Lazy Compilation in Classical Planning (Extended Abstract)

Zuzana Fílová, Pavel Surynek

Czech Technical University in Prague
Faculty of Information Technology, Thákurova 9, 160 00 Praha 6, Czechia
{filovzu1,pavel surynek}@fit.cvut.cz

Abstract

Classical planning is a task of finding a sequence of actions that achieve a given goal. One of many approaches to classical planning is compilation into propositional satisfiability (SAT). In this work, we propose a new method that uses lazy compilation into SAT. Different from the standard compilation method, lazy compilation constructs the target propositional formula step by step while the SAT solver is consulted at each step and refinements of the formula are suggested according to SAT solver's answers. The performed experiments pointed out that lazy compilation has the potential to improve the performance of the planners.

Introduction and Background

Planning is an offline process that selects and sorts available actions to achieve a given goal, taking into account expected results of actions without executing them (Fikes and Nilsson 1971; Ghallab, Nau, and Traverso 2004).

The task in planning is to find a sequence of actions (plan) that achieve a given goal. It is a triple $\mathcal{P} = (A, I, G)$, where A represents actions, I is an initial state, a set of ground atoms, and G is a goal, a set of ground literals. An action is a triple $(prec, \mathit{eff}^+, \mathit{eff}^-)$, where $prec$ is a precondition, atoms that must hold in a state before the action can be applied, eff^+ and eff^- are positive and negative effects respectively, the atoms that are added and removed to/from the state after the action is applied.

The original search-based techniques for classical planning often struggled with a large search space (Korf 1987; Currie and Tate 1990) which led to the development of numerous heuristics (Bonet and Geffner 2001; Haslum, Bonet, and Geffner 2005).

Planning graphs and the related Graphplan algorithm (Blum and Furst 1995) is an important milestone that revived planning via a stream of neoclassical techniques that make it possible to solve significantly larger problems. Planning graphs also significantly contributed to compilation-based approaches. The original idea of compiling planning as propositional satisfiability (SAT) (Cook 1971; Biere et al. 2009) was coined by Kautz and Selman 1992. It compiles the planning task into a series of propositional formulae that are answered by the SAT solver (Audemard and Simon 2018). This enables using efficient search, propagation, and learning techniques from SAT solvers for planning.

There are many SAT-based planners including Blackbox (Kautz and Selman 1998), SATPlan (Kautz, Selman, and Hoffmann 2006), Madagascar (Rintanen 2014), or a planner for SAS+ formalism (Huang, Chen, and Zhang 2012). However, SAT-based planners are no longer among the top performers in the International Planning Competition. After a period of their greatest success, better performing planners based on different principles have emerged - such as Delphi (Katz et al. 2018) and Fast Downward (Helmert 2006).

Although planning as SAT may seem outdated, there is still some potential for the future. The great advantage of converting planning to SAT is that any progress in SAT solving also automatically means improvement in planner performance. Thus, it may happen that existing algorithms are improved or new SAT solving algorithms are discovered, which will improve there planners as well.

The *lazy approach* problem solving and the *lazy compilation*, stemming from *counterexample-guided generation and refinement* - CEGAR (Clarke 2003) or Bender's decomposition (Benders 2005), has been successfully applied in domain-dependent planning. For example, in solving the problem of Multi-Agent Path Finding (MAPF) (Silver 2005; Surynek 2019; Gange, Harabor, and Stuckey 2019). There is therefore opportunity for further research and the question of whether the lazy approach could be successfully applied in a more general area such as classical planning.

In this work we propose a new method that uses a step-by-step lazy compilation of classical planning into SAT.

Lazy SAT-based Compilation in Planning

By the classical compilation we mean a procedure in which for a given length of plan n we compile the bounded planning problem, that is the question whether a plan of length n exists, at once into the propositional formula $\Phi(n)$.

If $\Phi(n)$ is satisfiable, the plan π is extracted from the truth values of the propositional variables. Otherwise n is increased (usually by one) to allow for longer plans.

In the case of lazy compilation, only a partial specification of the given planning problem is encoded into formula $\Phi'(n)$ in the first phase. The plan π' decoded from the truth values of the variables of $\Phi'(n)$ hence may not be valid.

Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

In some cases, not all actions from π' can be applied sequentially and/or the states obtained by applying the actions may not contain all the atoms from the goal state. The plan π' hence must be checked and, in case of errors being found, new clauses are added to the formula to eliminate the errors followed by consulting the SAT solver again.

