Optimal SAT-based Planning with Macro-actions and Learned Horizons

Alfonso E. Gerevini and **Alessandro Saetti** and **Mauro Vallati**
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Brescia
Via Branze 38, 25123 Brescia, Italy
{gerevini,saetti,mauro.vallati}@ing.unibs.it

Introduction

Planning as propositional satisfiability (SAT) is a powerful approach for computing optimal plans in terms of Graphplan plan length. SatPlan (Kautz and Selman 1992) is one of the most popular and efficient planning system adopting this approach. First, it computes a lower bound k of the optimal plan length. Then, using k as the planning *horizon*, i.e., a fixed time step after which actions cannot be executed, it translates the planning problem into a SAT problem Π, which is then solved by an existing SAT solver. If Π is satisfiable, then the assignment to propositional fluents satisfying the SAT problem is translated into a plan of actions that is a solution of the original planning problem. Otherwise (Π is unsatisfiable), the process is repeated using an increased value of k. A critical weakness of the approach is that often the initial value of k is much less than the optimal plan length, and hence many unsolvable SAT problems can be generated and processed.

The performance of a planning system is typically affected by the structure of the search space, which depends on the considered planning domain. In many domains, the planning performance can be improved by deriving and exploiting some knowledge about the domain structure that is not explicitly encoded in the input domain formalization. Well known examples of such knowledge are macro actions (e.g., (Botea *et al.* 2005; Newton et al. 2007)). A macro-action is a sequence of domain actions that can be planned at one time like a single action. Using macro-actions the planning process is often faster, but the length of the computed solution plan can be worse than optimal.

In this paper, we propose a SAT-based optimal planner, called MacroSatPlan, which exploits two types of knowledge learned for a given domain to speedup the SAT solving: (i) a predictive model based on some problem features estimating the optimal plan length, and (ii) useful sets of learned macro-actions.

A preliminary experimental analysis shows that the learned knowledge is useful for speeding up the computation of the optimal solution of the planning problem.

Architecture of MacroSatPlan

The architecture of MacroSatPlan, sketched in Figure 2, consists of three main modules, briefly described below.

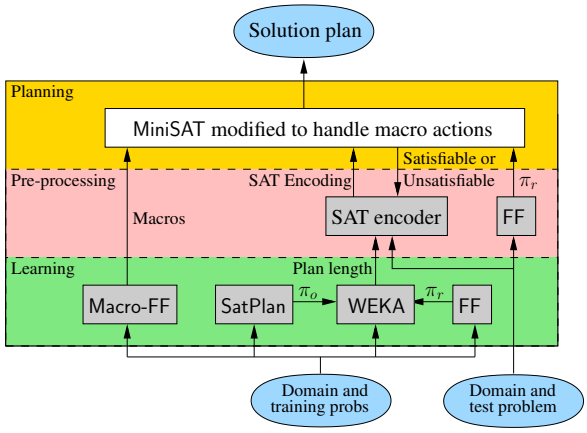

Figure 1: A sketch of MacroSatPlan's architecture. π_o and π_r indicate an optimal plan and a relaxed plan, respectively.

Learning module. This module consists of four main components: the 2006 version of SatPlan (Kautz, Selman, and Hoffmann 2006), a modified version of planner FF (Hoffmann and Nebel 2001), Macro-FF (Botea *et al.* 2005), and the machine learning tool WEKA (Witten and Frank 2005).

FF is a forward search planner which exploits GraphPlan for computing a relaxed plan π achieving the problem goals from the successor states. The number of actions in π is an estimate of the distance from the successor states to the goal states. We use a revised version of FF that only computes the length of the relaxed plan derived from the initial state of the planning problem, without computing a solution for it.

WEKA is a well-known tool for learning predictive models. Given a set of training problems for domain D, WEKA is used to identify a predictive model of the optimal plan length for a given problem in domain D from (i) the length of the optimal plan computed by SatPlan, (ii) some pre-identified features of the planning problem, and (iii) the length of the relaxed plan computed by FF. The optimal plan length predicted by WEKA is subsequently used by the SAT encoder of the planning problem as the initial value of the planning horizon. It is important to note that, differently from SatPlan, in our approach the initial horizon can be higher than the optimal plan length.

Macro-FF (Botea *et al.* 2005) is a planning system computing macro-actions and using them for solving planning problems. The method implemented into the Macro-FF system computes the macros by analyzing the solutions of a set of training problem instances, so that the macros that appear frequently and that reduce the required search effort significantly are preferred. The computed macros are subsequently used by a modified version of SAT-solver MiniSAT (Eèn and Sörensson 2003) during the planning phase.

