MaxPlan: Optimal Planning by Decomposed Satisfiability and Backward Reduction

Zhao Xing, Yixin Chen, and Weixiong Zhang
Department of Computer Science and Engineering
Washington University in St. Louis
Saint Louis, MO 63130
{zx2,chen,zhang}@cse.wustl.edu

Abstract

Planning as satisfiability is one of the best approaches to optimal planning. In addition to being effective and efficient on many planning domains, this approach is general, in that a generic method for satisfiability (SAT) can be used. As a result, it is able to take advantage of the general research devoted to SAT. Nevertheless, the potential of this approach has not been fully exploited. In the MaxPlan planner, we develop an efficient SAT solving algorithm that effectively exploits the structure of planning applications and decomposes the original SAT problem into a series of much simpler SAT subproblems. The decomposed SAT approach has been shown to be significantly more efficient than the previous approach which uses a generic SAT solver as a black-box without exploiting the problem structure. Several other novel techniques, including backward level reduction, accumulative learning of clauses, and search-space pruning based on multi-valued domain formulation, are also developed to further improve the search efficiency for both satisfiability solving and unsatisfiability proving.

1 Introduction

MaxPlan is an optimal planner for STRIPS-like propositional planning problems. It is optimal in terms of the number of parallel steps. It, in essence, follows the paradigm of planning as satisfiability [Selman & Kautz, 1992], which has emerged as one of the most effective formulations for optimal planning. The method of planning as satisfiability first transforms a STRIPS planning problem into a satisfiability (SAT) problem, and then solves the SAT problem using a generic SAT solver.

Representative planners under the paradigm of planning as satisfiability include Blackbox [Kautz & Selman, 1996] and SATPLAN [Kautz,]. The latest version of SATPLAN, SATPLAN04, has won the First Place prize in the optimal track of the Fourth International Planning Competition (IPC4). Despite its success, the current best realization of planning as satisfiability still has the following limitations.

a) SATPLAN performs a ***forward level expansion*** search that keeps increasing the estimated plan length and for each fixed length finds a solution or proves the problem unsolvable. We have noticed that most of the computing time that it spends on is to prove unsatisfiability, which could be expensive because the entire search space may need to be explored. This observation inspires us to search from the opposite direction, i.e. to reduce the estimated plan length from an upper bound. When the plan is long, however, finding a feasible solution can still be expensive. Effective strategies for reducing complexity are needed.

b) By transforming a planning problem into a SAT problem and solving the problem in a "blackbox" fashion by a SAT solver, the structural and goal information in the planning problem is discarded. In contrast, heuristic search-based planning methods typically combine directed forward search with backward chaining to explicitly exploit information in the planning goals. This motivates us to study and utilize the internal structure of the SAT encoding.

c) The existing SAT planners solve a SAT problem as a whole, which becomes prohibitively expensive for large problems. One objective in the design of MaxPlan is to decompose a planning problem based on its structure in order to reduce search complexity.

d) To achieve the optimal time steps, the current realization of planning as satisfiability follows an incremental scheme in which the number of time steps is increased by one after each failed iteration. A careful examination of the process of the current approaches has revealed key insights into this general planning paradigm. First, the SAT problem instances derived at different time steps share similar structures. Second, the knowledge learnt from solving SAT instances for shorter time steps can be accumulated and used to speed up the processes for solving SAT instances of longer time steps. However, the existing SAT planners generate SAT problem instances independently from scratch and solve them in isolation so that knowledge learnt during previous iterations was lost.

The design of MaxPlan aims at overcoming the above limitations.

2 Overall Architecture of MaxPlan

Based on an action-based SAT formulation, MaxPlan employs a new decomposition scheme to decompose a planning problem into a series of SAT subproblems, one for each subgoal variable, in order to reduce the search complexity. Goal-oriented rules for selecting variables in SAT solvers are introduced to exploit logic structures of planning problems. Other novel techniques, including ***backward level reduction***, accumulative learning, and search space pruning based on a multi-valued formulation, are also developed and integrated with a

generic SAT solver to further speed up the solution process.

The overall process of MaxPlan works as follows.

1. Estimate an upper bound of the optimal plan length. The search keeps tightening the upper bound until a plan length has been proved unsolvable.
2. For a given plan length, compile the input planning problem into a SAT formulation.
3. Solve the derived SAT problem using a novel SAT solver integrated with goal-oriented decomposition and search space pruning. Quit if the current plan depth is proven unsatisfiable.
4. After a plan depth is solved, use an accumulative learning scheme for taking advantage of the structural similarities among the SAT instances. Accumulate the knowledge (in the form of learnt clauses and pruning rules) to accelerate the processes of solving sebsequent instances.
5. Reduce the estimated optimal plan length and repeat from step 2.

