Conceptual Comparison of Compilation-based Solvers for Multi-Agent Path Finding: MIP vs. SAT

Pavel Surynek
Czech Technical University in Prague, Faculty of Information Technology, Thaákurova 9, 160 00 Praha 6, Czechia
pavel.surynek@fit.cvut.cz

Abstract

The task in multi-agent path finding (MAPF) is to find paths through which agents can navigate from their starting positions to given individual goal positions. The combination of two additional requirements makes the problem challenging: (i) agents must not collide with each other and (ii) the paths must be optimal with respect to some objective. We summarize and compare main ideas of contemporary compilation-based solvers for MAPF using MIP and SAT formalisms.

Introduction

Compilation is one of the most prominent techniques in computing. In the context of problem solving, compilation is represented by reduction of an input problem instance from its source formalism to a different usually well established formalism for which an efficient solver exists. The key idea behind using compilation in problem solving is that the solving process benefits from the advancements in solvers for the target formalism.

The compilation-based solving approach has been applied successfully in solving combinatorial problems like *plannnig* (Ghallab, Nau, and Traverso 2004), *verification* (Bradley and Manna 2007), or *scheduling* (Blazewicz, Brauner, and Finke 2004) where the target formalism is often represented by *Boolean satisfiability* (SAT) (Barrett et al. 2009), *mixed integer linear programming* (MIP) (Jünger et al. 2010; Rader 2010), *answer set programming* (ASP) (Lifschitz 2019), or *constraint satisfaction* (CSP) (Dechter 2003). Significant advancements have been achieved in compilation for specific domains, namely in *multi-agent path finding* (MAPF) (Silver 2005; Ryan 2008; Standley 2010) that we are focusing on in this paper.

The standard variant of MAPF is the problem of finding collision-free paths for a set of agents from their starting positions to individual goal positions. Agents move in an environment which is usually modeled as an undirected graph $G = (V, E)$, where vertices represent positions and edges the possibility of moving between positions. Agents in this abstraction are discrete items, commonly denoted $A = \{a_1, a_2, ..., a_k\}, k \leq |V|$, placed in vertices of the graph, moving instantaneously between vertices via edges provided that there is always at most one agent in a vertex and no two agents traverse an edge in opposite directions.

Relatively simple formulation of MAPF is important factor that made it an accessible target of various solving methods including compilation-based approaches. Contemporary state-of-the-art compilation-based solvers for MAPF go even beyond the standard single shot **reduction-solving-interpreting** loop coined for classical planning by the SAT-Plan algorithm (Kautz and Selman 1992) where Boolean satisfiability has been used as the target formalism and the SAT solver has been treated merely as a black-box solver. Intensive cross-fertilization between various methods for MAPF and compilation techniques led to numerous improvements in encodings and ways how the target solver is used, treating it less like a black-box.

The effort culminated recently in a combination of compilation and lazy conflict resolution introduced in Conflict-based search algorithm (CBS) (Sharon et al. 2015) resulting in approaches that construct the target encoding lazily in close cooperation with the solver. The solver in these lazy schemes suggests solutions for incomplete encodings of the input instance that do not specify it fully. After checking the interpreted solution against original specification, that is, if it is a valid MAPF solution, the high-level part of the MAPF solver suggests refinement of the encoding and the process is repeated. This scheme has been implemented using MIP (Gange, Harabor, and Stuckey 2019; Lam et al. 2019) and SAT (Surynek 2019) as target formalisms.

An Overview of Target Formalisms

Mixed integer linear programming (MIP). A linear program (LP) is a finite list of linear inequalities plus linear objective, the task is to minimize the objective such that the inequalities hold. Geometrically the inequalities define a polytope and the objective function a gradient so the task is to find some boundary point of the polytope that minimizes the gradient. Formally, the task is to minimize $c^\top x$ subject to $Px \leq b$, $x \geq 0$, where x is a real vector representing the decision variables, P is a matrix of coefficients of linear inequalities, and b is another real-valued vector.