Preprocessing module. This module consists of two existing systems: the modified version of FF and the SAT encoder used by SatPlan to compile a planning problem into a SAT-problem.

The compilation of the original planning problem (without macros) is done using as horizon the predicted plan length value. It is important to note that if the domain defining the input planning problem were extended with the macro-actions computed by the learning module, the plan derived from the solution of the SAT solver could be suboptimal.

The input of the planning module includes the SAT encoding, together with the computed macro-actions and the relaxed plan computed by FF for achieving the goals from the initial state of the planning problem.

Planning module. This module consists of a variant of MiniSAT that can exploit the macro-actions computed by the learning module to guide the search during SAT solving.

At each search step, the original version of MiniSAT selects an unassigned variable, and sets it to the *false* value. Then, this decision is in turn propagated by unit propagation. As soon as a clause becomes unary under the current assignment, the remaining literal in the clause is set to true and this decision is propagated, possibly reducing other clauses to unary clauses. The propagation process continues until no more information can be propagated. If a conflict is encountered (all literals of a clause are false), a "conflict clause" is constructed and added to the SAT problem. The decisions made are retracted by backtracking until the conflict clause becomes unary; this unary clause is propagated, and the search process continues.

The modified version of MiniSAT assigns true to the selected variable, instead of false, preferring unassigned variables belonging to the most promising macro-actions. These are macro actions involving actions in the relaxed plan previously computed by FF. Ties are broken using some secondary criteria, such as the ratio between the number of variables with *true* value and the size of the macro, and the time step of the first action forming the macro.

If the SAT encoding of the planning problem with predicted horizon t is initially solvable, the process is repeated with horizon $t-1$, and so on, until a horizon $q < t$ for which the SAT encoding is unsolvable has been identified. Otherwise, the process is repeated with an increased value of t, and it terminates when a solvable SAT encoding is generated. The optimal solution is derived from the last computed assignment satisfying the SAT problem.

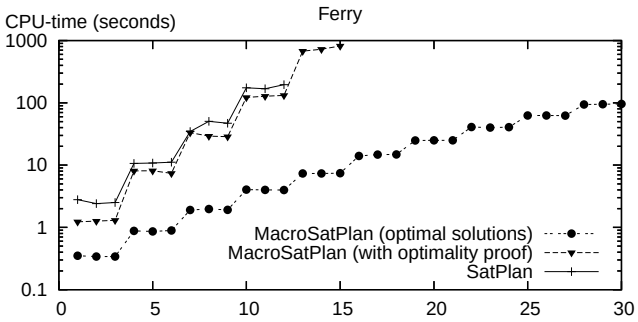

Figure 2: CPU times of SatPlan and MacroSatPlan for finding an optimal solution and proving its optimality for domain `Ferry`. On the x-axis, we have the problem names simplified by numbers; on the y-axis, the CPU times (logarithmic scale).

Preliminary Experimental results

Figure 2 shows the CPU time required by MacroSatPlan to find the optimal solution and proving its optimality after having found it with respect to the time required by SatPlan, using 30 instances of domain `Ferry`. Concerning finding and proving optimal solutions, MacroSatPlan is slightly faster than SatPlan. On the other hand, the CPU time used to generate the optimal solution (without proving its optimality) is much better than the CPU time of SatPlan.

Conclusions and Future Work

We have briefly described MacroSatPlan, a planner based on the propositional satisfiability approach using macro-actions and learned horizons. A preliminary experimental study shows that the learned knowledge can be useful for generating optimal solutions much more quickly than SatPlan. Future work includes additional experiments about the relative impact of the macro-actions and the predictive model of the optimal plan length on the performance of the planning system, and the integration of Wizard (Newton et al. 2007) as an alternative system for learning macro-actions.

References

A. Botea, M. Enzenberger, M. Müller and J. Schaeffer. 2005. Macro-FF: Improving AI Planning with Automatically Learned Macro-Operators. *JAIR*, 24:581–621.

Eèn, N.; Sörensson, N. 2003. An Extensible SAT-solver In *Proc. of SAT-03*.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: fast plan generation through heuristic search. *JAIR*, 14:253–302.

Kautz, H.; Selman, B. 1992. Planning as satisfiability. In *Proc. of ECAI-92*.

Kautz, H.; Selman, B.; and Hoffmann, J. 2006. Satplan: Planning as satisfiability. In *Abstracts of the 5th Int. Planning Competition*.

Newton, M. A. H.; Levine, J.; Fox, M.; and Long, D. 2007. Learning macro-actions for arbitrary planners and domains. In *Proc. of ICAPS-07*.

Witten, I. H., and Frank, E. 2005. *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*.