MaxPlan significantly differs from the previous SAT-based planners in the overall direction of plan-length estimation and inter-level learning. Furthermore, instead of using a generic "blackbox" SAT solver, MaxPlan integrates in the SAT solver a number of effective strategies for exploiting the structure of a planning problem.

3 Backward Level Reduction and Analysis of Action-Based Encoding

MaxPlan uses an approach to set the initial number of parallel steps which is different from that used by the previous SAT planners. Since the previous SAT planners use forward level expansion, they typically apply the Graphplan approach [Blum & Furst, 1997] to construct a relaxed planning graph to estimate a ***lower*** bound of the optimal parallel length. In contrast, since MaxPlan uses backward level reduction, it estimates an ***upper*** bound using a suboptimal planner to find a suboptimal sequential plan and parallelize it. We use the Fast Forward planner [Hoffmann & Nebel, 2001] in our current implementation.

To transform a planning problem into SAT, we use the same action-based encoding as used by SATPLAN04, which converts all fact variables to action variables.

We have closely studied the action-based SAT encoding. We classify the clauses into several classes, including the *E clauses* that are binary clauses generated from mutual exclusions, the *A clauses* that specifies the dependencies among actions, and the *G clauses* that specifies subgoals.

Most (but not all) variables in E clauses can be instantiated by applying unit propagation in a SAT solver. In other words, a majority of independent variables appear in the G clauses. As a result, the variables in the binary E clauses have a lower priority to be chosen. Therefore, the three classes of clauses should be treated differently. The G clauses should be treated as "core clauses" in SAT solving. In general, the G clauses only constitute a very small portion (typically less then 2 percents) of all the clauses. Focusing on branching those variables that appear in the G clauses first can significantly reduce the search space in SAT solving.

Most, if not all, existing SAT solvers lack mechanisms for exploiting structural information of real-world instances and for identifying independent variables. Identifying problem structural information is a hard problem. Being generic algorithms, these SAT solvers try to produce conflicts by branching on shorter clauses (such as heuristic [Li & Anbulagan, 1997] or the Jeroslow-Wang rule [Hooker & Vinay, 1995]), or attempt to detect conflicts by applying the locality of conflicts (such as the VSIDS heuristic [Moskewicz *et al.*, 2001]). These general heuristics can hardly detect structural properties of a planning problem.

4 Goal-Oriented Decomposition for SAT Solving

The majority part of the computation complexity of MaxPlan lies in solving the SAT problems at different parallel steps. In our implementation, we use a generic SAT solver, MiniSAT [Eén & Biere, 2005], enhanced with our new strategies, as the SAT engine for MaxPlan. Note that the decomposition and space pruning strategies proposed in MaxPlan are general and can be incorporated into other SAT solvers.

As mentioned, SAT problems derived from real-world planning problems are often highly structured and contain a substantial portion of variables whose values can be determined by other variables through unit propagation. In other words, these variables are *dependent* variables. In contrast, those variables whose values cannot be directly inferred from the assignments of other variables are *independent* variables. Since the assignments of independent variables determine the values of most dependent variables, the number of assignments that need to be tried can be reduced significantly if we branch on independent variables first. In MaxPlan, independent variables are those appearing in the G clauses, and dependent variables are the ones not in the G clauses.

The complexity of SAT solving can be significantly reduced if we ignore the assignments of some independent variables. Therefore, in MaxPlan, we propose a ***goal-oriented decomposition*** scheme to decompose the SAT problem into a series of much simpler SAT subproblems with incremental involvement of the goal-oriented independent variables.

Suppose that there are N subgoals, we iteratively solve N subproblems. In solving the i^{th} subproblem, we set the variables for the first i subgoals true, and set free the other subgoal variables. After we solve the instance, based on the solution, we increase the priority scores of all action variables whose assignments are true by a constant amount in MiniSat in order to force to branch on these action variable and set them to true first in solving the next subproblem. Therefore, in each iteration, we focus on solving one subgoal. In this process, we also try to explore first the partial solution space that is close to the ones searched in the previous iterations by giving priorities to the related G clauses.

There are several advantages of the goal-oriented decomposition approach. Our preliminary experimental analysis has indicated that after the G variables have been fixed, the problem can be solved quickly by unit propagation. This means that the G variables are the most critically constrained variables in a planning problem. By incrementally increase the number of active critical G variables in a series of subproblems, the complexity of solving each subproblem is significantly reduced. As a result, the total complexity of solving all decomposed subproblems is still much smaller than the complexity of solving

the original SAT problem without decomposition.

The proposed goal-oriented decomposition approach is different from incremental planning. A key difficulty of incremental planning is that it fixes the solutions of solved subproblems and often gets stuck at infeasible deadends. Our decomposition approach, on the contrary, allows backtracking and is a complete algorithm that can be used to prove unsatisfiability.

