Optimal STRIPS Planning by Maximum Satisfiability and Accumulative Learning*

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

Abstract

Planning as satisfiability (SAT-Plan) is one of the best approaches to optimal planning, which has been shown effective on problems in many different domains. However, the potential of the SAT-Plan strategy has not been fully exploited. Following the SAT-Plan paradigm, in this paper we formulate a STRIPS planning problem as a maximum SAT (max-SAT) and develop a general two-phase algorithm for planning. Our method first represents a planning problem as a combinatorial optimization in the form of a SAT compounded with an objective function to be maximized. It then uses a goal-oriented variable selection to force goal-oriented search and a accumulative learnt strategy to avoid to learn a learnt clause multiple times. We integrate our new methods with SATPLAN04. We evaluate and demonstrate the efficacy of our new formulation and algorithm with SATPLAN04 on many well-known real-world benchmark planning problems. Our experimental results show that our algorithm significantly outperforms SATPLAN04 on most of these problems, sometimes with an order of magnitude of improvement in running time.

1 Introduction and overview

A STRIPS planning problem involves arranging actions in order to accomplish a set of given tasks and objectives over multiple time steps. Planning problems can be solved based on different problem formulations. Over the past decade, planning as satisfiability (SAT-Plan) [9, 6] has emerged as one of the most effective formulations for optimal planning. Taking a contrarian's strategy against the common belief that there was no efficient algorithmic method for planning, SAT-Plan was proposed to take advantage of much celebrated progress made in the research of satisfiability over the years. The method of SAT-Plan first transforms a STRIPS planning problem into a satisfiability (SAT) problem, and then solves the SAT problem using a generic SAT solver.

Comparing to other planning paradigms, such as heuristic search and systematic search, the SAT-Plan approach has several salient advantages. First, by transforming a planning problem into a SAT, an extensively studied problem in computer science, we are able to utilize many of the powerful pruning and ordering methods developed for SAT. Second, this approach has a broad applicability because it can make use of any SAT algorithm. Third, unlike many heuristic methods that can hardly find optimal solutions, the SAT formulation can easily model time steps in a solution plan and be effectively used to find optimal plans.

Despite its success, however, the potential of the SAT-Plan paradigm has been neither fully explored nor exploited. The current best realization of SAT-Plan still has at least two limitations. First, by transforming a planning problem into a SAT and solving it in a "blackbox" fashion by a SAT solver, we completely discard the structural and goal information in the planning problem. In contrast, heuristic search-based planning methods typically combine directed forward search with backward chaining to effectively prune search space and explicitly exploit information in the planning goal.

Second, to achieve optimal (minimum) time steps, the current realization of SAT-Plan 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 approach has revealed crucial insights to this general planning paradigm. First, the SAT problem instances derived at different time steps share similar structures. Second, the knowledge learnt at shorter time steps can be accumulated and used to speed up the processes for solving SAT instances at longer time steps. Although some SAT solvers support incremental solving, the existing SAT planners do not utilize this feature. For each iteration, current SAT planners generate SAT problem instances independently from scratch and solve them in isolation so that knowledge learnt during previous iterations was lost.

To remedy these shortcomings, we develop a new approach for optimal STRIPS planning in this paper. We first propose a novel max-SAT formulation in which the original SAT formulation is maintained as hard constraints that cannot be violated, and furthermore, these constraints are coupled with an objective function to specify the goal of minimizing the number of time steps. Based on this max-SAT formulation, we develop a new general variable ordering heuristic that can improve the search efficiency for solving SAT. We also develop an accumulative learning scheme to collect and utilize the knowledge learnt from solving multiple SAT problems during an incremental planning process.

*This research was funded in part by NSF grants EIA-0113618 and IIS-0535257

2 A brief overview of SATPLAN04

SATPLAN04 [5] is the latest SAT-Plan system. Like its predecessors, i.e., SATPLAN [9] and BLACKBOX [6], SATPLAN04 accepts a planning problem in the STRIPS encoding.

SATPLAN04 aims at finding solutions with minimal steps of a parallel plan. In a parallel plan, two or more actions can be taken in parallel at one time step as long as they are not mutually exclusive. SATPLAN04 is optimal in that it can always find a parallel plan with the minimum number of time steps.

Given a planning problem, SATPLAN04 first applies the Graphplan approach [2] to construct a planning graph containing all facts and actions reachable from the initial state, and then encodes the resulting graph into a SAT formulation.

