Extending Graphplan to Domain Axiom Planning

Massimiliano Garagnani
Department of Computing - The Open University
Milton Keynes, MK7 6AA (U.K.)
m.garagnani@open.ac.uk

1 Introduction

This short paper proposes an extension of the Graphplan [1] planning system. The extension allows Graphplan to solve planning problems containing *Domain Axioms* (DAs) in the form $p_1 \wedge p_2 \wedge \ldots \wedge p_n \rightarrow c$.

The problem of DAs arises when the domain representation language adopted is augmented with more expressive terms. For example, in the blocks-world domain, if the set of blocks present in the initial state is unknown, it is not possible to fully express the condition/goal "*block* A *is on the same pile of block* B" using only the terms **on**(*x*,*y*), **clear**(*x*) and **on-table**(*x*).[1]

On the other hand, if the terms **above**(*x*,*y*) or **under**(*x*, *y*) (with their obvious meaning) are introduced in the domain definition language, the planning problem needs to be augmented with the following domain axioms:

$$\begin{cases} \textbf{on}(x,y) & \rightarrow \textbf{above}(x,y) \quad (1) \\ \textbf{on}(x,y) \wedge \textbf{above}(y,z) & \rightarrow \textbf{above}(x,z) \quad (2) \\ \textbf{above}(x,y) & \rightarrow \textbf{under}(y,x) \quad (3) \end{cases}$$

The solution proposed by Gazen and Knoblock in [5] to planning problems containing DAs consists of transforming each domain axiom $p_1 \wedge p_2 \wedge \ldots \wedge p_n \rightarrow c$ into an equivalent '*deduce*' operator (P, A, D) = ([$p_1 \wedge p_2 \wedge \ldots \wedge p_n$], [$c$], [ ]), which is then employed by the planner in the usual way during its search for a plan. In addition to this transformation, if the effects of the operator '*O*' contain one of the premises $p_1, p_2, \ldots p_n$ of an axiom having consequence *c*, the method requires that the deletion $\forall \vec{v} \overset{Del}{\Rightarrow} c(\vec{v})$ be appended to the effects of *O*. This is because any deduced proposition may lose its validity when an operator 'clobbers' [2] the premise of the axiom from which the proposition was derived. However, the clobbering of the premise of an axiom does not require to delete *every possible instance* of the axiom's consequence. Because of this 'conservative' approach adopted to invalidate the deduced propositions, Gazen and Knoblock's solution results to be *incorrect* and inefficient [3]. A correct (and more efficient) version of the above pre-processing algorithm has been described in [3] [4]. The new version makes still use of deduce operators, but introduces also conditional effects that add and remove '*deduction facts*' from the planning state.

In this paper, an extension of Graphplan is proposed, which allows the system to solve planning problems containing domain axioms (in the form $p_1 \wedge p_2 \wedge \ldots \wedge p_n \rightarrow c$) without employing conditional effects and without cluttering the planning state with extra deduction facts.

[1] Even if the set '*B*' of blocks is known, the complexity of this expression increases *exponentially* with #*B* [4].

2 The Extended Algorithm

The new algorithm proposed uses the Planning Graph to identify unequivocally which propositions have been derived from a given premise of an axiom through the application of deduce operators.

The basic idea is better clarified with an example. Consider a blocks-world domain with DAs (1) and (2), introduced earlier on in Section 1. Assume that (1) and (2) are transformed into the following (STRIPS-like) deduce operators:

A1(*x*,*y*)

P	**on**(*x*,*y*)
A	**above**(*x*,*y*)
D	

A2(*x*,*y*,*z*)

P	**on**(*x*,*y*), **above**(*y*,*z*)
A	**above**(*x*,*z*)
D	

Given the operator schema

Put-On(*x*,*y*,*z*)

P	**on**(*x*,*y*), **clear**(*z*), **clear**(*x*)
A	**on**(*x*,*z*), **clear**(*y*)
D	**on**(*x*,*y*), **clear**(*z*)

and the initial state I={**on**(**A**,**B**), **on**(**B**,**C**), **clear**(**A**), **clear**(**D**)}, Figure 1 shows a portion of the first few levels of the Planning Graph that Graphplan would build.

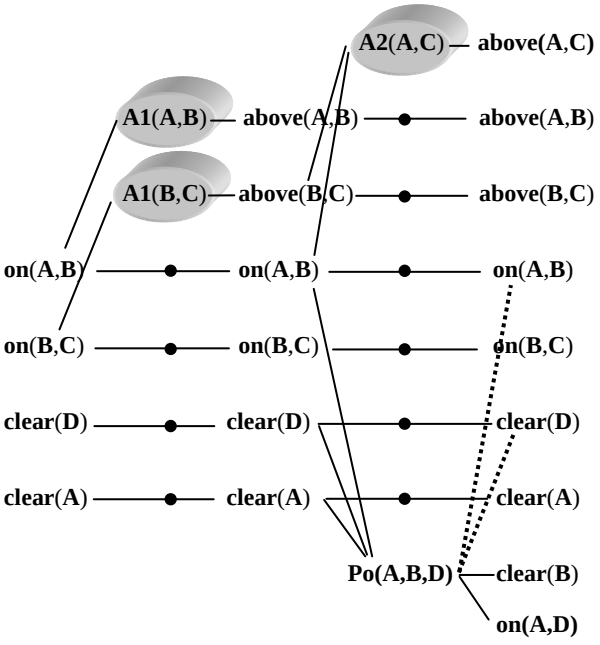

Figure 1. Planning Graph with instances of the 'deduce' operators **A1**( ) and **A2**( )

Notice that the '*deduce'* nodes **A1**( ) and **A2**( ) have been given here a particular graphical representation, distinct from that of instances of 'normal' operator schemes (such as **Put-On**(**A**,**B**,**D**), represented by **Po**(**A,B,D**) in Fig. 1). Although such nodes are treated identically to the others during the construction of the graph, they behave differently with respect to the *propagation* of the effects, as explained below.

