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

Subbarao Kambhampati
Department of Computer Science
Arizona State University
Tempe AZ, 85287-8809
rao@asu.edu

Thomas Vossen
Leeds School of Business
University of Colorado at Boulder
Boulder CO, 80309-0419
vossen@colorado.edu

Abstract

We present an integer programming approach for handling preferences and trajectory constraints in planning. Our main aim is to illustrate through examples how simple it is to express preferences and trajectory constraints by linear constraints over 0-1 variables. We are currently in the process of incorporating these constraints in the context of efficient integer programming encodings that we developed recently.

Introduction

Given the recent success of integer programming approaches to automated planning (van den Briel, Vossen, & Kambhampati 2005), we 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 explored 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. 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 paper we show numerous examples of how to express preferences and trajectory constraints by linear constraints over 0-1 variables. These constraints would need to be added to the integer programming formulation of the planning problem. Currently, we are still in the process of incorporating these constraints in the formulations described by van den Briel, Vossen and Kambhampati (2005).

The organization of this this paper is as follows. The next section is concerned with the formulation of integer optimization problems, that is, how to translate a verbal description of a problem into mathematical state-

ment. In particular, we will show how we can use 0-1 variables to model different relations between events. We then show various planning examples for both simple preferences and qualitative preferences, and some concluding remarks will be given at the end.

Modeling with 0-1 Variables

Integer programming is a powerful and natural modeling framework for solving optimization problems involving integer decision variables. The case where the integer variables are restricted to be 0 or 1 are referred to as 0-1 programming problems or binary integer programming problems. In mathematical programming, 0-1 variables are commonly used to represent binary choice, where binary choice is simply a choice between two things. For example, consider the problem of deciding whether an event should or should not occur. This decision can be modeled by a binary variable x, where $x = 1$, if the event occurs and, $x = 0$, if the event does not occur. Depending on the problem being considered, the event itself could be almost anything. For example, in scheduling, an event x could represent whether some job should be scheduled before another job.

Here we show some standard relationships between events and how they can be modeled by 0-1 variables.

- The relation that *at most one* of a set J of events is allowed to occur is represented by a packing constraint, $\sum_{j \in J} x_j \leq 1$.
- The relation that *at least one* of a set J of events is allowed to occur is represented by a cover constraint, $\sum_{j \in J} x_j \geq 1$.
- The relation that *exactly one* of a set J of events is allowed to occur is represented by a partitioning constraint, $\sum_{j \in J} x_j = 1$.
- The relation that neither or both events 1 and 2 must occur, that is, event 1 *equals* event 2, is represented by the linear equality $x_2 - x_1 = 0$.
- The relation that event 2 can occur only if event 1 occurs, that is, event 2 *implies* event 1, is represented by the linear inequality $x_2 - x_1 \leq 0$.

Sometimes, the occurrence of an event is limited to a set of pre-specified time periods $1 \leq t \leq T$. The decision whether an event should or should not occur at time period t can can be modeled by a time-indexed binary variable x_t, where $x_t = 1$, if the event occurs at time period t and, $x_t = 0$, if the event does not occur at time period t. For example, in planning, an event x_t could represent the execution of an action or the truth value of a proposition at a specific plan step.

Here we show some standard relationships between time dependent events and how they can be modeled by time-indexed 0-1 variables.

- The relation that event 1 may occur *at most once* during the time horizon, is represented by the linear inequality $\sum_t x_{1,t} \leq 1$.
- The relation that event 1 must occur *sometime* during the time horizon, is represented by the linear inequality $\sum_t x_{1,t} \geq 1$.
- The relation that if event 1 occurs, event 2 must occur *sometime-before* event 1, is represented by the linear inequalities $x_{1,t} \leq \sum_{1 \leq s < t} x_{2,s}$ for all $1 \leq t \leq T$.
- The relation that if event 1 occurs, event 2 must occur *sometime-after* event 1, is represented by the linear inequalities $x_{1,t} \leq \sum_{t \leq s \leq T} x_{2,s}$ for all $1 < t \leq T$.
- The relation that event 1 must *always* occur is represented by the linear equalities $x_{1,t} = 1$ for all $1 \leq t \leq T$.
- The relation that event 1 must occur *at the end* of the time horizon is represented by the linear equality $x_{1,T} = 1$.

