DPLL(MAPF): an Integration of Multi-Agent Path Finding and SAT Solving Technologies

Martin Čapek, Pavel Surynek
Faculty of Information Technology, Czech Technical University in Prague Thákurova 9, 160 00 Praha 6, Czechia
{capekma9, pavel.surynek}@fit.cvut.cz

Abstract

The task in multi-agent path finding (MAPF) is to find non-conflicting paths connecting agents' start and goal positions. The MAPF problem is often compiled to Boolean satisfiability (SAT) and solved by existing SAT solvers. Contemporary compilation approaches of MAPF to SAT regard the SAT solver as an external tool whose task is to return an assignment of all decision variables of a Boolean model of the input MAPF instance. We present in this paper a novel compilation scheme called DPLL(MAPF) in which the consistency checking of partial assignments of decision variables with respect to the MAPF rules is integrated directly into the SAT solver. This scheme allows for far more automated compilation where the SAT solver and the consistency checking procedure work together simultaneously to create the Boolean model and to search for its satisfying assignment.

Introduction

In this paper, we focus on improving lazy encoding schemes for solving the *multi-agent path finding* problem (MAPF) by compilation to Boolean satisfiability (SAT) (Biere et al. 2009). MAPF is a planning task defined on an undirected graph $G = (V, E)$ in which the task is to navigate agents across edges from their start vertices to individual goal vertices in a collision free way (Silver 2005; Ryan 2008; Sharon et al. 2015; Li et al. 2020). Both optimal and sub-optimal (de Wilde, ter Mors, and Witteveen 2013) MAPF solvers for various cumulative objectives such as *makespan* (Surynek 2012) or *sum-of-costs* (Sharon et al. 2013) exist.

Contemporary state-of-the-art sum-of-costs optimal SAT-based solver for MAPF, SMT-CBS (Surynek 2019), uses the SAT solver as a black box. The SAT solver is given a formula encoding the MAPF instance and is asked to provide a complete assignment of all Boolean variables. We suggest here an extension that allows the SAT solver to interact more intensively with the MAPF encoding part.

DPLL(T) (Nieuwenhuis, Oliveras, and Tinelli 2006; Katz et al. 2016) is a framework for integrating the SAT solver with a decision procedure for the conjunctive fragment of some complex theory T from which we inspired ourselves. The two components of DPLL(T) together form a decision procedure for general problems in theory T. Our contribution is adaptation of the idea of DPLL(T) for MAPF. We present DPLL(MAPF), that is DPLL(T) where theory T is substituted by a theory defining MAPF movement rules.

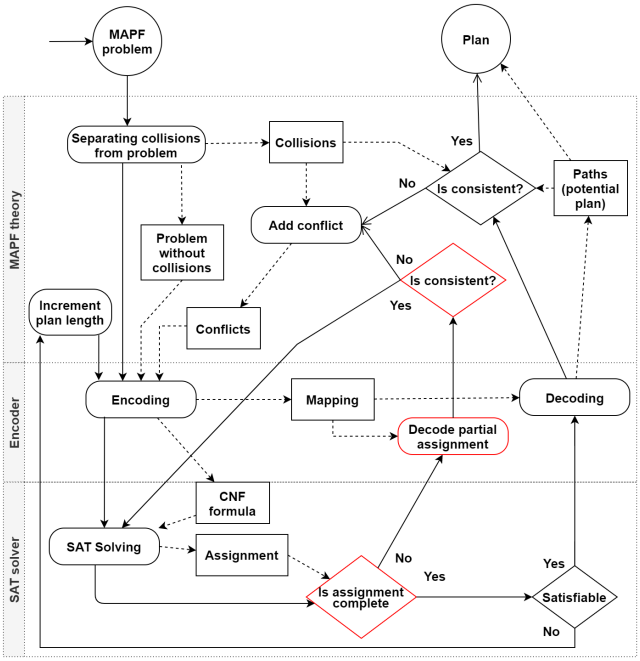

Figure 1: Scheme of DPLL(MAPF) consisting of SMT-CBS scheme and consistency checking (marked red) against MAPF rules.

Lazy Encoding: SMT-CBS

SMT-CBS is an optimal SAT-based solver employing the idea of encoding MAPF as a Boolean formula **lazily**. The lazy encoding is formalized through the concept of *incomplete Boolean model* defined as follows.

