Mutex Propagation for SAT-based Multi-Agent Path Finding

Pavel Surynek[1 [0000−0001−7200−0542]], Jiaoyang Li[2], Han Zhang[2], T. K. Satish Kumar[2], and Sven Koenig[3]

[1] Faculty of Information Technology, Czech Technical University in Prague
Thákurova 9, 160 00 Praha 6, Czechia
pavel.surynek@fit.cvut.cz
[2] University of Southern California, Henry Salvatori Computer Science Center
941 Bloom Walk, Los Angeles, USA
{jiaoyanl,skoenig,zhan645}@usc.edu
[3] University of Southern California, Information Sciences Institute
4676 Admiralty Way, Marina del Rey, USA
tkskwork@gmail.com

Abstract. Multi-agent path finding (MAPF) is the problem of planning a set of non-colliding paths for a set of agents so that each agent reaches its individual goal location following its path. A mutex from classical planning is a constraint forbidding a pair of facts to be both true or a pair of actions to be executed simultaneously. In the context of MAPF, mutexes are used to rule out the simultaneous occurrence of a pair of agents in a pair of locations, which can prune the search space. Previously mutex propagation had been integrated into conflict-based search (CBS), a major search-based approach for solving MAPF optimally. In this paper, we introduce mutex propagation into the compilation-based (SAT-based) solver MDD-SAT, an alternative to search-based solvers. Our experiments show that, despite mutex propagation being computationally expensive, it prunes the search space significantly so that the overall performance of MDD-SAT is improved.

Keywords: multi-agent path finding, mutex propagation, satisfiability solving (SAT)

1 Introduction

Multi-agent path finding (MAPF) instance is specified by a graph $G = (V,E)$ and a set of k agents $\{a_1 \ldots a_k\}$, where agent a_i has start location $s_i \in V$ and goal location $g_i \in V$. Time is discretized into time steps. Between successive timesteps, each agent can either *move* to an adjacent empty location or *wait* in its current location. Both move and wait actions incur a cost of one, unless the agent terminally waits at its goal location, which does not incur any costs. A *path* for agent a_i is a sequence of move and wait actions that lead agent a_i from location s_i to location g_i. A *conflict* happens when two agents are at the same location at the same timestep (called *a vertex conflict*) or traverse the same edge in opposite directions at the same timestep (called *an edge conflict*). The

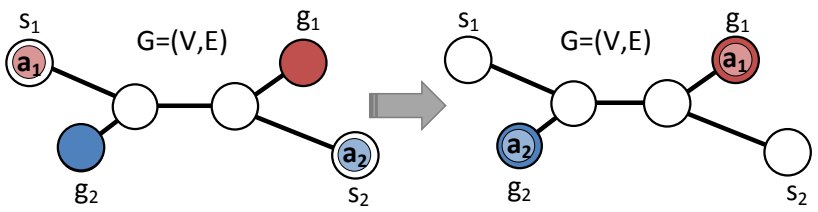

Fig. 1. A MAPF instance with two agents.

objective is to find a set of conflict-free paths for all agents while minimizing the sum of the costs of these paths (called *sum-of-costs*). Although MAPF is NP-hard to solve optimally [19], it has many real-world applications, such as warehouse robots [18] and quadrotor swarms [7].

A MAPF instance is illustrated in Figure 1. One of the agents (say a_1) has to wait until the other agent (a_2) reaches its goal location since the agents would otherwise collide in the middle section of G.

Zhang et al. [20] introduced *mutex propagation*, a technique originally from artificial intelligence planning, to MAPF and used it to identify and efficiently resolve symmetric conflicts occurred in *conflict-based search* (CBS) [11], a popular search-based algorithm that solves MAPF optimally. They used mutex propagation to capture reachability information for agent pairs and automatically generate symmetry-breaking constraints, resulting in a branching rule for CBS that is capable of outperforming both CBS [6] and CBS with handcrafted symmetry-breaking constraints [9].

In this paper, we demonstrate the power of mutex propagation in the framework of MDD-SAT [16], a popular SAT-based algorithm that can also solve MAPF optimaly. MDD-SAT expands the underlying graph G in time resulting in a structure analogous *planning graph* [4]. A Boolean formula whose satisfying assignments represents MAPF solutions up to the certain sum-of-costs is then built using the expansion and consulted to the SAT solver [2].

