Enhancing Constraint Models for Planning Problems

Roman Barták*, Daniel Toropila*†
{roman.bartak, daniel.toropila}@mff.cuni.cz

*Charles University, Faculty of Mathematics and Physics
Malostranské nám. 2/25, 118 00 Praha 1, Czech Republic

†Charles University, Computer Science Center
Ovocný trh 5, 116 36 Praha 1, Czech Republic

Introduction

Planning deals with finding a (shortest) sequence of actions transferring the world from its initial state to a state satisfying the goal condition. We assume that each state is specified by values of multi-valued state variables that can be changed only by actions (SAS^+ formalism). Each action consists of preconditions specifying required values of certain state variables and effects of setting the values of state variables. The planning problem is given by values of all state variables in the initial state and by required values of certain state variables as the goal condition.

The planning problem can be solved by translation into formalisms such as constraint satisfaction. The idea is that the problem of finding a plan of given length is encoded as a constraint satisfaction problem (CSP). If the CSP has a solution then we decode back the plan, otherwise the length of sought plan is extended by one and the process is repeated. There exists a straightforward constraint model of this type but more advanced constraint models exploit the structure of a so called planning graph, namely GP-CSP and CSP-PLAN. In (Barták and Toropila 2008), we presented reformulations of above mentioned approaches to state-variables formalism and we showed that a much better efficiency can be achieved by encapsulating the logical constraints describing changes of state variables by actions into combinatorial constraints with an extensionally defined set of admissible tuples. The best results were obtained by the reformulated modal à la CSP-PLAN (Lopez and Bacchus, 2003) – called the *base model* here.

Model Enhancements

It is rare that a direct constraint model is enough to solve complex problems and frequently some extensions are necessary. In this paper we propose three improvements of the base model. In particular, we suggest using singleton consistency to prune more of the search space by eliminating certain unreachable actions, using dominance rules to break plan permutation symmetries in the problem, and finally using lifting (called domain splitting in constraint satisfaction) to decrease the branching factor.

Lifting

The search strategy used for the base model resembles the labeling technique in constraint satisfaction. At each step, we select an action that contributes to the current goal (regression/backward planning is used). As noted in (Ghallab *et al.* 2004) this strategy may be overcommitted, for example, when requiring a robot to be at certain location by selecting the move action we are also deciding from which location the robot will go. There might be many such actions (depending on the number of locations) so it seems more appropriate to postpone some particular decision to later. This is called *lifting* as instead of selecting a particular action we lift the decision by assuming a set of "similar" actions reaching the same goal. In terms of constraint satisfaction, this is realized by splitting the domain of action variable rather than instantiating the variable.

Let us now describe the process of lifting more formally. Let PrecVars(a) be the state variables appearing in the precondition of action a and EffVars(a) be the state variables changed by action a (these variables appear in effects of a). We say that actions a and b have the same scope if and only if PrecVars(a) = PrecVars(b) and EffVars(a) = EffVars(b). Let the base search procedure select action a to be assigned to variable A^s; in other words we split the search space by resolving the disjunction $A^s = a \vee A^s \neq a$. In the lifted version, we are resolving the following disjunction:

$$A^s \in \text{SameScope}(a) \vee A^s \notin \text{SameScope}(a),$$

where SameScope(a) = $\{ b \mid b$ has the same scope as $a \}$.

Dominance Rules (a.k.a. Symmetry Breaking)

Recall, that we are looking for sequential plans. Assume that we have two actions a_1 and a_2 such that these actions do not interfere, for example, move action of a robot and load action of a different robot. If we have a valid plan where a_1 is right before a_2 then a plan where we swap both actions is also valid. This feature, called *plan permutation symmetry* (Long and Fox 2003) can be exploited during search in the following way.

First, we need to define formally what it means that two actions do not interfere. Recall, that our motivation is that two actions a_1 and a_2 can be swapped without influencing validity of the plan. Swapping of actions a_1 and a_2 can be realized if for any state s the following condition holds:

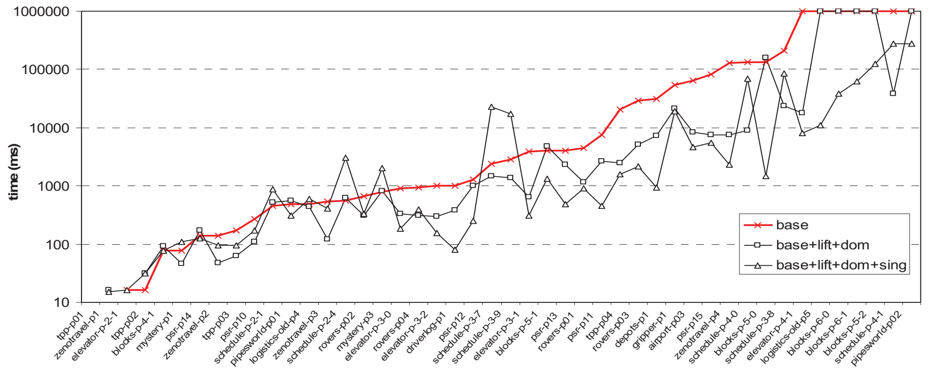

Fig. 1. Comparison of runtimes (logarithmic scale) for selected problems from IPC 1-5 when the methods are combined.

$\gamma(\gamma(s,a_1),a_2) = \gamma(\gamma(s,a_2),a_1)$, where $\gamma(s,a)$ is a state obtained by applying action a to state s. Such situation happens if actions a_1 and a_2 are independent (Ghallab *et al.* 2004). We suggest the following two conditions for action independence in the multi-valued state representation:

$$\mathrm{EffVars}(a_2) \cap (\mathrm{PrecVars}(a_1) \cup \mathrm{EffVars}(a_1)) = \emptyset,$$
$$\mathrm{EffVars}(a_1) \cap (\mathrm{PrecVars}(a_2) \cup \mathrm{EffVars}(a_2)) = \emptyset.$$

Once we know how to recognize independent actions, we propose to include the following dominance rule to the search procedure. We choose arbitrary ordering of actions such that action a_i is before action $a_\mathrm{i+1}$ in the ordering (this ordering has nothing in common with the ordering of actions in the plan). Assume that action a_i has been assigned to state variable A^s (action at position s). Then, when selecting action for state variable A^{s-1}, we only consider actions a_j for which at least one of the following conditions holds: either a_j and a_i are not independent, or $\mathrm{j} > \mathrm{i}$. This way, we prevent the solver from exploring permutations of mutually independent actions.

Singleton Consistency

So far, we discussed improvements of the search strategy. Another way to improve efficiency of constraint solving is incorporating a stronger consistency technique. Singleton arc consistency (SAC) would be a good candidate because it is easy to implement on top of arc consistency. However, it is computationally expensive to make the problem SAC so we suggest applying SAC in a restricted form.

When a new layer is added to the constraint model we check whether the newly assumed actions have a support in the previous layer. Formally, let P be a constraint model describing the problem of finding a plan of length n+1 and a be an action that appears in the domain of A^n but not in the domain of A^{n-1} (a newly introduced action). For any precondition V_i=v of a, if there is no action b such that V_i=v is among its effects, and $\mathrm{P}|A^n$=a, A^{n-1}=b is arc consistent, then a can be removed from the domain of A^n. The reason for filtering out action a is that there is no plan of length n giving the preconditions of a (the n-th layer is the first layer where action a appeared so the precondition cannot be provided by actions before layer n-1).

Conclusions

The paper proposed three enhancements of the base constraint model for solving sequential planning problems. Common feature of these enhancements is an attempt to reduce search space, large size of which is a major obstacle when solving planning problems. We implemented the proposed enhancements in SICStus Prolog 4.0.2 and compared them using selected planning problems from past International Planning Competitions (STRIPS versions). The experiments ran on Pentium M 730 1.6 GHz processor with 1GB RAM under Windows XP. Figure 1 shows a comparison of several combinations of the methods and it clearly demonstrates that they significantly outperformed (orders of magnitude) the base model, especially when the problems become hard.

Acknowledgments

The research is supported by the Czech Science Foundation under the project 201/07/0205.

References

Barták, R. and Toropila D. 2008. Reformulating Constraint Models for Classical Planning. *Proceedings of the 21st International Florida AI Research Society Conference (FLAIRS 2008)*. AAAI Press, pp. 525-530.

Ghallab, M., Nau, D., Traverso P. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann.

Long, D. and Fox. M. 2003. Plan Permutation Symmetries as a Source of Planner Inefficiency. *Proceedings of 22nd Workshop of UK Planning and Scheduling Special Interest Group (PlanSIG-22).*

Lopez, A. and Bacchus, F. 2003. Generalizing GraphPlan by Formulating Planning as a CSP. *Proceedings of IJCAI*, 954-960.