Planning with Preferences and Trajectory Constraints by Integer Programming

Menkes van den Briel
Department of Industrial Engineering
Arizona State University
Tempe AZ, 85287-8809
menkes@asu.edu

Abstract

The focus of my research is on the formulation and analysis of mathematical programming techniques in automated planning. This extended abstract provides a brief overview of the paper that will be presented at the ICAPS Workshop on Planning with Preferences and Trajectory Constraints. A synopsis of some of my future research plans is given at the end.

Introduction

Given the recent success of integer programming approaches to automated planning (van den Briel, Vossen, & Kambhampati 2005), I believe that these approaches are a good avenue to explore further both because of the recent improvements, and the fact that with preferences, planning becomes an optimization problem, which integer programming is naturally equipped to handle.

Preferences and trajectory constraints are two new language features in PDDL3.0 that can be used to express hard and soft constraints on plan trajectories, and that can be used to differentiate between hard and soft goals. Hard constraints and goals define a set of conditions that must be satisfied by any solution plan, while soft constraints and goals define a set of conditions that merely affect solution quality.

In particular, preferences assume a choice between alternatives and the possibility to rank or order these alternatives. In PDDL3.0, preferences can be defined on states, on action preconditions, on trajectory constraints, or on some combination of these. Since preferences may or may not be satisfied for a plan to be valid they impose soft constraints or goals on the planning problem. Trajectory constraints, on the other hand, define a set of conditions that must be met throughout the execution of the plan. They can be used to express control knowledge or simply describe restrictions of the planning domain. Since trajectory constraints define necessary conditions for a plan to be valid (except in the case where the trajectory constraint is a preference) they impose hard constraints or goals on the planning problem.

Neither preferences nor trajectory constraints have yet gotten a lot of attention from the planning community, but the importance of solution quality and the efficient handling of hard and soft constraints and goals has increasingly been addressed by some recent works.

Planning with preferences is closely related to oversubscription planning. In oversubscription planning goals are treated as soft goals as there are not enough resources to satisfy all of them. This problem has been investigated by Smith (2004) and further investigated by several other works.

Preferences, however, are more general than soft goals as they also include soft constraints. Son and Pontelli (2004) describe a language for specifying preferences in planning problems using logic programming. Their language can express a wide variety of preferences, including both soft goals and soft constraints, but it seems that it has not been used for testing yet. Empirical results for planning with preferences are provided by Rabideau, Engelhardt and Chien (2000) and Brafman and Chernyavsky (2005). Rabideau, Engelhardt and Chien describe an optimization framework for the ASPEN planning system, and Brafman and Chernyavsky describe a constraint based approach for the GP-CSP planning system.

Planning with trajectory constraints is closely related to reasoning about temporal control knowledge and temporally extended goals, which are discussed by Ghallab Laruelle (1994) and Muscettola 1994. Edelkamp (2005) handles trajectory constraints by converting a PDDL3.0 description into a PDDL2.2 description and then using a heuristic search planner.

In this extended abstract I will show a few examples of how to express preferences and trajectory constraints by linear constraints over 0-1 variables. These constraints are then to be added to the integer programming formulation of the planning problem after which the model is solved. Currently, I'm in the process of incorporating these constraints in the integer programming formulations described in van den Briel, Vossen, & Kambhampati (2005).

Simple Preferences

Simple preferences are preferences that appear in the goal or that appear in the preconditions of an action. Goal preferences can be violated at most once (at the end of the plan), whereas precondition preferences can be violated multiple times (each time the corresponding action is executed).

For each goal preference in the planning problem we introduce a 0-1 variable p, where $p = 1$, if the goal preference is violated and, $p = 0$ if the goal preference is satisfied. Similarly, for each precondition preference for action a at step t ($1 \leq t \leq T$) we introduce a 0-1 variable $p_{a,t}$, where $p_{a,t} = 1$, if the precondition preference is violated for action a at step t and, $p_{a,t} = 0$ if the precondition preference is satisfied for action a at step t. This way all violations can be counted for separately and given different costs in the objective function of the formulation.

Constraints for goal and precondition preferences are easily modeled by integer programming. There are only finitely many operators in PDDL3.0, including some standard operators like `or`, `and`, and `imply`, which can all be represented by one or more linear constraints.