Definition 1 (incomplete model). *Propositional formula $\mathcal{H}(\xi)$ is an* incomplete Boolean model *of MAPF Σ iff $\mathcal{H}(\xi)$ is satisfiable $\Leftarrow$ Σ has a solution of sum-of-costs ξ.*

In an incomplete Boolean model $H(\xi)$ we do not specify all constraints defining the movement rules of MAPF.

	UNI 10	EXP 2	UNI 5	UNI 3	EXP 1.6	EXP 1.4
Empty 16x16	-0.21	-0.29	-0.30	-0.23	-0.18	-0.23
Empty 32x32	20.63	-9.05	-2.34	-1.71	-10.13	-11.12
Maze 32x32	194.82	28.32	48.58	16.21	8.81	4.72
Warehouse 100x63	-13.57	17.66	-23.44	4.53	-17.99	-16.92
Sum	201.67	36.64	22.51	18.81	-19.49	-23.56
Percentage	-32.09%	-7.91%	-5.01%	-4.22%	4.79%	5.84%

Table 1: Sums of $TIME_{DPLL(MAPF)} - TIME_{SMT-CBS}$ in seconds and percentage improvement counted as $(TIME_{DPLL(MAPF)}/TIME_{SMT-CBS}) - 1$. Columns are sorted by row Sum. Negative numbers are highlighted.

Specifically in SMT-CBS, see figure 1, we are initially creating an incomplete model by omitting collision avoidance constraints while constraints that ensure that agents move across edges, do not skip, disappear etc. are always included in the model. Hence, the answer from the SAT solver needs to be checked for collisions. If this consistency check is successfully passed, we can return the valid MAPF solution otherwise the incomplete model needs to be refined with constrains eliminating detected collisions. For the details of encoding in term of decision variables and clauses see (Surynek et al. 2016).

SAT Solver + MAPF = DPLL(MAPF)

The drawback of SMT-CBS is that it needs to wait for the SAT solver to produce assignment of all decision variables of the model even if the assignment eventually causes a collision. The SAT solving phase is due to its exponential-time search the most expensive part of MAPF compilation-based solving process. Hence it could be helpful to detect the MAPF rule violations in the early stages of SAT solving.

We suggest to check the MAPF rules that are not encoded directly into formula $\mathcal{H}(\xi)$ not only after obtaining the complete assignment but also during the SAT solving process where only partial assignments are available. This improvement of SMT-CBS is captured in Figure 1 as the red elements.

The core idea consists in adding the MAPF consistency rule check of partial assignment of incomplete models. In contrast to SMT-CBS, the SAT solver can no longer be a separate procedure in DPLL(MAPF) scheme but rather the MAPF specific reasoning part and the SAT solver are integrated into a single solver. The MAPF rule consistency check is integrated directly into the conflict analysis within the backtracking mechanism of the standard CDCL SAT solver (Silva and Sakallah 1996).

Proposition 1 *DPLL(MAPF) is a sound sum-of-costs optimal MAPF algorithm.*

Experimental Evaluation

We implemented DPLL(MAPF)[1] in C++ via integrating the MAPF consistency check directly into the Glucose 4 SAT solver (Audemard and Simon 2018) which effectively changed the SAT solver into a MAPF solver.

[1]Available on https://github.com/CapekM/DPLL_MAPF.

The MAPF rule consistency check turned out to be relatively expensive computation so it seems we cannot afford to perform it after every Boolean variable assignment. Hence, in our evaluation, we check the MAPF rule consistency in regular intervals using two strategies:

- **Uniform *N*** - check consistency after each *(#V)/N* Boolean variables are assigned.
- **Exponential *N*** - check consistency after each $100 * N^x$ Boolean variables are assigned, where *N* is our parameter and *x* is a number that starts at 0 and increment after each check. The number 100 was chosen by simple experiment.

We compared DPLL(MAPF) with SMT-CBS which has been reimplemented in C++ so it shares the code with the DPLL(MAPF) implementation [2].

The maps were taken from `movingai.com` (Sturtevant 2012). On each map, we generated 2..30 agents and each instance was run 10x with different random initial and goal positions of agents. From these 10 runs were averaged (Only Warehouse map was shorten, and instances at maximum 16 agents) and are shown in Table 1.

Although DPLL(MAPF) is not a clear winner over SMT-CBS, it demonstrated its potential as often the runtime is significantly reduced in comparison with SMT-CBS.