The default encoding for SATPLAN04 is the action-based encoding. Its central idea is to convert all fact variables to action variables. A fact at certain time step can be represented as a disjunction of the actions that add this fact at previous time steps. Action-based encoding then generates three classes of clauses based on action variables as follows:

1. *A clauses* – from actions' precondition constraints
   An action implies its preconditions
2. *E clauses* – from mutually exclusive constraints
   Two actions cannot hold true at the same time if they are mutually exclusive [2].
3. *G clauses* – from goal constraints
   A subgoal fact holding at time step k implies disjunction of the actions that add this fact at time step $k-1$.

In this paper, We call the variables in G clauses *G variables* and others *non-G variables*.

3 Planning in max-SAT formulation

To improve upon SAT-Plan strategy, we propose the following maximum satisfiability (max-SAT) formulation for STRIPS planning problems.

$$\begin{aligned} \text{objective} \quad &: \quad \text{maximize} \quad \sum_{k_0 \le t \le k_m, i=1\cdots n} w_t * g_{t,i} \\ \text{subject to} \quad &: \quad \text{SATPLAN constraints,} \end{aligned} \tag{1}$$

where $g_{t,i}$ takes value 1 if the i-th subgoal is fulfilled at time step t, or 0 otherwise; k_0 and k_m are, respectively, lower and upper bounds of the optimal number of steps; n is the number of subgoals in the planning problem; and w_t, for $k_0 \le t \le k_m$, are weights assigned to the goal-related variables at different time steps t. In order to ensure solution optimality, the weights w_t are assigned such that a weight for a smaller time step is exponentially larger than that for a larger time step. For example, in our current implementation, we use $w_t = n^{k_m - k_o - t}$ to force the satisfiability of all subgoals at a shallower time step prior to satisfy any subgoal at a longer time step. Finally, the constraints in (1) are directly from the SATPLAN formulation.

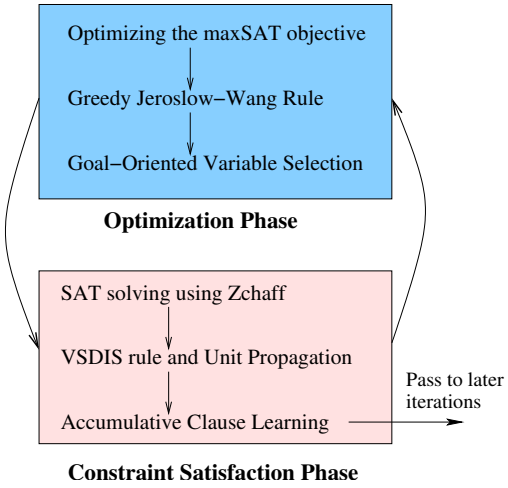

Figure 1: A sketch of main steps of the two-phase algorithm for solving the max-SAT problem.

Intuitively, the max-SAT formulation combines the SAT formulation with an objective function that aims at minimizing the length of a parallel plan. Optimizing the objective function in this formulation leads to effective variable selection methods that can better utilize the intrinsic structures of planning problems.

4 A two-phase strategy for max-SAT

Based on the new max-SAT formulation in (1), and in order to better use the structural information of planning problems, we develop what we call *goal oriented variable selection* (GOVS) heuristic that can be combined with an existing SAT algorithm to solve a max-SAT problem.

The main idea of GOVS is to facilitate *an alternating two-phase constrained optimization search* illustrated in Figure 1. In the *optimization phase*, we try to optimize the objective function. To this end, we select a variable, from the set of unassigned goal-related variables (i.e., G variables), to instantiate in order to minimize the objective function. In other words, we give a high priority to the G variables over the variables appeared only in SAT constraints. In the *constrained satisfaction phase*, we perform a chain of unit propagations to try to assign the remaining variables to satisfy all the SAT constraints using an existing SAT solver. We alternate between these two phases until an optimal solution is found or it is confirmed that not all SAT constraints can be satisfied within a predetermined upper bound on plan steps.

The selection and assignment of G variables are based on a greedy heuristic to optimize the objective function in (1). To choose a G variables to branch, we currently apply *Jeroslow-Wang rule* [4] to the G clauses.