2 Background

We first introduce mutex propagation in classical planning and in MAPF as well as CBS with mutex propagation.

Mutex propagation has its origin in classical planning. It is a form of constraint propagation that corresponds to directed 3-consistency, which in turn is a truncated form of path consistency [17]. Like all constraint propagation techniques, it makes implicit constraints explicit, and it does so efficiently. Applied to the planning graph [4], mutex propagation tightly approximates the set of all states reachable from a given start state in polynomial time [17]. Therefore, it has been successfully used to design reachability heuristics for plan-space and state-space planners [10] and to improve SAT-based planners [8].

A mutex can be regarded as a constraint forbidding a pair of facts to be both *TRUE* or a pair of actions to be executed simultaneously.

States in classical planning are expressed as finite sets of ground logic atoms. Actions change states via set operations. Formally, a classical planning action is a triple of sets of atoms (p, e^+, e^-), called *precondition*, *positive effects*, and *negative effects* respectively. The action is applicable in state S represented as a finite set of atoms iff $p \subseteq S$, and the result of the application is the new state $S' = (S \setminus e^-) \cup e^+$.

We say that actions (p_1, e_1^+, e_1^-) and (p_2, e_2^+, e_2^-) applicable in state S are *dependent*, a basic form of mutex, iff $(p_1 \cup e_1^+) \cap e_2^- \neq \emptyset$ or $(p_2 \cup e_2^+) \cap e_1^- \neq \emptyset$. In other words, actions conflict in their precondition and effects. The mutex relation can be inductively propagated further to atoms: We say that atoms q_1 and q_2 are mutex iff all pairs of actions producing them are pair wise mutex and there is no action that produces both q_1 and q_2 [4]. Similarly, two actions (p'_1, e'^+_1, e'^-_1) and (p'_2, e'^+_2, e'^-_2) in the next timestep are mutex iff **(i)** they are dependent or **(ii)** there are atoms $r_1 \in p'_1$ and $r_2 \in p'_2$ that are mutex.

We show one possible representation of MAPF instance, with the *move-to-unoccupied* rule using classical planning actions. Action $move(a,u,v)$ with $a \in A$ and $u, v \in V$, that moves agent a from location u to location v can be defined as $(p_{move}, e^+_{move}, e^-_{move})$ (a,u,v) and action $wait(a,u)$, that makes agent a wait in location u, can be defined as $(p_{wait}, \emptyset, \emptyset)$ (a,u) where:

$$
\begin{aligned}
p_{move}(a,u,v) &= \{empty(v), at(a,u)\},\\
e^+_{move}(a,u,v) &= \{empty(u), at(a,v)\},\\
e^-_{move}(a,u,v) &= \{empty(v), at(a,u)\},\\
p_{wait}(a,u) &= \{at(a,u)\}.
\end{aligned}
$$

This representation of MAPF actions naturally rules out vertex conflicts through the dependence of actions (a pair of agents can collide in a vertex only through a pair of dependent/mutex actions). This first level of mutexes is covered in the compilation-based MAPF formulations by a constraint permitting at most one agent per vertex. However, the next levels of mutexes obtained through propagation were omitted in previous compilation-based approaches, such as MDD-SAT and others [15].

2.1 Mutex Propagation in MAPF

Multi-valued decision diagrams (MDDs) for MAPF were introduced in [12]. An MDD of $l+1$ levels for agent a_i (denoted MDD_i) is a directed acyclic graph that consists of all paths of cost at most l for agent a_i. The nodes at level t in the MDD correspond to all locations at timestep t on these paths. A node v^t in MDD_i represents a copy of location v at level t, while a directed edge (u^t, v^{t+1}) in MDD_i represents a copy of edge $\{u,v\} \in E$ between level t and $t+1$. There are only one node at level 0 and one node at level $l+1$, which correspond to the start location s_i and the goal location g_i, respectively, of agent a_i.

MDDs are constructed for single agents and essentially capture reachability information similar to planning graphs in classical planning. Therefore, mutex propagation

[4] No operation actions (noops) are usually assumed for each atom q where $noop(q) = (\{q\}, \{q\}, \emptyset)$.