Since in discrete decision problems like MAPF it is not much convenient to use fractional assignments of decision variables, some or all decision variables are often declared to be integers making LP an integer program (IP) or mixed integer program (MIP) respectively.

Boolean satisfiability (SAT) problem consists in deciding whether there exists a truth-value assignment of variables that satisfies a given Boolean formula. The formula is often specified using the *conjunctive normal form* (CNF), which is a conjunction of clauses where each clause is a disjunction of literals, and a literal is either a variable or its negation.

Most modern SAT solvers are based on the *conflict-driven clause learning* algorithm (Silva and Sakallah 1999; Eén and Sörensson 2003; Audemard and Simon 2018) that implements constraint propagation and back-jumping techniques known from CSP. The significant challenge when SAT is used as the target formalism is bridging the original representation of the problem and the yes/no environment of SAT. The lack of expressiveness is balanced by thr efficiency of SAT solvers.

Lazy Compilation

Conflict-based search (CBS) is currently the most popular approach for MAPF. It is due to its elegant idea which enables to implement the algorithm relatively easily.

From the compilation perspective, the CBS algorithm should be understood as a lazy method that tries to solve an under-specified problem and relies to be lucky to find a correct solution even using this incomplete specification. There is another mechanism that ensures soundness of this lazy approach, the branching scheme. If the CBS algorithm is not lucky, that is, the candidate solution is incorrect in terms of MAPF rules, then the search branches for each possible refinement of the discovered MAPF rule violation and the refinement is added to the problem specification in each branch. Concretely, the MAPF rule violations are conflicts of pairs of agents such as a collision of $a_i \in A$ and $a_j \in A$ in v at time step t and the refinements are conflict avoidance constraints for single agents in the form that $a_i \in A$ should avoid v at time step t (for a_j analogously).

While in CBS the branching scheme and the refinements must be explicitly implemented, in the compilation-based approach we can eliminate the MAPF rule violation by adding a new constraint into the problem specification and leave branching to the solver for the target formalism. In this way, the encoding of MAPF (or any other problem) is built dynamically and eventually may end up by the complete specification of the problem as done in MDD-SAT. However a solution or a proof of that is does not exists is often found for incomplete specification, that is, well before all constraints are added to the encoding.

Surprisingly intuitive explanation why this is possible comes from the geometry of linear programming. The finite set of inequalities define a polytope of feasible solutions as an intersection of half-spaces. Optimal feasible solution is often an element of some of the planes defining the polytope but not an element of all of them (in other words some inequalities are satisfied because optimal solution is deep inside their half-space). Hence, to specify the optimal solution one does not need all the constraints. Similarly if there is no solution, the polytope is empty. Again an empty polytope may be obtained by intersecting only some of the half-spaces.

Beyond Simple Time Expansion

MIP as the target formalism allows for reasoning about both integer and real-valued decision variables which opens opportunities to use decision variables with completely different meaning than in SAT.

SAT-based approaches assume Boolean decision variables for each copy of a vertex for a relevant time step (Surynek et al. 2016), that is, the underlying graph G is expanded for every relevant timestep and corresponding Boolean variables are introduced. Constraints modeling MAPF rules are expressed on top of these variables. As already mentioned some constraints are introduced in a lazy style.

In contrast to this, the model suggested in (Lam et al. 2019) considers a large but finite **pool of paths** $\Pi(a_i)$ for each agent $a_i \in A$ connecting its start and goal vertex. Decision variables determine the proportion of path $\pi \in \Pi(a_i)$ being selected by the agent. Constraints ensure that agents use at least one path and the overall cost of path is minimized.

On the other hand, the MIP-based approach compared to SAT-based compilation requires to deal with fractional solutions, which adds non-trivial complexity to the high level solving process.

MIP-based vs. SAT-based Compilation

MIP-based and SAT-based compilation schemes have different opportunities how to enhance each approach with MAPF specific heuristics and pruning techniques such as *symmetry breaking* (Li et al. 2020) or *mutex reasoning* (Zhang et al. 2020). In this regard, the MIP-based scheme is more open for integration of domain specific improvements as more decisions are made at the high-level, namely branching strategy where heuristics can be included, paths pool refinement that could further guide the search can be modified at the high level.