Note that most of these standard relationships coincide with the new modal operators: `at-most-once`, `sometime`, `sometime-before`, `sometime-after`, `always`, `at end`, in PDDL3.0. Even though there some differences in semantics, this suggests that modeling these modal operators and the preferences and trajectory constraints that are expressed by them through integer programming should be rather straightforward.

In the next two sections we give various examples of how to model preferences and trajectory constraints by linear constraints over 0-1 variables. The examples are all borrowed from the International Planning Competition resources [1].

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 $y_{f,t}$

[1] http://zeus.ing.unibs.it/ipc-5/

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 it provides a concise representation of the resulting constraints.

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:

$$p_1 \geq 1 - y_{\text{at person1 city2},T} \quad (1)$$

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:

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

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:

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

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

The goal preference p_4 "We would like that person3 and person4 are at city3" is expressed as follows.

```
(:goal (and (preference p4 (and
  (at person3 city3) (at person4 city3)))))
```

Note that preference p_4 is very similar to preference p_1 with the exception that p_4 is defined over a conjunction of fluents. For each fluent in the conjunction we will state a separate constraint, thus the inequalities corresponding to preference p_4 are given by:

$$p_4 \geq 1 - y_{\text{at person3 city3},T} \quad (4)$$

$$p_4 \geq 1 - y_{\text{at person4 city3},T} \quad (5)$$

Now, preference p_4 is violated if either or both person3 and person4 are not at city3 at the end of the plan.

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 $p_{5,\text{fly}?a?c1?c2,t}$ "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 p5
     (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:

$$p_{5,\text{fly}?a?c1?c2,t} \geq x_{\text{fly ?a ?c1 ?c2,t}} - \sum_{?p} y_{\text{in ?p ?a},t} \quad \forall 1 \leq t \leq T \quad (6)$$

Thus, preference $p_{5,\text{fly}?a?c1?c2,t}$ 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:

$$y_{\text{fragile ?b},t} - y_{\text{clear ?b},t} \leq 0 \quad \forall ?b, 1 \leq t \leq T \qquad (7)$$

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:

$$y_{\text{holding ?b},0} + \sum_{a \in A, 1 \leq t \leq T : \text{holding ?b} \in ADD(a)} x_t^a \leq 1 \quad \forall ?b \quad (8)$$

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:

$$\sum_t y_{\text{holding ?b},t} \geq 1 \quad \forall ?b \qquad (9)$$

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:

$$\sum_{1 \leq s < t} y_{\text{at ?t city2},s} \geq y_{\text{at ?t city1},t} \quad \forall ?t, 1 \leq t \leq T \quad (10)$$

Similarly the trajectory constraint "If a taxi has been used and it is at the depot, then it has to be cleaned."

```
(:constraints (and (forall (?t - taxi)
  (sometime-after (and (at ?t depot) (used ?t))
    (clean ?t)))))
```

This translates to a *sometime-after* relationship for all taxis $?t$. Note, however, that this trajectory constraint has two conditions, which if satisfied, require that taxi $?t$ is to be cleaned. The inequality corresponding to this trajectory constraint is given by:

$$y_{\text{at ?t depot},t} + y_{\text{used ?t},t} - 1 \leq \sum_{t \leq s \leq T} y_{\text{clean ?t},s} \quad \forall ?t, 1 \leq t \leq T \qquad (11)$$

Now, if taxi $?t$ is at the depot at step t ($y_{\text{at ?t depot},t} = 1$) and if it has been used ($y_{\text{used ?t},t} = 1$), then it must be cleaned sometime after step t ($\sum_{t \leq s \leq T} y_{\text{clean ?t},s} \geq 1$).

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

Conclusions

We have shown numerous examples of how to model preferences and trajectory constraints by integer programming. 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. Even though we haven't had the time to implement this yet, we believe this can be done.

An interesting analysis for future work would be to see the impact on performance when preferences and trajectory constraints are added to integer programming formulation of the planning problem. Also we would like to compare the performance of the integer programming formulations that use preferences and trajectory constraints as side constraints (as shown in the examples in this paper) with integer programming formulations that handle preferences and trajectory constraints which are compiled down into PDDL2.2.

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.

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 using 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.