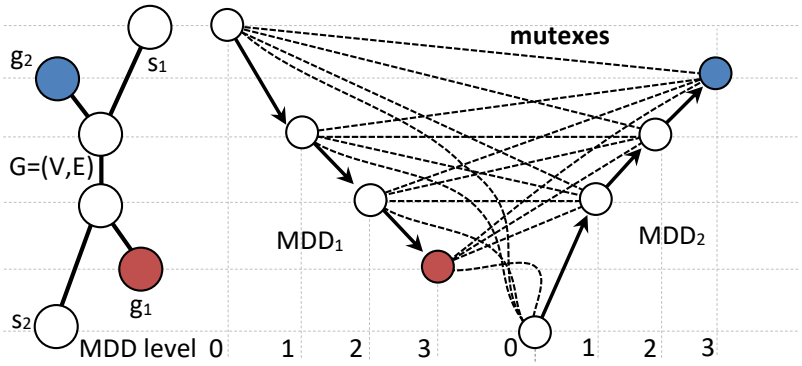

Fig. 2. MDDs for a_1 and a_2 from Figure 1 for sum-of-costs = 3. Mutexes form a biclique between nodes of MDD_1 and MDD_2.

is used on MDDs in [20] to capture reachability information for agent pairs. Two MDD nodes n_i and n_j at the same level t of the MDDs of two different agents a_i and a_j are *mutex* iff any path that moves agent a_i from its start location s_i at timestep 0 to the location of n_i at timestep t has at least one conflict with any path that moves agent a_j from its start location s_j at timestep 0 to the location of n_j at timestep t. The definition of MDD edges being mutex is similar. Any pair of MDD nodes (edges) that corresponds to a vertex (edge) conflict is mutex (called *initial mutex*), and any pair of MDD nodes (edges) whose incoming edges (source nodes) are pairwise mutex is also mutex (called *propagated mutex*).

2.2 Conflict-Based Search with Mutex Propagation

Conflict-based search (CBS) is an optimal two-level search-based MAPF algorithm. The low level plans shortest paths for all agents (each one ignoring the path of other agents) that satisfy the spatio-temporal constraints introduced by the high level, while the high level searches in a *constraint tree* (CT) and resolves the conflicts between agents by adding constraints. The root CT node contains no constraints and a shortest path on graph G for each agent. When expanding a CT node N, CBS first checks for conflicts among the paths of N. If the paths are conflict-free, CBS terminates and returns the paths. Otherwise, CBS chooses a conflict and resolves it by generating two child CT nodes that inherit the constraints and the paths of N. For each child CT node, CBS first adds a new constraint that prohibits one of the agents from using the conflicting vertex/edge at the conflicting timestep and then calls the low-level search to replan a path for this agent. CBS guarantees optimality by performing best-first searches on both its high and low levels.

CBS was improved by using MDDs with mutex propagation in [20] to identify and efficiently resolve *cardinal conflicts*, a type of conflict where all pairs of shortest paths of the two conflicting agents have at least one vertex or edge conflict. CBS can try many combinations of these shortest paths, which results in many CT nodes expansions. CBS using MDDs with mutex propagation, on the other hand, immediately identifies such

cardinal conflicts and generate two sets of constraints (instead of two single constraints) to resolve them with a single CT node expansion. It significantly outperforms both CBS [6] and CBS with handcrafted symmetry-breaking constraints [9].

3 SAT-Based Approach: MDD-SAT

MDD-SAT first builds MDDs for individual agents with regard to given sum-of-costs ξ. The number of levels μ for the MDDs is $\mu = \mu_0 + (\xi - \xi_0)$ [16], where μ_0 is a lower bound on the *makespan* calculated as the maximum length of the shortest individual paths of the agents from their start locations to their goal locations and ξ_0 is a lower bound on the sum-of-costs calculated as the sum of lengths of the shortest individual paths.