SAT-based approach still leaves lot of decisions on a general purpose SAT-solver into which it is difficult to include any MAPF specific heuristics without changing the implementation of the solver. On the other hand, the role of the solver is bigger as it solves without any intervention from the high-level the entire NP-hard component of the problem. In the MIP-based approach, the MIP solver solves independently a linear problem which is done in polynomial time while the hard exponential-time part is solved in cooperation with the high-level.

Conclusion

We summarized and compared main ideas of recent compilation-based approaches to MAPF, the MIP-based and SAT-based solvers. Our brief summary highlights important common features of both approaches such as lazy conflict elimination, but also focuses on significant differences such as the need in SAT to build time expansion of the underlying graph at the level of Boolean formula or the need in MIP to eliminate fractional values of decision variables.

Acknowledgment. The author has been supported by GAČR - the Czech Science Foundation, grant registration number 19-17966S.

References

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

Barrett, C. W.; Sebastiani, R.; Seshia, S. A.; and Tinelli, C. 2009. Satisfiability Modulo Theories. In Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds., *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, 825–885. IOS Press.

Blazewicz, J.; Brauner, N.; and Finke, G. 2004. Scheduling with Discrete Resource Constraints. In Leung, J. Y., ed., *Handbook of Scheduling - Algorithms, Models, and Performance Analysis*. Chapman and Hall/CRC.

Bradley, A. R.; and Manna, Z. 2007. *The calculus of computation - decision procedures with applications to verification*. Springer.

Dechter, R. 2003. *Constraint processing*. Elsevier Morgan Kaufmann.

Eén, N.; and Sörensson, N. 2003. An Extensible SAT-solver. In *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Selected Revised Papers*, volume 2919 of *Lecture Notes in Computer Science*, 502–518. Springer.

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.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated planning - theory and practice*. Elsevier.

Jünger, M.; Liebling, T. M.; Naddef, D.; Nemhauser, G. L.; Pulleyblank, W. R.; Reinelt, G.; Rinaldi, G.; and Wolsey, L. A., eds. 2010. *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*. Springer.

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

Lam, E.; Bodic, P. L.; Harabor, D. D.; and Stuckey, P. J. 2019. Branch-and-Cut-and-Price for Multi-Agent Pathfinding. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019*, 1289–1296. ijcai.org.

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 *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020*, 193–201. AAAI Press.

Lifschitz, V. 2019. *Answer Set Programming*. Springer.

Rader, D. 2010. *Deterministic Operations Research: Models and Methods in Linear Optimization*. Wiley. ISBN 9780470484517.

Ryan, M. R. K. 2008. Exploiting Subgraph Structure in Multi-Robot Path Planning. *J. Artif. Intell. Res.* 31: 497–542.

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

Silva, J. P. M.; and Sakallah, K. A. 1999. GRASP: A Search Algorithm for Propositional Satisfiability. *IEEE Trans. Computers* 48(5): 506–521.

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

Standley, T. S. 2010. Finding Optimal Solutions to Cooperative Pathfinding Problems. In Fox, M.; and Poole, D., eds., *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010*. AAAI Press.

Surynek, P. 2019. Unifying Search-based and Compilation-based Approaches to Multi-agent Path Finding through Satisfiability Modulo Theories. In Kraus, S., ed., *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. An Empirical Comparison of the Hardness of Multi-Agent Path Finding under the Makespan and the Sum of Costs Objectives. In *Proceedings of the Ninth Annual Symposium on Combinatorial Search, SOCS 2016, Tarrytown, NY, USA, July 6-8, 2016*, 145–147. AAAI Press.

Zhang, H.; Li, J.; Surynek, P.; Koenig, S.; and Kumar, T. K. S. 2020. Multi-Agent Path Finding with Mutex Propagation. In *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020*, 323–332. AAAI Press.