Modifying Optimal SAT-Based Approach to Multi-Agent Path-Finding Problem to Suboptimal Variants

Pavel Surynek
National Institute of Advanced Industrial Science and Technology (AIST)
Tokyo 135-0064, Japan
pavel.surynek@aist.go.jp

Ariel Felner, Roni Stern
Ben Gurion University
Beer-Sheva, Israel 84105
felner,sternron@bgu.ac.il

Eli Boyarski
Bar-Ilan University
Ramat-Gan, Israel
eli.boyarski@gmail.com

Abstract

In *multi-agent path finding* (MAPF) the task is to find non-conflicting paths for multiple agents. Recently, a SAT-based approach was developed to solve this problem and proved beneficial in many cases when compared to other search-based solvers. In this paper, we introduce SAT-based unbounded- and bounded-suboptimal algorithms and compare them to relevant search-based algorithms.

1 Introduction

The *multi-agent path finding* (MAPF) problem consists of a graph $G = (V, E)$ and a set $A = \{a_1, a_2, \ldots a_k\}$ of k agents. Time is discretized into time steps. The arrangement of agents at time-step t is denoted as α_t. Each agent a_i has a start position $\alpha_0(a_i) \in V$ and a goal position $\alpha_+(a_i) \in V$. At each time step an agent can either *move* to an adjacent location or *wait* in its current location. The task is to find a sequence of move/wait actions for each agent a_i, moving it from $\alpha_0(a_i)$ to $\alpha_+(a_i)$ such that agents do not *conflict*, i.e., do not occupy the same location at the same time.

MAPF is usually solved aiming to minimize one of the two commonly-used global cumulative cost functions: **(1) Sum-of-costs** (denoted ξ) is the summation, over all agents, of the number of time steps required to reach the goal location (Standley 2010; Sharon *et al.* 2015). **(2) Makespan:** (denoted μ) is the time until the last agent reaches its destination (i.e., the maximum of the individual costs) (Surynek 2010).

Finding optimal solutions for both variants is difficult as the state-space grows exponentially with k (# of agents). Therefore, many suboptimal solvers were developed. Some suboptimal solvers aim to to quickly find paths for all agents while paying no attention to the quality of the solution, i.e., how far it is from the optimal solution (Silver 2005). These algorithms - also called *any solution* - are usually used when k is large and some of them are not complete.

In some cases, the user might ask for some guarantee on the quality of the solution returned. A common type of such a requirement is that the solution found is **bounded suboptimal**, that its cost is $\leq (1 + \epsilon) \times c_{opt}$ where c_{opt} is the cost of the optimal solution and ϵ is a parameter that sets the desired amount of suboptimality - sometimes called the *error*. A solver that returns bounded-suboptimal solutions is referred to as a **bounded-suboptimal** algorithm or more specifically $(1 + \epsilon)$-bounded suboptimal.

2 From Optimal to Suboptimal Solver

We introduce algorithms that are based on a SAT-based optimal MAPF algorithm (called MDD-SAT) for the sum-of-costs variant (Surynek *et al.* 2016). The main idea in MDD-SAT is to convert the optimization problem (finding minimal sum-of-costs) to a sequence of decision problems – is there a solution of a given sum-of-costs ξ. A formula $\mathcal{F}_\xi$ has been introduced such that $\mathcal{F}_\xi$ is satisfiable if and only if there is a solution of sum-of-costs ξ.

To verify that a solution to $\mathcal{F}_\xi$ represents a solution with sum of costs lower than ξ, a *cardinality constraint* is added (Bailleux and Boufkhad 2003). Cardinality constraint allows counting variables set to $TRUE$ in a formula. By mapping agents' actions to propositional variables, cardinality constraint can be used to bound a numeric cost. More information on this formula and its exact variables can be found in (Surynek *et al.* 2016).

To convert MDD-SAT to a suboptimal any solution algorithm, we simply remove the cardinality constraint from the construction of $\mathcal{F}_\xi$. Let $\mathcal{F}$ denote the resulting formula. Since $\mathcal{F}$ has all the constraints in $\mathcal{F}_\xi$ except the cardinality constraint, then clearly a satisfying assignment to $\mathcal{F}$ still represents a feasible solution (no collisions between agents etc.). Since $\mathcal{F}$ is less constrained than $\mathcal{F}_\xi$, we expect it to be solved faster. Indeed, we observed this in our preliminary experiments. Using $\mathcal{F}$ in MDD-SAT algorithm instead of $\mathcal{F}_\xi$ looses, however sum-of-cost optimality. The solver without cardinality constraint will be called uMDD-SAT.

The formula $\mathcal{F}_\xi$ can be relaxed not only by complete removal of the cardinality constraint but also by setting the cardinality constraint to impose a less restrictive cost bound. It can be shown that for any error ϵ we can set the cardinality constraint in the MDD-SAT algorithm so that the resulting solution will be $(1 + \epsilon)$-bounded suboptimal. The solver with this relaxation will be called eMDD-SAT.