Conclusion

We suggested a new innovative step in the SAT-based solving of the multi-agent path finding problem (MAPF) in which the SAT solver and the MAPF reasoning part are closely integrated. The collision resolution in solutions obtained from partial assignments of incomplete Boolean models is integrated directly in the SAT solver's search loop which effectively turns the SAT solver into a MAPF solver. We demonstrated in our evaluation that DPLL(MAPF) has a potential to outperform previous SAT-based MAPF solvers.

The impact of DPLL(MAPF) is reaching further as we can use the same framework for implementing DPLL(MAPF_R), that is a SAT-based solver for the continuous variant of MAPF (Andreychuk et al. 2019). Even DPLL(classical planning) seems to be possible and could represent a contribution to SAT-based classical planners (Ghallab, Nau, and Traverso 2004; Froleyks, Balyo, and Schreiber 2019).

Acknowledgement. This work has been supported by the Czech Science Foundation - GAČR (project no. 19-17966S).

[2]Tests were done on a system with Core i7 CPU, 16 GB RAM, under Windows 10.

References

Andreychuk, A.; Yakovlev, K. S.; Atzmon, D.; and Stern, R. 2019. Multi-Agent Pathfinding with Continuous Time. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao*, 39–45. ijcai.org.

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

Biere, A.; Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T. 2009. *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. IOS Press. ISBN 1586039296, 9781586039295.

de Wilde, B.; ter Mors, A.; and Witteveen, C. 2013. Push and rotate: cooperative multi-agent path planning. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2013*, 87–94. IFAAMAS.

Froleyks, N. C.; Balyo, T.; and Schreiber, D. 2019. PASAR - Planning as Satisfiability with Abstraction Refinement. In *Proceedings of the Twelfth International Symposium on Combinatorial Search, SOCS 2019*, 70–78. AAAI Press.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated planning - theory and practice*. Elsevier. ISBN 978-1-55860-856-6.

Katz, G.; Barrett, C. W.; Tinelli, C.; Reynolds, A.; and Hadarean, L. 2016. Lazy proofs for DPLL(T)-based SMT solvers. In *2016 Formal Methods in Computer-Aided Design, FMCAD 2016*, 93–100. IEEE.

Li, J.; Gange, G.; Harabor, D.; Stuckey, P. J.; Ma, H.; and Koenig, S. 2020. New Techniques for Pairwise Symmetry Breaking in Multi-Agent Path Finding. In Harabor, D.; and Vallati, M., eds., *Proceedings of the Thirteenth International Symposium on Combinatorial Search, SOCS 2020, Online Conference [Vienna, Austria], 26-28 May 2020*, 129–130. AAAI Press.

Nieuwenhuis, R.; Oliveras, A.; and Tinelli, C. 2006. Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *J. ACM* 53(6): 937–977. doi:10.1145/1217856.1217859. URL https://doi.org/10.1145/1217856.1217859.

Ryan, M. R. K. 2008. Exploiting Subgraph Structure in Multi-Robot Path Planning. *J. Artif. Intell. Res.* 31: 497–542. doi:10.1613/jair.2408. URL https://doi.org/10.1613/jair.2408.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artif. Intell.* 219: 40–66.

Sharon, G.; Stern, R.; Goldenberg, M.; and Felner, A. 2013. The increasing cost tree search for optimal multi-agent pathfinding. *Artif. Intell.* 195: 470–495.

Silva, J. P. M.; and Sakallah, K. A. 1996. Conflict Analysis in Search Algorithms for Satisfiability. In *Eigth International Conference on Tools with Artificial Intelligence, ICTAI 1996*, 467–469. IEEE Computer Society.

Silver, D. 2005. Cooperative Pathfinding. In *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference, June 1-5, 2005, Marina del Rey, California, USA*, 117–122. AAAI Press.

Sturtevant, N. R. 2012. Benchmarks for Grid-Based Pathfinding. *IEEE Trans. Comput. Intell. AI Games* 4(2): 144–148.

Surynek, P. 2012. Towards Optimal Cooperative Path Planning in Hard Setups through Satisfiability Solving. In *PRICAI 2012: Trends in Artificial Intelligence - 12th Pacific Rim International Conference on Artificial Intelligence, Kuching, Malaysia, September 3-7, 2012. Proceedings*, volume 7458 of *Lecture Notes in Computer Science*, 564–576. Springer.

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. ijcai.org.

Surynek, P.; Felner, A.; Stern, R.; and Boyarski, E. 2016. Efficient SAT Approach to Multi-Agent Path Finding Under the Sum of Costs Objective. In *ECAI 2016 - 22nd European Conference on Artificial Intelligence*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, 810–818. IOS Press.