Planning for PDDL3- An OCSP Based Approach

Bharat Ranjan Kavuluri Naresh Babu Saladi Deepak Khemani

bharat@cs.iitm.ernet.in snaresh@cse.iitm.ernet.in khemani@iitm.ac.in

Department of Computer Science and Engineering

Indian Institute of Technology Madras

Chennai-36, India

Abstract

Recent research in AI Planning is focused on improving the quality of the generated plans. PDDL3 incorporates hard and soft constraints on goals and the plan trajectory. Plan trajectory constraints are conditions that need to be satisfied at various stages of the plan. Soft goals are goals, which need not necessarily be achieved but are desirable. An extension of Constraint Satisfaction Problem, called Optimal Constraint Satisfaction Problem (OCSP) has allowance for defining soft constraints and objective functions. Each soft constraint is associated with a penalty, which will be levied if the constraint is violated. The OCSP solver arrives at a solution that minimizes the total penalty (Objective function) and satisfies all hard constraints. In this paper, an OCSP encoding for the classical planning problems with plan trajectory constraints, soft and hard goals is proposed. Modal operators associated with hard goals and hard plan trajectory constraints are handled by preprocessing and imposing new constraints over the existing GP-CSP encoding. A new encoding for each of the modal operators associated with the soft goals and soft plan trajectory constraints is proposed. Also, a way of encoding conditional goal preference constraints into OCSP is discussed. Based on this research, we intend to submit a planner for the coming planning competition-IPC2006.

Introduction and Background

Current emphasis in the planning community is on planning in richer domains. The latest version of the planning domain description language, PDDL3 [1] has constructs that depict soft and hard constraints on plan trajectory and soft and hard goals. The focus is on generating plans that satisfy all the hard constraints and minimize the penalty due to violation of soft constraints.

Definition of valid plan (PDDL3)[1]*: Given a domain D, an initial state S_o and a goal G, a plan Π generates the trajectory of states $\langle(S_0,0), (S_1,t_1) \ldots(S_n, t_n)\rangle$. Π is valid if $<(S_0,0), (S_1,t_1) \ldots (S_n, t_n)> \models G$.*

Let ϕ and ψ be atomic formulae over the predicates of the planning problem, plus equalities and inequalities between numeric terms. Let *t* be any real constant value. The interpretation of various modal operators for a valid plan trajectory $<(S_0, 0), (S_1, t_1), ..., (S_n, t_n)>$ is as shown in the table 1.

The planning graph representation in Graphplan [2] was used to automatically generate a CSP [8] encoding in the GP-CSP planner [5]. In the CSP community, very recent research is focused on extending the constraint framework from satisfaction to optimization by adding soft constraints. Several solvers were extended to accommodate these constraints [3,6]. Brafman and Chernyavsky propose a method, which deals with conditional and unconditional goal preferences [6].

Modal operator	Condition to be satisfied
at end ϕ	$S_n \models \phi$
ϕ	$S_n \models \phi$
always ϕ	$\forall i : 0 \le i \le n, S_i \models \phi$
sometime ϕ	$\exists i : 0 \le i \le n, S_i \models \phi$
within t ϕ	$\exists i : 0 \le i \le n, S_i \models \phi$ and $t_i \le t$
at-most-once ϕ	$\forall i : 0 \le i \le n$, if $S_i \models \phi$ then $\exists j : j \ge i, \forall k : i \le k \le j, S_k \models \phi$ and $\forall k : k > j, S_k \models \neg\phi$
sometime-after ϕ ψ	$\forall i$: if $S_i \models \phi$ then $\exists j : i \le j \le n, S_j \models \psi$
sometime-before ϕ ψ	$\forall i$: if $S_i \models \phi$ then $\exists j : 0 \le j < i, S_j \models \psi$
always-within t ϕ ψ	$\forall i$: if $S_i \models \phi$ then $\exists j : i \le j \le n, S_j \models \psi$ and $t_j - t_i \le t$
hold-during t_1 t_2 ϕ	$\forall i : t_1 \le i < t_2, S_i \models \phi$
hold-after t ϕ	$\forall i : t < i \le n, S_i \models \phi$

Table 1: PDDL3 Modal operators and their interpretation[1]

Optimal Constraint Satisfaction Problem

A more recent extension to the CSP, Optimal Constraint Satisfaction Problem (OCSP), introduces an objective function into the constraint satisfaction problem (for solving optimization problems). Each constraint in OCSP is associated with a decision variable whose domain is {true, false}. Assignment of false value to this decision variable incurs a penalty, which is the cost of not satisfying the constraint. The formal definition of OCSP is as follows:

An optimal CSP (OCSP) [9,4] *consists of a classical CSP (X, D, C), with valuation structure (E, $\preceq$, $\oplus$, $\bot$, $\top$), and a set U of unary functions $u_j : y_j \rightarrow E$ defined over a subset $Y \subseteq X$ of the variables. The variables in Y are called decision variables, and the variables in X/Y (X not in Y) are called non-decision variables.*

A Valuation structure (E, $\preceq$, $\oplus$, $\bot$, $\top$) is defined as a tuple where E is a set of valuations, totally ordered by $\preceq$ with a minimum element $\bot \in$ E and a maximum element $\top \in$ E, and $\oplus$ is an associative, commutative, and monotonic binary operation with identity element $\bot$ and absorbing element $\top$. Here, the set of valuations, E expresses different levels of constraint violation. $\bot$ means satisfaction and $\top$ means unacceptable violation. A

constraint is hard if its valuation is either $\bot$ or $\top$. A soft constraint has valuation between $\bot$ and $\top$. A CSP augmented with a valuation structure is called valued CSP (VCSP) [7]. An OCSP is an extension of VCSP. Here each soft constraint c_j is defined on a set of non-decision variables and a decision variable. A solution (optimal solution) to OCSP is an assignment to Y with minimum valuation such that there exists a value assignment to all non-decision variables X/Y that satisfies all constraints in the OCSP.

We propose methods to encode the hard and soft constraints of PDDL3 for a classical planning problem. The hard constraints are accommodated by pre-processing and including additional constraints into GP-CSP encoding. We base our approach to accommodate soft constraints on OCSP. We give out equivalent OCSP constraints for each of the modal operators of PDDL3 and express the PDDL3 soft constraints in terms of OCSP soft constraints. Like the GP-CSP planner, we base our approach on the Planning Graph. We present analysis of our encoding.

Encoding PDDL3 Modal Operators into OCSP

Terminology

Let

- $P = \{P_1, P_2, \ldots, P_n\}$ be the set of Ground Predicates.
- P_j^i signifies that P_j is present at predicate level *i* in the planning graph.
- P_j^i is the variable name associated with the predicate P_j at level *i*.
- ϕ is the null value introduced in the domain of every variable in the OCSP.

OCSP Encoding for PDDL3

Assume that the planning graph is being expanded up to '*k*' levels. We deal with different modal operators concerned with goal and trajectory constraints as shown in table 2.

Modal Operator	Hard Constraint		Soft Constraint
	Pre-processing / Goal test modification	Constraints	
At End <P_i>	-	$P_i^k \neq \phi$	$\neg((d=\text{true}) \oplus (P_i^k \neq \phi))$, penalty= *p* where *d* is the decision variable associated with this soft constraint
Always <P_i>	Delete the generic or ground actions which have this predicate in their delete list	-	$\neg((d_j=\text{true}) \oplus (P_i^j \neq \phi))$, penalty= *p* where, $j=1,2,..k$ and d_j is the decision variable associated with the j^{th} soft constraint
Sometime <P_i>	For each predicate P_i which has a **sometime** modal operator associated with it, add a dummy predicate D_i to the add effects of all actions that add P_i. Include D_i into the final goal set.	$V_j (P_i^j \neq \phi)$ where $j=l,l+1..,k$ *l* is the first level at which P_i occurs.	$\neg((d=\text{true}) \oplus V_j (P_i^j \neq \phi))$, penalty= *p* where $j=l,l+1..,k$ *l* is the first level in the planning graph at which P_i occurs *d* is the decision variable associated with this soft constraint
Within <num> <P_i>	A dummy predicate of the same type as discussed for the **Sometime <P_i>** is introduced in this case. Also, to ensure that the planning graph is expanded until the predicate P_i appears in it. If D_i does not appear in the <num>th level of the planning graph, then failure is reported and planning is abandoned.	$V_j (P_i^j \neq \phi)$ where $j=l,l+1,..,$<num> and *l*(<num) is the first level in the planning graph at which P_i occurs.	$\neg((d=\text{true}) \oplus V_j (P_i^j \neq \phi))$, penalty= *p* where $j=l,l+1,..,$<num> *l*(<num) is the first level in the planning graph at which P_i occurs. *d* is the decision variable associated with this soft constraint
At most once P_i	Insert a dummy predicate D_i 1. in the initial state of the planning problem 2. in the preconditions and delete effects of the actions which have P_i in their add effects.	-	$\neg((d=\text{true}) \oplus (P_i^j = \phi \text{ V } P_i^j = \text{no-op}))$, $((d=\text{true}) \oplus (V_j(P_i^j \neq \phi \wedge P_i^j \neq \text{no-op})))$, penalty = **-***p* where $j=l,l+1,..,k$ *l* is the first planning graph level at which P_i occurs *d* is the decision variable associated with this soft constraint
Sometime Before P_i P_j	If P_i is present in the initial state, return failure and abandon planning. If P_j is not present in the initial state, then 1. Put a dummy predicate D in pre-conditions for actions causing P_i. 2. Assert this predicate D in the add effects of actions which have P_j in their add effect list.	-	$\forall_r \neg((d_r=\text{true}) \oplus ((P_i^r \neq \phi) \Rightarrow V_l (P_j^l \neq \phi)))$, penalty=*p* where *r* takes the level numbers at which P_i occurs. $l=s,s+1,..r-1$ and *s*(<*r*) is the first level at which P_j becomes true d_r is the decision variable associated with these soft constraints