Examples

In the examples we will use variables $x_{a,t}$ to denote the execution of an action a at step t, and use variables $y_{f,t}$ to denote the truth value of a fluent f at step t. This is slightly different from the notation and variables used in the formulations by van den Briel, Vossen, and Kambhampati 2005, but for explanation purposes we think it is more obvious this way.

In PDDL3.0, the goal preference p_1 "We would like that person1 is at city2" is expressed as follows.

```
(:goal (and (preference p1
  (at person1 city2))))
```

The inequality corresponding to preference p_1 is given by:

$$p_1 \geq 1 - y_{\text{at person1 city2},T} \tag{1}$$

Thus preference p_1 is violated ($p_1 = 1$) if person1 is not at city2 at the end of the plan ($y_{\text{at person1 city2},T} = 0$).

The goal preference p_2 "We would like that person1 or person2 is at city2" is expressed as follows.

```
(:goal (and (preference p2 (or
  (at person1 city2) (at person2 city2)))))
```

The inequality corresponding to preference p_2 is given by:

$$p_2 \geq 1 - y_{\text{at person1 city2},T} - y_{\text{at person2 city2},T} \tag{2}$$

Now, preference p_2 is violated if neither person1 nor person2 is at city2 at the end of the plan. Preference p_2 is satisfied when either or both person1 and person2 are at city2 at the end of the plan.

The goal preference p_3 "We would like that person2 is at city1 if person1 is at city1" is expressed as follows.

```
(:goal (and (preference p3 (imply
  (at person1 city1) (at person2 city1)))))
```

The inequality corresponding to preference p_3 is given by:

$$p_3 \geq y_{\text{at person1 city1},T} - y_{\text{at person2 city1},T} \tag{3}$$

So preference p_3 is violated if person2 is not at city1 while person1 is.

Preferences over preconditions are different from goal preferences as they depend on both the execution of an action and on the state of the precondition of that action. Moreover, a precondition preference is defined for each plan step t, where $1 \leq t \leq T$. In PDDL3.0, the precondition preference $p_{4,\text{fly}?a?c1?c2,t}$ "We would like that some person is in the aircraft" whenever we fly aircraft $?a$ from city $?c1$ to city $?c2$ is expressed as follows:

```
(:action fly
 :parameters (?a - aircraft ?c1 ?c2 - city)
 :precondition (and (at ?a ?c1)
   (preference p4
     (exists (?p - person) (in ?p ?a))))
 :effect (and (not (at ?a ?c1))
   (at ?a ?c2)))
```

The inequalities corresponding to each ground fly ?a ?c1 ?c2 action is given by:

$$p_{4,\text{fly}?a?c1?c2,t} \geq x_{\text{fly ?a ?c1 ?c2,t}} - \sum_{?p} y_{\text{in ?p ?a},t} \qquad \forall 1 \leq t \leq T \tag{4}$$

Thus, preference $p_{4,\text{fly}?a?c1?c2,t}$ is violated at step t if we fly aircraft $?a$ from city $?c1$ to city $?c2$ at step t ($x_{\text{fly ?a ?c1 ?c2,t}} = 1$) without having any passenger $?p$ onboard at step t ($y_{\text{in ?p ?a},t} = 0$, for each $?p$).

Qualitative Preferences

In propositional planning, qualitative preferences include trajectory constraints and preferences over trajectory constraints none of which involve numbers. Given the space limitations we will mainly concentrate on the trajectory constraints here that use the new modal operators of PDDL3.0 in this section.

There is a general rule of thumb for the operators `forall` and `always`. `forall` indicates that the trajectory constraint must hold for each object to which it is referring to. For example, `forall (?b block)` means that the trajectory must hold for each instantiation of $?b$, thus we generate the trajectory constraint for all blocks $?b$. `always` in propositional planning is equivalent to saying for all t, thus we generate the trajectory constraint for all t where $1 \leq t \leq T$.

Constraints for trajectories are easily modeled by integer programming through observing the different operators carefully. It is often the case, that the trajectory constraint simply represent one of the standard relationships described earlier in this paper.