5 Search Space Pruning

In addition to goal-oriented decomposition, we develop two novel strategies to further reduce the search space of SAT solving. A common feature of the two strategies is that they both prune the search space by adding more constraints to the problem formulations.

5.1 Mutual Exclusion Constraints from Multi-Valued Formulations

In the SAT formulation, the E clauses encode binary mutual exclusion relations among actions. It is very difficult and expensive to detect all mutual exclusions, and most previous SAT planners only detect a small subsets of mutual exclusions. In MaxPlan, we use a new approach that can generate more mutual exclusions efficiently. The approach uses a multi-valued domain formulation (MDF) that translates the binary fact representation into a more compact representation with multi-valued variables. Each variable typically represents an invariant group for an object, and the variable can only take one value in the group at any time. For example, a truck may only be at one location at any time. The location of a truck is a variable, and different locations are the values that the variable may take.

Based on the MDF formulation, we can derive more mutual exclusions among actions. We first derive mutual exclusions among facts. In fact, all the binary facts associated with a multi-valued variable are mutually exclusive. For example, at(truck, location1) and at(truck, location2) are mutually exclusive. Based on the mutual exclusions among facts, we further derive mutual exclusions among actions based on the original definition of mutual exclusion [Blum & Furst, 1997]. The new mutual exclusions among actions are added to the action-based SAT formulation as additional constraints to prune infeasible regions of the search space. This technique improves the efficiency in both finding a feasible plan and proving unsatisfiability.

5.2 Accumulative Learning

An additional technique to further improve the search efficiency is accumulative learning. Typically a SAT solver can learn during search new redundant clauses which provide additional pruning power. This mechanism is called *clause learning*, and the derived clauses are called *learnt clauses*. In the existing SAT planners such as SATPLAN04, learnt clauses are not inherited across iterations. In each iteration, the SAT solver learns clauses from scratch.

Since MaxPlan uses an iterative process to search for an optimal plan, we note that a SAT instance at any iteration resembles the SAT instance for the previous iteration, except for some G clauses due to the change of plan length. We take advantage of such a structural similarity and propose the accumulative learning scheme. This scheme has two key components. Instead of re-encoding the whole problem from scratch after each iteration, we simply modify and patch the previous encoding to specify the new constraints of the next iteration. Therefore, the time for encoding can be significantly reduced, and such saving can be significant for large planning problems. More importantly, we can retain all the learnt clauses (not related to goal clauses) in all the previous iterations and re-use them in the next iteration. As a result, most learnt clauses only need to be learnt once, which saves running time.

It is important to note that the learnt clauses that are retained for the next iteration may be learnt again in the next iteration if they were not retained. Most existing efficient SAT solvers have efficient mechanisms for clause learning that support managing and deleting learnt clauses intelligently. Therefore, our accumulative learning scheme incurs very little additional memory over the original SAT solver.

6 Conclusions

In summary, MaxPlan follows the general paradigm of planning as satisfiability and incorporates significant extensions. First, MaxPlan uses backward level reduction instead of the previous forward expansion approach. Second, MaxPlan uses a novel goal-oriented decomposition method, developed in this paper, to greatly improve the efficiency of solving SAT problems derived from planning. This decomposition method exploits structures of a planning problem and the effects of different classes of clauses in the SAT encoding. Finally, two new strategies, MDF-based constraints and accumulative learning, are developed to further prune the search space.

References

[Blum & Furst, 1997] Blum, A., and Furst, M. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90:281–300.

[Eén & Biere, 2005] Eén, N., and Biere, A. 2005. Effective preprocessing in SAT through variable and clause elimination. In *SAT*.

[Hoffmann & Nebel, 2001] Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *J. of Artificial Intelligence Research* 14:253–302.

[Hooker & Vinay, 1995] Hooker, J., and Vinay, V. 1995. Branching rules for satisfiability. *J. Automated Reasoning* 15:359–383.

[Kautz & Selman, 1996] Kautz, H., and Selman, B. 1996. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of AAAI-96*, 1194–1201.

[Kautz,] Kautz, H. SATPLAN04: Planning as satisfiability. IPC4 abstract, 2004.

[Li & Anbulagan, 1997] Li, C., and Anbulagan. 1997. Heuristics based on unit propagation for satisfiability problems. In *Proceedings of IJCAI-97*, 366–371.

[Moskewicz *et al.*, 2001] Moskewicz, M. W.; Madigan, C. F.; Zhao, Y.; Zhang, L.; and Malik, S. 2001. Chaff: Engineering an Efficient SAT Solver. In *Proceedings of the 38th Design Automation Conference (DAC'01)*.

[Selman & Kautz, 1992] Selman, B., and Kautz, H. 1992. Planning as satisfiability. In *Proceedings ECAI-92*, 359–363.