3 Experimental Evaluation

We performed a set of experiments to evaluate uMDD-SAT and eMDD-SAT. We used various 4-connected grids as the

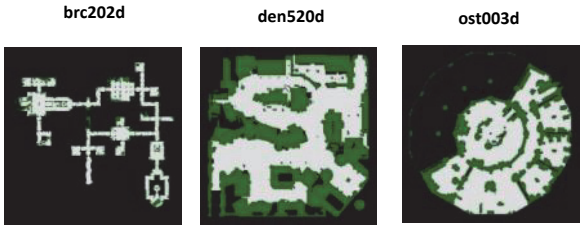

Figure 1: Dragon Age maps include: narrow corridors in `brc202d`, large open space in `den520d`, and open space with almost isolated rooms in `ost003d`.

underlying graphs.

Instances for our tests were based on three structurally different large maps taken from Sturtevant's repository (Sturtevant 2012). These are Dragon Age Origin (DAO) maps denoted as `brc202d`, `den520d`, and `ost003d` which are a standard benchmark for MAPF (see Figure 1). The number of agents was varied from 1 to 256 to obtain instances of various difficulties (the step ranged from 1 to 16) and 10 random instances were generated for each number of agents.

All tests were run on a machine with CPU Intel i7 3.2 Ghz, 8 GB RAM under Ubuntu Linux 15 and Windows 10 respectively. The timeout for all solvers has been set to 500 seconds.

We compared uMDD-SAT with two suboptimal algorithms: PUSH-AND-SWAP (Luna and Bekris 2011), which is a polynomial time rule-based algorithm unbounded by design, and ECBS (Barer *et al.* 2014) a bounded-suboptimal algorithm that is based on the CBS MAPF solver (Sharon *et al.* 2015) where we set the suboptimality bound to a very large number.

Runtime results shown in Figure 2 (runtimes for individual instances are sorted) suggest that in the unbounded setup ECBS is faster than other two algorithms with a minor exception on harder instances (requiring more runtime) where PUSH-AND-SWAP wins.

Results for the bounded variant (with $\epsilon = 0.01$) indicate that in easier instances containing fewer agents ECBS is faster. However with the increasing difficulty of instances and density of agents the gap in performance is narrowed until eMDD-SAT starts to perform better in harder instances. This trend is best visible on the `ost003d` map.

4 Conclusions

We demonstrated how to modify a SAT-based optimal MAPF solver to suboptimal variants both unbounded and bounded. Especially promising results were obtained for a bounded variant eMDD-SAT on harder instances where the learning mechanism of underlying SAT solver gains advantage over search-based solver ECBS.

5 Acknowledgements

This paper is supported by a project commissioned by the New Energy and Industrial Technology Development Organization Japan (NEDO) and the joint grant of the Israel Ministry of Science and the Czech Ministry of Education Youth and Sports number 8G15027.

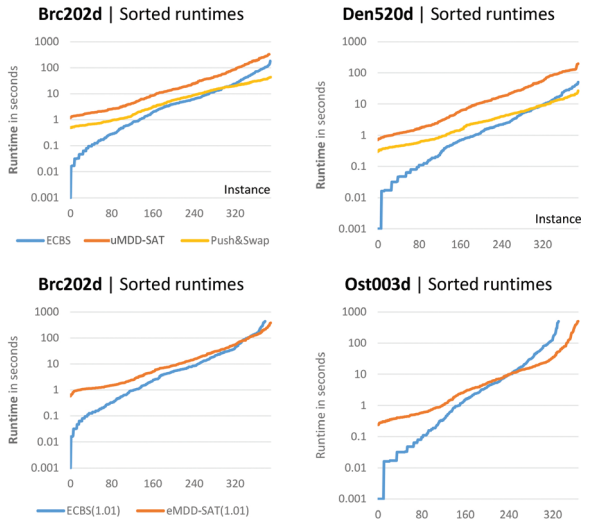

Figure 2: Runtimes of unbounded and bounded algorithms ($\epsilon = 0.01$) on DAO maps `brc202d`, `den520d`, and `ost003d`.

References

O. Bailleux and Y. Boufkhad. Efficient CNF encoding of boolean cardinality constraints. In *CP*, pages 108–122, 2003.

Max Barer, Guni Sharon, Roni Stern, and Ariel Felner. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *Symposium on Combinatorial Search (SoCS)*, 2014.

R. Luna and K. E. Bekris. Push and swap: Fast cooperative path-finding with completeness guarantees. In *IJCAI*, pages 294–300, 2011.

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

D. Silver. Cooperative pathfinding. In *AIIDE*, pages 117–122, 2005.

T. Standley. Finding optimal solutions to cooperative pathfinding problems. In *AAAI*, pages 173–178, 2010.

Nathan R. Sturtevant. Benchmarks for grid-based pathfinding. *Computational Intelligence and AI in Games*, 4(2):144–148, 2012.

Pavel Surynek, Ariel Felner, Roni Stern, and Eli Boyarski. Efficient SAT approach to multi-agent path finding under the sum of costs objective. In *ECAI*, pages 810–818, 2016.

P. Surynek. An optimization variant of multi-robot path planning is intractable. In *AAAI*, 2010.