The current default SAT solver in SATPLAN04 is the Siege algorithm [8], whose source code is unfortunately not available. In our implementation, we thus chose Zchaff [10], one of the best SAT solvers, as the SAT engine for SATPLAN04, and integrate our GVOS heuristic with Zchaff. The original decision heuristic in Zchaff is VSIDS [7]. We

integrated GOVS with VSIDS in our implementation, resulting in the following search procedure:

Zchaff-GOVS for solving max-SAT

1. Assign each literal (positive or negative) of every non-G variable a counter, initialized to 0.
2. If there exist uninstantiated G variables, select one from them according to the Jeroslow-Wang rule.
3. Otherwise (i.e., all G variables have been assigned), choose a non-G variable whose literal (either positive or negative) has the highest counter value.
4. instantiate the selected variable (from step 2 or 3) into true or false, simplify the CNF formula according to the variable assignment, and then apply unit propagation to the simplified formula.
5. Whenever a conflict occurs, apply the resolution rule to the conflicting clauses, record the newly generated learnt clauses into clause database, increase the counters of literals in the learnt clause, and backtrack to an early decision level that causes the conflict.
6. Periodically decay all the counters for non-G variables, e.g., divide all count values by 2 every 1000 steps of backtrack as in VSIDS.
7. If there remains an uninstantiated variable, go to step 2. Otherwise, terminate.

In the above procedure, step 2 corresponds to the optimization phase, and step 3 corresponds to the constraint satisfaction phase.

There are several advantages of our GVOS heuristic and two-phase max-SAT strategy. Our preliminary experimental analysis (data not shown) has indicated that after all the G variables have been fixed, the constraint part of the original planning problem can be solved quickly by unit propagation. This means that the G variables are the most critically constrained variables of a planning problem. By focusing on such critical variables, the overall search procedure is geared toward the regions of the search space where high quality solutions locate. In other words, GVOS leads to a focused search. GVOS can also be viewed as a heuristic for backward search because those variables directly related to goal (or objective) in the G clauses are considered first. Furthermore, goal related variables are often substantially fewer than all the variables in a planning problem; therefore, GVOS in many cases contributes to significant improvement of the search procedure. Finally, the GOVS heuristic can be easily integrated with a SAT solver to develop a max-SAT solver.

5 Accumulative Learning

As mentioned in Section 1, the problem structures over two consecutive iterations resemble each other except some G clauses because of the increase in time steps. We take advantage of such structural similarity and propose what we called *Accumulative Learning* (AL) scheme, which 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 meet the new constraint requirements for the next iteration. Therefore, the time for encoding can be substantially reduced. 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 use them in the next iteration. As a result, clauses that have to be learnt only need to be learnt once, which dramatically reduces running time. More specifically, the accumulative encoding and learning for time steps k can be described as follows:

Accumulative encoding and learning

1. Delete all G clauses for time step $k-1$; and delete any learnt clauses related to these G clauses (but retain other learnt clauses).
2. Add new G clauses for time step k.
3. Add all additional A and E clauses required for time step k.

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 effectively. Therefore, our accumulative learning scheme incurs very limited additional overhead, if any, over the SAT solver used.

6 Experimental evaluation and analysis

We experimentally evaluated the performance of the proposed planner. We adopted Zchaff as the SAT engine for SATPLAN04 and integrated the two-phase max-SAT algorithm and accumulative learning method with Zchaff.

To fully evaluate the proposed methods, we ran different combinations of the new methods and the original SATPLAN04 on problems in logistics-STRIPS domain [3] and benchmark instances used in the IPC4 Competition [1]. The specific algorithmic combinations that we compared include SATPLAN04 with the original Zchaff, with Zchaff+AL, with Zchaff+GOVS, and with Zchaff+GOVS+AL. We ran all the experiments on a PC workstation with Intel Xeon(TM) 2.4GHZ CPU and 2G memory. Since we considered optimal planning, i.e., these algorithms returned shortest plans, we evaluated and compared their running time. For all runs, we set a CPU time limit of 1,800 seconds.

Table 1 lists the number of instances of each planning domain that each algorithmic combination can solve within 1800 seconds. The combination of GOVS with AL solves the most number of instances for 5 (out of 8) planning domains.

Figure 2 shows a comparison of the solution times of the four algorithmic combinations on the same set of planning domains. The instances not solved in 1,800 seconds are considered unsolvable and are not shown in the graph. From Figure 2, it is evident that the two proposed methods can consistently improve upon the original SAT planner.