A propositional variable is introduced for each node and edge in the MDDs. We use $\mathcal{X}_v^t(a_i)$ to denote the variables corresponding to nodes and $\mathcal{E}_{u,v}^t(a_i)$ to denote the variables corresponding to edges. The meaning of the variables reflects the correspondence between the existence of directed paths connecting from the start nodes to the goal nodes in the MDDs and the existence of a solution to the MAPF instance. $\mathcal{X}_v^t(a_i)$ is *TRUE* iff agent a_i is in location $v \in G$ at timestep t and $\mathcal{E}_{u,v}^t(a_i)$ is *TRUE* iff agent a_i moves from location u to location v at timestep t.

The MAPF movement rules are encoded as clauses using these variables. Satisfying assignments correspond to directed paths in MDDs due to the following constraints for all agents $a \in A$ and locations $u, v \in V$ and timestep $t \in \{0, 1, ..., \mu - 1\}$. Intuitively, the constrains express that, if an agent is in a location at timestep t, it must leave via exactly one directed edge and appear in the target location of the edge at timestep $t+1$:

$$\mathcal{X}_u^t(a_i) \Rightarrow \bigvee_{v^{t+1} \mid (u^t, v^{t+1}) \in MDD_i} \mathcal{E}_{u,v}^t(a_i) \tag{1}$$

$$\sum_{v^{t+1} \mid (u^t, v^{t+1}) \in MDD_i} \mathcal{E}_{u,v}^t(a_i) \leq 1 \tag{2}$$

$$\mathcal{X}_v^{t+1}(a_i) \Rightarrow \bigvee_{u^t \mid (u^t, v^{t+1}) \in MDD_i} \mathcal{E}_{u,v}^t(a_i) \tag{3}$$

$$\sum_{u^t \mid (u^t, v^{t+1}) \in MDD_i} \mathcal{E}_{u,v}^t(a_i) \leq 1 \tag{4}$$

$$\mathcal{E}_{u,v}^t(a_i) \Rightarrow \mathcal{X}_u^t(a_i) \wedge \mathcal{X}_v^{t+1}(a_i) \tag{5}$$

The following constraint is defined for all locations $v \in V$ and all timesteps $t \in \{0, 1, ..., \mu\}$:

$$\sum_{a_i \in A \mid v^t \in MDD_i} \mathcal{X}_v^t(a_i) \leq 1 \tag{6}$$

Constraints (1-5) ensure that paths in the MDDs are represented. Constraints (1) and 2 state that if an agent is in a location it must leave it via exactly one outgoing edge.

Similarly, constraints (3) and (4) state that, if an agent is in a location then it must arrive there via exactly one incoming edge. Constraints (6) encode conflict avoidance. It states that at most one agent is in a location v at a timestep t.

All pseudo-Boolean *at-most-one* constraints are expressed as clauses. One possible representation of this constraint is a clique of binary clauses that forbid agents a_i and a_j to be in location v at timestep t: $\neg\mathcal{X}_v^t(a_i) \vee \neg\mathcal{X}_v^t(a_i)$.

Finally, the bound on the number of edge variables set to *TRUE* can be represented by a *cardinality constraint* [3, 13] that ensures that the sum-of-costs of the represented solutions is at most ξ (see [16] for details). Altogether, we construct a Boolean formula $\mathcal{F}(\xi)$ that is satisfiable iff the MAPF has a solution with sum-of-costs ξ. Due to the construction of $\mathcal{F}(\xi)$, a MAPF solution can be read off from a satisfying truth-value assignment of $\mathcal{F}(\xi)$, which can be found by an off-the-shelf SAT solver [2].

Solution to the MAPF instance with minimal sum-of-costs corresponds to the first satisfiable Boolean formula in $\mathcal{F}(\xi_0)$, $\mathcal{F}(\xi_0+1)$, ... since the satisfiability of $\mathcal{F}(\xi)$ is monotonic in ξ.

3.1 Mutexes in SAT-Based Solver

Mutex propagation can be integrated into the MDD-SAT solver at the stage of construction of the MDDs. The knowledge of which nodes are mutex in MDDs can be reflected in the construction of $\mathcal{F}(\xi)$ via binary clauses.