The Planning Graph of Fig. 1 contains the application of **Po**( ) in step 2, which deletes the term **on**(**A**,**B**) (and **clear**(**D**)). This causes other propositions (which had been derived from it) to be no longer valid. More precisely, the planning graph shows that the terms **above**(**A**,**B**) and **above**(**A**,**C**) have been deduced from **on**(**A**,**B**) and **on**(**B**,**C**) using the operators **A1**( ) and **A2**( ) (i.e., axioms (1) and (2)). Hence, the deletion of the premise **on**(**A**,**B**) should be '*propagated'* to its two (and only) consequences, **above**(**A**,**B**) and **above**(**A**,**C**). On the other hand, even though **on**(**A**,**B**) is a *precondition* of **Po**(**A**,**B**,**D**), its deletion is not propagated on to **on**(**A**,**D**) or **clear**(**B**). This is because 'normal' operators represent a *change* in the state of things, whereas 'deduce' operators represent steps in a *chain of reasoning* which can be fully invalidated at any time during the plan.

Notice that, in the previous example, Gazen and Knoblock's method would have simply appended the effect $\forall u,v \overset{Del}{\Rightarrow}$ **above**(*u*,*v*) to the operator **Po**(*x*,*y*,*z*). However, this would have led **Po**(**A**,**B**,**D**) to delete also the proposition **above**(**B**,**C**), which is still valid.

In order to extend Graphplan so that it realises the propagation of effects described above, it is necessary to modify the algorithm for the Planning Graph construction. In brief, the following procedure should be executed every time a new proposition level is generated and for each operator *O*=(P,A,D) which appears in the preceding level:

PROCEDURE ε

For each proposition d∈D:
if 'd' is premise of a domain axiom, then
1. *let S be the set of nodes labelled 'd' in the planning graph;*
2. *let L be the list of leaves[2] of the graph that can be reached [3] from S;*
3. *for each leaf l∈L, add a 'delete' edge going from the operator O to the leaf node 'l';*

End-if;
End.

In the previous example (Fig. 1), let *O*=**PO**(**A**,**B**,**D**). Since **on**(**A**,**B**) is a premise of both axioms, *S* will contain the three nodes of the graph labelled **on**(**A**,**B**). Thus, *L* will contain (*only*) **above**(**A**,**B**) and **above**(**A**,**C**), as we expected.

[2]The leaves of a Planning Graph are the nodes present in the rightmost (proposition) level, or '*fringe'*.
[3]The path must traverse the graph from left to right and cannot contain 'normal' (i.e. non-*deduce*) operator instances (however, it can contain 'No-Op' nodes).

3 Discussion

The most computationally expensive part of procedure ε is step 2, where the identification of the set *L* for a given node labelled *'d'* could potentially lead to a combinatorial explosion. However, this is not true in the long run, since the Planning Graph is *polynomial* in size (cf. [1]) and, after a certain number of levels, proposition and action nodes do 'level-off'. In addition, in order to reduce the computational load of this calculation, it is possible to build the '*L*-sets' *incrementally* during the graph construction, by keeping in memory only those calculated for the previous level. In short, given a Planning Graph containing just two proposition levels, it is easy to calculate the *L*-sets for all the (*relevant*) nodes of the initial state (level 0). Then, when proposition level $i+2$ ($i \geq 0$) is added, the *L*-sets of the nodes appearing in level $i+1$ can be calculated by *extending* those of level *i*, which will then be discarded. The new proposition nodes to be added are those of the 'fringe' (level $i+2$) that can be reached from the nodes (and the *L*-sets) of level $i+1$. Thus, this method avoids recalculating the *L*-sets from scratch every time procedure ε requires them.

Although the ideas presented here are still preliminary, a possible future direction that could be investigated consists of applying this approach to other Graphplan-based planners (such as IPP [6]), or to causal-link planners [7]. Furthermore, an interesting extension would be to modify the procedure ε so that it could deal with DAs containing *existentially* and *universally* quantified terms.

Finally, work is currently in progress to develop a Planning-Graph based system that can discover and *learn* new deduce operators using the given set of domain axioms. This could lead to further improvements in efficiency.

In conclusion, the extension of Graphplan to planning problems with DAs opens up new directions that seem promising and worth investigating.

4 References

[1] Blum, A.L., Furst, M.L. (1997) "Fast Planning Through Planning Graph Analysis", *Artificial Intelligence*, **90**:281-300.

[2] Chapman, D. (1987) "Planning for Conjunctive Goals", *Artificial Intelligence*, **32(3)**:333-377.

[3] Garagnani, M. (to appear) "A Correct Algorithm for Efficient Planning with Preprocessed Domain Axioms", *Proceedings of ES-2000*, December 2000, Cambridge, England.

[4] Garagnani, M. (1999) "Improving the efficiency of Pre-processed Domain-axioms Planning", *Proceedings of PLANSIG-99*, Manchester, England, pp.190-192.

[5] Gazen, B.C., Knoblock, C.A. (1997) "Combining the Expressivity of UCPOP with the Efficiency of Graphplan", *Proceedings ECP-97*, Toulouse, France.

[6] Koehler, J., Nebel, B., Hoffmann, J., Dimopoulos, Y. (1997) "Extending Planning Graphs to an ADL Subset", *Proceedings ECP-97*, Toulouse, France.

[7] Young, R.M., Moore, J.D., Pollack, M.E. (1994) "Decomposition and Causality in Partial Order Planning", *Proceedings AIPS-94*, Chicago, IL.