Sometime After P_i P_j	1. Assert a dummy predicate D_1 in the add effects of actions which have P_i in their add effect list. 2. Assert a dummy predicate D_2 in the add effects of actions which have P_j in their add effect list. 3. Modify goal test condition: If D_1 occurs in the final predicate level and is non-mutex with the goal set, then D_2 must also be part of final predicate level and be mutex free with the goal set and D_1.	$\forall_r$ ($P_i^r \neq \phi \Rightarrow V_s$ ($P_j^s \neq \phi$)) where r takes the level numbers at which P_i occurs $s=l,l+1,..,k$ $l(>=r)$ is the first level at which P_j occurs	$\forall_r \neg((d_r=\text{true}) \oplus ((P_i^r \neq \phi) \Rightarrow V_s (P_j^s \neq \phi)))$, penalty=p where r takes the level numbers at which P_i occurs $s=l,l+1,..k$ and $l(>=r)$ is the first level at which P_j becomes true d_r is the decision variable associated with these soft constraints
Always-within <num> <P_i> <P_j>	Add dummy predicates similar to those of Sometime after and modify the goal condition similarly. If D_1 occurs in the <num>th level of the planning graph, then D_2 must also occur at that level and should be non-mutex with D_1.	$\forall_r$ ($P_i^r \neq \phi \Rightarrow V_s$ ($P_j^s \neq \phi$)) where r takes the level numbers at which P_i occurs $s=l,l+1,..,r+$num $l(>=r)$ is the first level at which P_j occurs	$\forall_r \neg((d_r=\text{true}) \oplus ((P_i^r \neq \phi) \Rightarrow V_s (P_j^s \neq \phi)))$, penalty= p where r takes the level numbers at which P_i occurs $s=l,l+1,..r+$num and $l(>=r)$ is the first level at which P_j becomes true d_r is the decision variable associated with these soft constraints
Hold-during <num1> <num2> <P_i>	-	If P_i appears in <num1>th level, add a hard constraint of the form $\wedge_j$ ($P_i^j \neq \phi$) where j = num1, num1+1,.., min(num2-1,k)	If the planning graph is expanded beyond num1 levels, add a soft constraint of the form $\neg((d_r=true) \oplus \wedge_j (P_i^j \neq \phi))$, penalty= p where j=num1,num1+1,..,min(num2-1,k) d_r is the decision variable associated with these soft constraints
Hold-after <num> P_i	-	If P_i appears in <num+1>th level, add a hard constraint of the form: $\wedge_j$ ($P_i^j \neq \phi$) where j=num+1,…,k	If the planning graph is expanded beyond <num>th level, add a soft constraint of the form: $\neg((d_r=\text{true}) \oplus \wedge_j (P_i^j \neq \phi))$, penalty= p where j=num+1,...,k d_r is the decision variable associated with these soft constraints

Table2: OCSP Encoding for PDDL3 Modal Operators

In all the above cases, the planning graph is expanded until all the hard constraints are satisfied. If the planning graph levels-off before the hard constraints are satisfied, then a failure is announced and planning is abandoned. In the case of soft constraints, the appropriate penalty will be levied.

Apart from PDDL3 modal operators, we can also handle conditional goal preferences. These are not part of PDDL3. These imply that the desirability of one goal proposition depends on whether or not other goal propositions are satisfied. For example, " I prefer airport A_1 to airport A_2 as the final location for plane P_1, if the final location for plane P_2 is airport A_2". These can be handled using conditional activation constraints. Let G_1, G_2 and G_3 are goal propositions. Let us assume that G_2 is preferred to G_3 if G_1 happens and one of G_2 and G_3 should definitely be included in the goal set.

This can be expressed as:

$G_1 \neq \phi \Rightarrow G_2 \neq \phi$, penalty=$p_1$

$G_2 \neq \phi$ V $G_3 \neq \phi$

Analysis

The hard constraints are pre-processed to prevent the OCSP encoding process being called prematurely without expanding the planning graph to the required level. The preprocessing done is of two types

- Introduction of dummy predicates in the pre-conditions and effect lists of actions, and initial and final states.
- Modification of goal-test condition in the planning graph expansion phase.

