navale, 79 - 00146 Roma
{cialdea,limongel,orlandin,poggioni}@dia.uniroma3.it

Abstract

This work can be seen as a first approach to a new planning model that takes into account the possibility to express actions and fluents with non-boolean values. According to this model, a planning problem is defined using both graded (multi-valued) and classical (boolean) fluents. Moreover, actions that can have different application degrees can be defined. In this work a PDDL extension allowing to describe such new problems is proposed and a planning algorithm for such problems is presented.

1 Introduction

In the last years, extensions of the classical planning model have been investigated, such as temporal models, conditional and contingent models, probabilistic models and other mixed models (for example [Hoffmann, 2002; Petrick and Bacchus, 2002; Bonet and Geffner, 2003; Chien and al., 2000]). But, to the best of our knowledge, a feature of the classical model has never been modified: the use of boolean expressions to describe fluents and actions in the domains. This is often too restrictive in order to represent realistic domains because the world is not black and white (i.e. true or false) but it has a lot of colors (i.e. intermediate truth-values, or "degrees of truth"). A different approach to planning that takes into account a numerical "state" associated to a fluent is proposed in the probabilistic planning model, but, in this case, "numbers" represent our knowledge or uncertainty about the state or the success of an action, while the real world is always two-valued.

This work can be seen as a first approach to a new planning model that takes into account the possibility to express both actions and fluents with non-boolean values. According to this model, a planning problem is defined using both graded (multi-valued) and classical (boolean) fluents; moreover actions having different application degrees can be defined (i.e. actions having adjustable intensity and that can affect fluents proportionally to how much they are applied). In such a way, also the efficiency of actions can be easily represented.

The work defines an extension of the planning language PDDL that allows us to define planning domains and planning problems having graded fluents and graded actions, and a solving algorithm for such problems, where first a candidate plan with partially instantiated actions is constructed, then the plan applicability and correctness is verified by means of a translation into a MIP (Mixed Integer Programming) problem and finally, if a solution of the MIP problem exists, a complete instantiation is made and the solution plan is found.

2 The Planning Language

The planning language proposed in this paper is based on standard PDDL. It provides two kinds of actions and fluents, representing both classical $boolean$ fluents and actions, and $graded$ fluents and actions. Both actions and fluents have an additional argument denoting their "degree of truth": in a fluent it means "how much" the predicate is true (0 means that it is false, 1 that it is true) and in an action it means "how much" the action is applied (0 means that it is not applied at all, 1 means that it is applied with the maximum efficiency).

The type of such terms is declared as a new type *degree* in the PDDL domain definition. If the fluent is $boolean$ then its truth-value is a natural number in $\{0, 1\}$ (as in the classical model), otherwise, if the fluent is $graded$, its truth-value is a real number in $[0, 1]$. The difference between the two kinds of fluents is declared in the domain description. There are different declaration sections for boolean and graded predicates.

The additional parameter in an action denotes the degree of application of the action. Again, the value of this parameter in *boolean actions* belongs to $\{0, 1\}$, while it ranges in $[0, 1]$ in *graded actions*. Actions may also have other additional parameters referring to fluent degrees. A specific section in action declaration allows one to associate fluents to degree parameters. Such variables maybe used both in preconditions and effects: action preconditions may contain inequality constraints over fluent degrees and effects can be described by means of expressions that define the new "degree of truth" of a fluent as a linear combination of previous fluent values and the application degree of the action.

The choice to include the degrees among the action parameters does not increase the complexity of the operators because, as explained in Section3, such parameters are not instantiated during the solution search phase.

The definition of a graded planning problem may contain an *objective function*, that is a linear function of the action application degrees that must be minimized or maximized when looking for action degree values satisfying a given partially instantiated plan. For example, it can be used in or-

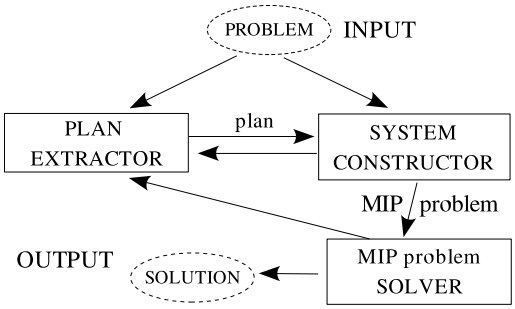

Figure 1: The system architecture

der to minimize the cost of the extracted candidate plan giving different operator costs. It is defined in a new section `:objective_function(...)` in the problem description.