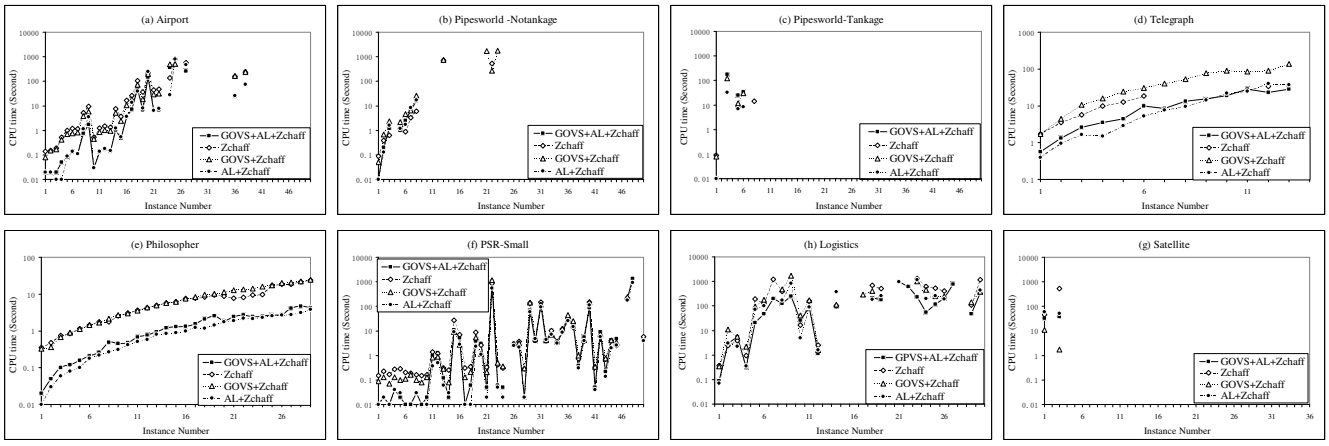

Figure 2: A comparison of the performance of SATPLAN04 with the original Zchaff, Zchaff+AL, Zchaff+GOVS, and Zchaff+GOVS+AL.

Domains	N	Zchaff	GOVS	AL	GOVS+AL
Airport	50	27	26	26	27
PW-Notankage	50	9	11	7	7
PW-Tankage	50	3	4	4	4
Telegraph	14	7	13	13	13
Philosopher	29	29	29	29	29
PSR-small	50	46	46	47	47
Satellite	36	2	2	2	2
Logistics	30	20	20	20	24

Table 1: The number of instances each method can solve. N is the totoal number of instances available in each domain. We highlight in boxes the methods that solve the most number of problems when there are differences among the four methods.

The GOVS method is particularly helpful in the Pipesworld-Notankage, Telegraph, Logistics, and Satellite domains. Zchaff+GOVS can solve several large problems in Pipesworld-Notankage domain, instances numbered 21, 22, and 23, in 1711, 271, 1779 seconds, respectively, whereas Zchaff spends 521 seconds on instance number 22 and fails on instances number 21 and 23. For the Telegraph domain, Zchaff+GOVS solves 13 problem instances, whereas Zchaff only finishes on 6. Zchaff+GOVS also significantly reduces the running time of Zchaff for the largest problems in Logistics and Satellite.

The AL method is found to be most useful for the Telegraph, Philosopher, PSR, and Logistics domains. For these domains, Zchaff+AL can significantly and consistently improve upon Zchaff on all problem instances. The improvement made by AL can be more than one order of magnitude on the Telegraph and Philosopher domains. An interesting observation is that combining GOVS with AL does not seem to have significant effects except for some individual instances. On some cases, combining these two may even lead to worse performance. We plan to study the interactions of these two methods in the future.

7 Conclusions

We have proposed a novel maximum satisfiability (max-SAT) formulation for optimal STRIPS planning and developed a general max-SAT solver to planning which exploit a goal-oriented variable ordering scheme and accumulative learning strategy. We have integrated our max-SAT approach with SATPLAN04 and Zchaff, and performed an extensive experimental analysis on planning benchmarks from IPC4 and other domains. Experimental results show that our new approach can significantly and consistently outperform the previous SAT-Plan methods across different planning domains.

References

[1] ICAPS 2004. Fourth international planing competition. *http://ls5-www.cs.uni-dortmund.de/ edelkamp/ipc-4/*, 2004.

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

[3] http://www.cs.washington.edu/homes/kautz/satplan/blackbox/blackbox-download.html.

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

[5] H. Kautz. Satplan04: Planning as satisfiability. IPC-04 Publication.

[6] H. Kautz and B. Selman. Unifying sat-based and graph-based planning. In *Proceedings of IJCAI-99*, pages 318–325, 1999.

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

[8] L. Ryan. Efficient algorithms for clause-learning SAT solvers. Master's thesis, Simon Fraser University, 2003.

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

[10] L. Zhang, C. F. Madigan, M. W. Moskewicz, and S. Malik. Efficient conflict driven learning in boolean satisfiability solver. In *ICCAD*, pages 279–285, 2001.