Lazy SAT Encoding

We used parallel encoding based on planning graphs (Kautz, McAllester, and Selman 1996) where some constraints are omitted. The encoding uses the following rules, where a_i indicates the action performed in the action layer A_i and p_i is an atom from the predicate layer P_i of the planning graph:

1. All atoms from the initial state are *true* in layer P_0 and all atoms from the goal state must be *true* in the layer P_n:
$$\bigwedge_{p\in I} p_0 \wedge \bigwedge_{p\notin I} \neg p_0 \wedge \bigwedge_{p\in G} p_n$$
2. Operators imply their preconditions. For each action a in the layer A_i there is a formula:
$$a_i \implies \left(\bigwedge_{p\in prec(a)} p_{i-1} \right)$$
3. Each atom in layer P_i implies the disjunction of all the actions at previous action layer A_{i-1} that have it as an positive effect. For each atom p in the layer P_i there is a sub-formula:
$$p_i \implies \left(\bigvee_{a\in A_{i-1} \mid p_i\in eff^+(a))} a_{i-1} \right)$$

Keeping the Completeness

This encoding is not complete because it lacks the rules that actions imply their effects and also mutually exclusive actions are not forbidden. This can admit invalid solutions that contain conflicting actions. The following propositions applie to planning problem $\mathcal{P}$:

$$\Phi'(n) \text{ is satisfiable } \;\not\Longrightarrow\; \mathcal{P} \text{ has a plan of length } n$$

$$\Phi'(n) \text{ is not satisfiable } \;\Longrightarrow\; \mathcal{P} \text{ has no plan of length } n$$

All possible errors in the plan are caused by dependent actions in some layer of the layered plan π. For the next iteration, clauses eliminating the detected pairs of dependent actions a, b are added to the formula $\Phi'(n)$ and the SAT solver is consulted again: $\neg a \vee \neg b$.

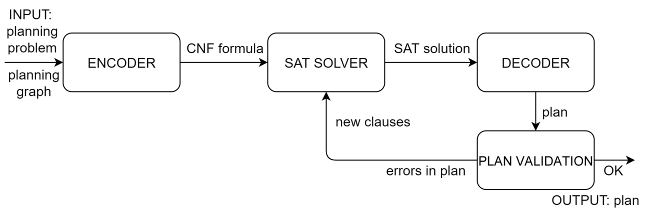

Figure 1: A scheme of the lazy SAT-based classical planner.

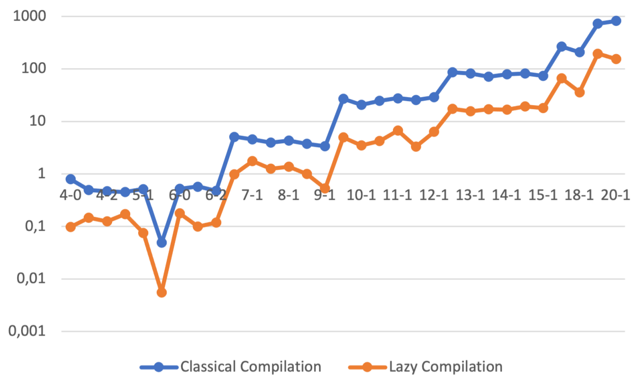

Figure 2: Time of solving Logistics domain problems for our planner with different types of compilation (the y-axis shows time in seconds on the logarithmic scale, and the x-axis shows individual problems from the Logistics domain).

Experimental Evaluation

A planner using two compilation variants - the proposed method for lazy compilation and classical compilation - was implemented to evaluate the proposed method. The planner was tested on planning problems from the International Planning Competition (Gerevini et al. 2009). A total of 79 problems of varying difficulty from four domains (Blocks World, Logistics, ZenoTravel, Mystery) were used. The experiments focused mainly on comparing the total time needed to solve planning problems and the parameters of the formula $\Phi'(n)$ - the number of variables and clauses.

Results suggest some advantages of lazy compilation method: the time-consuming construction of mutexes can be eliminated when constructing the planning graph which had a positive effect on the overall time. The results from the Logistics domain are shown in the Figure 2. The lazy planner was able to solve 63 from 79 problems faster than the classical planner.

It was found that in lazy compilation, much smaller formulas were sufficient to solve the problem (usually between 2 - 20 % of the number of classic compilation clauses). In some cases, the difference was significant (e.g. for the Mystery02 problem - 70,000 clauses in $\Phi'(n)$ compared to 4.8 million in $\Phi(n)$).