Table 3 indicates the type of pre-processing done for each modal operator. Pre-processing is not useful for the soft constraints as the satisfaction of these constraints is not mandatory. The modal operators that introduce more constraints are *sometime-after, always-within* (in the case of hard constraints). This is because these two modal operators impose precedence constraints, which cannot be handled through pre-processing.

Modal Operator	Hard Constraints			Soft Constraints	
	No. of Dummy Predicates Used	**No of additional Constraints**	**No. of Variables in each Constraint**	**No of additional Constraints**	**No. of Variables in each Constraint**
at end <P_i>	0	1	1	1	2
always <P_i>	0	0	0	k	2
*sometime <P_i>	1	1	$k - l$	1	$k - l + 1$
*within <num><P_i>	1	1	$num - l$	1	$num - l + 1$
+at-most-once <P_i>	1	0	0	$k - l + 1$	$k - l$: Ternary 1 : $k - l + 1$
#sometime-after <P_i> <P_j>	2	$k - l + 1$	$k - r + 1$	$k - l + 1$	$k - r + 2$
sometime-before <P_i> <P_j>	1	0	0	$k - l + 1$	$l - r + 3$
#always-within <num> <P_i> <P_j>	2	$k - l + 1$	$l + num - r + 1$	$k - l + 1$	$l + num - r + 2$
hold-during <num1> <num2> <P_i>	0	1	$min(k,num2) - num1$	1	$min(k, num2) - num1 + 1$
hold-after <num> <P_i>	0	1	$k - num$	1	$k - num + 1$

l is the first level in the planning graph at which P_i occurs.
r is the first level in the planning graph at which P_j occurs
$r > l$ for some-time-after and always-within
$r < l$ for some-time-before
* indicates that goal set is modified; + indicates that initial state is modified; # indicates goal test condition is modified

Table 3: Number of Constraints and Variables in the encoding

In the case of soft-constraints, *always, at-most-once, sometime-after, sometime-before and always-within* are the modal operators with the most number of constraints. Every violation of the soft constraint incurs a certain penalty. This implies that more constraints are added as the number of possibilities of violation increases. So, soft constraints contribute more to the number of constraints. Also, the number of variables in the soft constraints is more than that of hard constraints, because of the introduction of decision variable along with each soft constraint.

Conclusions

In this paper, we investigate the encoding of classical planning problems with plan constraints and goal preferences into OCSP. An OCSP encoding for the new features of PDDL3, plan trajectory constraints (hard/soft constraints), goal preferences (hard/soft goals), is proposed. Hard trajectory constraints and goals are tackled by using dummy predicates to pre-process the constraints and converting them into OCSP constraints with infinite penalty. Soft trajectory constraints and goals are addressed by converting them into OCSP constraints with the specified penalty and augmenting each soft constraint with a decision variable. A method to encode conditional goal preference constraints into OCSP is also proposed. We plan to extend this research to metric-temporal domains.

References

[1] Alfonso Gerevini and Derek Long. August 2005. Plan Constraints and Preferences in PDDL3. Technical Report, Department of Electronics for Automation, University of Brescia, Italy.

[2] Avrim L Blum and Merrick L Furst. 1997. Fast Planning Through Planning Graph Analysis. Artificial Intelligence. 90: 281-300.

[3] Martin Cooper and Thomas Schiex. 2004. Arc consistency for soft constraints. Artificial Intelligence 154: 199-227.

[4] Martin Sachenbacher and Brian C. Williams. 2005. 7th International Workshop on Preferences and Soft Constraints, held in conjunction with 11th International Conference on Principles and Practice of Constraint Programming.

[5] Minh Binh Do and Subbarao Kambhampati. 2001. Planning as Constraint Satisfaction: Solving the Planning Graph by Compiling it into CSP. Artificial Intelligence 132(2): 151-182.

[6] Ronen I Brafman and Yuri Chernyavsky. 2005. Planning with Goal Preferences and Constraints.in proceedings of ICAPS-05, Menlo Park, CA, USA.

[7] Thomas Schiex, Fargier, H., Verfaillie, G. 1995. Valued Constraint Satisfaction Problems: Hard and easy problems. Proc. IJCAI-95: 631-637.

[9] Vipin Kumar. 1992. *Algorithms for constraint satisfaction problems: A survey*. AI magazine, 13(1):32-44.

[10] Williams, B., Ragno, R. Conflict-directed A* and its Role in Model-based Embedded Systems. Journal of Discrete Applied Mathematics.