3 System Architecture

The system takes in input a graded planning problem $(I, G, f, \mathcal{O})$ and if a solution exists it returns a graded solution plan P, otherwise,if it terminates it returns a *NoSolution* message. The system architecture is presented in Fig.1: it is composed by three modules, the *plan extractor*, the *system constructor*, the *MIP solver*.

The *plan extractor* synthesizes a "candidate plan" $P = (A_1(x_1), A_2(x_2), \ldots, A_m(x_m))$ where actions are partially instantiated, i.e. where all application and fluent degrees are unbound but only the variables x_i standing for the degree of application of A_i are in evidence. It implements a simple backward algorithm with heuristic functions that solves relaxed problems. Then the *system constructor* computes the world evolution using these actions (the resulting states depend on the action application degrees) and reduces the verification and full instantiation of the plan to a MIP problem; it builds the MIP problem corresponding to a candidate plan and passes it to the *MIP solver*. During the system construction phase some conditions in the action preconditions are directly checked and if they are not satisfied the module fails and another candidate plan must be extracted. In this case the information about which action causes failure is used and another candidate plan is constructed replanning from this point. Finally the *MIP solver* computes a solution of the generated problem; if a solution exists it is a set of real and/or integer values $\vec{g} = (g_1, \ldots, g_m)$ and the plan $P = (A_1(g_1), \ldots, A_m(g_m))$ is the graded solution plan of the given problem, otherwise a new candidate plan is extracted and a new MIP problem is generated.

4 Conclusions and related works

In this work a language and a model of planning with graded fluents and actions are presented. A prototype of the system has been developed.

To the best of our knowledge this is the first system able to manage non boolean fluents and actions. Recent works have proposed languages [Fox and Long, 2003; Giunchiglia *et al.*, 2004; Lee and Lifschitz, 2003] and systems (for example [Koehler, 1998; Baioletti *et al.*, 2003; Hoffmann, 2002; Haslum and Geffner, 2000]) that can handle numerical values. Degrees can be represented in PDDL 2.1, but only if their values range on finite sets. Graded fluents could be represented in PDDL 2.1 by means of numerical fluents (functions) but respecting some more restriction w.r.t. what is done in this work. However, graded actions are not representable at all.

At the moment we are working in two main directions. First of all an algorithm for intelligent backtracking when the *MIP solver* fails and an improvement of the backtracking phase when the *system constructor* fails are under investigation. Moreover a set of graded domains and graded problems is under construction in order to carry out a wide set of experiments. The second research direction is theoretical: an extension of the algorithm to planning under uncertainty on the initial state is straightforward, introducing variables for fluent degrees in the world construction. Moreover, we are investigating the possibility of representing and treating vague (fuzzy) fluents, using intervals or sets to represent fluent degrees.

References

[Baioletti *et al.*, 2003] M. Baioletti, A. Milani, and V. Poggioni. Planning with fuzzy resources. In *AI*IA 2003: Advances in Artificial Intelligence. In LNAI 2829, 336-348*, 2003.

[Bonet and Geffner, 2003] B. Bonet and H. Geffner. Faster heuristic search algorithms for planning with uncertainty and full feedback. In *Proc. of IJCAI 2003*, 2003.

[Chien and al., 2000] S. Chien and al. ASPEN: Automated planning and scheduling for space mission operation. In *Proc. of SpaceOps 2000, Colorado (USA)*, 2000.

[Fox and Long, 2003] M. Fox and D. Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *JAIR*, 20:61–124, 2003.

[Giunchiglia *et al.*, 2004] E. Giunchiglia, J. Lee, V. Lifschitz, McCain N., and H. Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153:49–104, 2004.

[Haslum and Geffner, 2000] P. Haslum and H. Geffner. Heuristic planning with resources. In *Proc. of ECAI-00*, 2000.

[Hoffmann, 2002] J. Hoffmann. Extending FF to numerical state variables. In *Proc. of ECAI 2002*, 2002.

[Koehler, 1998] J. Koehler. Planning under resource constraints. In *Proc. of ECAI-98*, 1998.

[Lee and Lifschitz, 2003] J. Lee and V. Lifschitz. Describing additive fluents in action language ⌋+. In *Proc. of IJCAI 2003*, 2003.

[Petrick and Bacchus, 2002] R.P.A. Petrick and F. Bacchus. A knowledge-based approach to planning with incomplete information and sensing. In *Proc. of AIPS 2002*, 2002.