The disadvantage of lazy compilation turned out to be that more time is usually needed to solve the encoded SAT instances. However this difference was mostly negligible in terms of overall time except several Blocks World problems where a large increase in time has been observed.

Conclusion

In this work we propose a new method that uses a lazy compilation of the classical planning into SAT. The performed experiments pointed out the advantages and possible disadvantages of lazy compilation. The results of the experiments indicate that the use of lazy compilation has the potential to improve the performance of SAT-based planners.

Acknowledgments

This research has been supported by GAČR - the Czech Science Foundation, grant registration number 22-31346S.

References

Audemard, G.; and Simon, L. 2018. On the Glucose SAT Solver. *Int. J. Artif. Intell. Tools*, 27(1): 1840001:1–1840001:25.

Benders, J. F. 2005. Partitioning procedures for solving mixed-variables programming problems. *Comput. Manag. Sci.*, 2(1): 3–19.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2009. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

Blum, A. L.; and Furst, M. L. 1995. Fast Planning Through Planning Graph Analysis. *Artificial Intelligence*, 90(1): 1636–1642.

Bonet, B.; and Geffner, H. 2001. Planning as heuristic search. *Artif. Intell.*, 129(1-2): 5–33.

Clarke, E. M. 2003. SAT-Based Counterexample Guided Abstraction Refinement in Model Checking. In *Automated Deduction - CADE-19, 19th International Conference on Automated Deduction Miami Beach, 2003, Proceedings*, volume 2741 of *Lecture Notes in Computer Science*, 1. Springer.

Cook, S. A. 1971. The Complexity of Theorem-Proving Procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, 151–158. ACM.

Currie, K.; and Tate, A. 1990. Using domain knowledge to restrict search in an AI planner. In *Proceedings of the First International Conference on Expert Planning Systems, 1990*, 186–190. AAAI.

Fikes, R.; and Nilsson, N. J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence, 1971*, 608–620. William Kaufmann.

Gange, G.; Harabor, D.; and Stuckey, P. J. 2019. Lazy CBS: Implicit Conflict-Based Search Using Lazy Clause Generation. In *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS 2018*, 155–162. AAAI Press.

Gerevini, A.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artif. Intell.*, 173(5-6): 619–668.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. The Morgan Kaufmann Series in Artificial Intelligence. Elsevier Science. ISBN 9781558608566.

Haslum, P.; Bonet, B.; and Geffner, H. 2005. New Admissible Heuristics for Domain-Independent Planning. In *Proceedings, The Twentieth National Conference on Artificial Intelligence, 2005*, 1163–1168. AAAI Press / The MIT Press.

Helmert, M. 2006. The Fast Downward Planning System. *J. Artif. Intell. Res.*, 26: 191–246.

Huang, R.; Chen, Y.; and Zhang, W. 2012. SAS+ Planning as Satisfiability. *J. Artif. Intell. Res.*, 43: 293–328.

Katz, M.; Sohrabi, S.; Samulowitz, H.; and Sievers, S. 2018. Delfi: Online planner selection for cost-optimal planning. *IPC-9 planner abstracts*, 57–64.

Kautz, H.; McAllester, D.; and Selman, B. 1996. Encoding plans in propositional logic. *KR*, 96: 374–384.

Kautz, H.; and Selman, B. 1998. BLACKBOX: A new approach to the application of theorem proving to problem solving. In *AIPS98 workshop on planning as combinatorial search*, volume 58260, 58–60. sn.

Kautz, H.; Selman, B.; and Hoffmann, J. 2006. SatPlan: Planning as satisfiability. In *5th international planning competition*, volume 20, 156.

Kautz, H. A.; and Selman, B. 1992. Planning as Satisfiability. In Neumann, B., ed., *10th European Conference on Artificial Intelligence, ECAI 92, Vienna, Austria, August 3-7, 1992. Proceedings*, 359–363. John Wiley and Sons.

Korf, R. E. 1987. Planning as Search: A Quantitative Approach. *Artif. Intell.*, 33(1): 65–88.

Rintanen, J. 2014. Madagascar: Scalable planning with SAT. *Proceedings of the 8th International Planning Competition (IPC-2014)*, 21.

Silver, D. 2005. Cooperative Pathfinding. In *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference, 2005*, 117–122. AAAI Press.

Surynek, P. 2019. Unifying Search-based and Compilation-based Approaches to Multi-agent Path Finding through Satisfiability Modulo Theories. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019*, 1177–1183.