Examples

In PDDL3.0 the trajectory constraint "A fragile block can never have something above it" is expressed as follows.

```
(:constraints (and (always (forall (?b block)
  (implies (fragile ?b) (clear ?b))))))
```

The inequality corresponding to this trajectory constraint corresponds to the relation that fragile *implies* clear for all blocks $?b$, for all steps t, where $1 \leq t \leq T$. It is given by:

$$y_{\text{fragile ?b},t} - y_{\text{clear ?b},t} \leq 0 \quad \forall ?b, 1 \leq t \leq T \tag{5}$$

The trajectory constraint "Each block should be picked up at most once" which is expressed as follows.

```
(:constraints (and (forall (?b block)
  (at-most-once (holding ?b)))))
```

It translates to an *at most once* relation for all blocks $?b$ and is given by:

$$y_{\text{holding ?b},0} + \sum_{a \in A, 1 \leq t \leq T: \text{holding ?b} \in ADD(a)} x_t^a \leq 1 \quad \forall ?b \tag{6}$$

Likewise the trajectory constraint "Each block should be picked up at least once" is expressed as follows.

```
(:constraints (and (forall (?b block)
  (sometime (holding ?b)))))
```

This translates to a *sometime* relation for all blocks $?b$ and is given by:

$$\sum_t y_{\text{holding ?b},t} \geq 1 \quad \forall ?b \tag{7}$$

Continuing in the same way, the trajectory constraint "A truck can visit city1 only if it has visited city2 sometime before" is expressed in PDDL3.0 as follows.

```
(:constraints (and (forall (?t truck)
  (sometime-before
    (at ?t city1) (at ?t city2)))))
```

The corresponding inequality describes a *sometime-before* relationship for all trucks $?t$ and is given by:

$$\sum_{1 \leq s < t} y_{\text{at ?t city2},s} \geq y_{\text{at ?t city1},t} \quad \forall ?t, 1 \leq t \leq T \tag{8}$$

More examples can be presented, but it seems enough to bring the point across that integer programming provides a natural framework for modeling propositional planning with preferences and trajectory constraints.

Conclusions and Future Work

This extended abstract shows a few examples of how to model preferences and trajectory constraints by integer programming manually. The main challenge is to automatically generate these constraints and add them to the integer programming formulation of the planning problem. Especially, generating constraints for complicated instances of preferences and trajectory constraints that contain nested expressions can be tricky, but is feasible.

An interesting analysis for future work would be to compare the performance of the integer programming formulations that use preferences and trajectory constraints as side constraints (as shown in the above examples) with integer programming formulations that handle preferences and trajectory constraints which are compiled down into PDDL2.2.

The general focus of my future research is to extend and improve the current integer programming formulations for automated planning, and to apply integer programming techniques to a broader range of planning problems, including resource planning and temporal planning. Techniques like branch-and-cut and branch-and-price will be at the base of most of these extensions.

References

Brafman, R., and Chernyavsky, Y. 2005. Planning with goal preferences and constraints. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS)*, 182–191.

Edelkamp, S. 2005. Efficient planning with state trajectory constraints. In Sauer, J., ed., *Proceedings Workshop Planen, Scheduling und Konfigurieren / Entwerfen*, 89–99.

Ghallab, M., and Laruelle, H. 1994. Representation and control in IxTeT, a temporal planner. In *Proceedings of the 2nd International Conference on Artificial Intelligence Planning and Scheduling (AIPS)*, 62–67.

Muscettola, N. 1994. *Intelligent Scheduling*. Morgan Kaufmann. chapter HSTS: Integrating planning and scheduling, 169–212.

Rabideau, G.; Engelhardt, B.; and Chien, S. 2000. Using generic preferences to incrementally improve plan quality. In *Proceedings of the 2nd NASA International Workshop on Planning and Scheduling for Space*, 11–16.

Smith, D. 2004. Choosing objectives in over-subscription planning. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS)*, 393–401.

Son, T., and Pontelli, E. 2004. Planning with preferences unsing logic programming. In *Proceedings of the 7th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 247–260.

van den Briel, M.; Vossen, T.; and Kambhampati, S. 2005. Reviving integer programming approaches for ai planning: A branch-and-cut framework. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS)*, 310–319.