Assume that nodes n_i and n_j the MDDs for agents a_i and a_j respectively are mutex at timestep t. Then, we add clause $\neg\mathcal{X}_{n_i}^t(a_i) \vee \neg\mathcal{X}_{n_j}^t(a_j)$ to $\mathcal{F}(\xi)$, which directly follows the definition of two nodes being mutex by stating that a_i and a_j cannot be in the locations of nodes n_i and n_j respectively at timestep t.

To understand the role of mutexes in Boolean formulae, we study them in relation to *unit propagation* (UP) [5], a standard technique implemented in SAT solvers. UP is a form of resolution inference that extends the partial consistent assignment of truth values to Boolean variables. In a clause, in which all literals but one are *FALSE* by the partial assignment, the last literal must be *TRUE* for the clause to be *TRUE*.

When mutexes are expressed as binary clauses, setting $\mathcal{X}_u^t(a_i)$ to *TRUE* enables UP to infer that $\mathcal{X}_v^t(a_j)$ is *FALSE* due to clause $\neg\mathcal{X}_u^t(a_i) \vee \neg\mathcal{X}_v^t(a_j)$. Sometimes the same inference can be made without having the mutex clause.

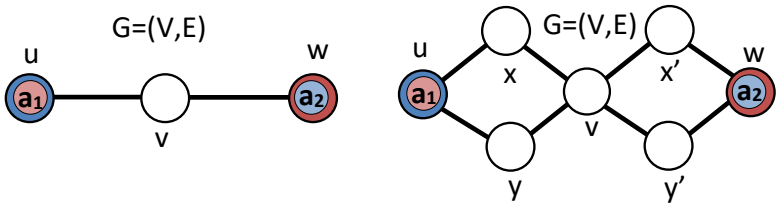

Fig. 3. Unit propagation cannot simulate mutex propagation due to branching.

Consider a corridor consisting of three locations u, v, and w. Agents a_1 and a_2 are trying to move through the corridor from opposite directions (Figure 3 left). After

constructing the MDDs for 2 timesteps we have the Boolean variables $\mathcal{X}_u^0(a_1)$, $\mathcal{X}_v^1(a_1)$, $\mathcal{X}_w^2(a_1)$, $\mathcal{X}_w^0(a_2)$, $\mathcal{X}_v^1(a_2)$, and $\mathcal{X}_u^2(a_2)$ as well as the corresponding variables for the edges (omitted) (Figure 4). There is a conflict at timestep 1 since the agents would collide in location v. Mutex propagation hence discovers that agent a_1 cannot be in location w at timestep 2 if agent a_2 is in location u (timestep 2) resulting in the mutex clause $\neg\mathcal{X}_w^2(a_1) \vee \neg\mathcal{X}_u^2(a_2)$. Setting $\mathcal{X}_w^2(a_1)$ to *TRUE* propagates to setting $\mathcal{X}_u^2(a_2)$ to *FALSE* via UP. However, the same effect can be achieved without having the mutex clause.

Assume again that $\mathcal{X}_w^2(a_1)$ is set to *TRUE*. Through $\neg\mathcal{X}_w^2(a_1) \vee \mathcal{E}_{v,w}^1(a_1)$ (3) via UP, $\mathcal{E}_{v,w}^1(a_1)$ is set to *TRUE*. Then, through $\neg\mathcal{E}_{v,w}^1(a_1) \vee \mathcal{X}_v^1(a_1)$ (5) via UP, $\mathcal{X}_v^1(a_1)$ is set to *TRUE*. Through $\neg\mathcal{X}_v^1(a_1) \vee \mathcal{X}_v^1(a_2)$ (6) via UP, $\mathcal{X}_v^1(a_2)$ is set to *FALSE*. Through $\neg\mathcal{E}_{v,u}^1(a_2) \vee \mathcal{X}_v^1(a_2)$ (5) $\mathcal{E}_{v,u}^1(a_2)$ is set to *FALSE*. Finally, through $\neg\mathcal{X}_u^2(a_2) \vee \mathcal{E}_{v,u}^1(a_2)$ (3) via UP, $\mathcal{X}_u^2(a_2)$ is set to *FALSE* (we note that different propagation chains can be used to obtain the same outcome).

We were quite fortunate in the above example because all relevant clauses were binary, and UP propagates very well through them. In general, however, UP is less powerful than mutex propagation, as formalized in the following proposition:

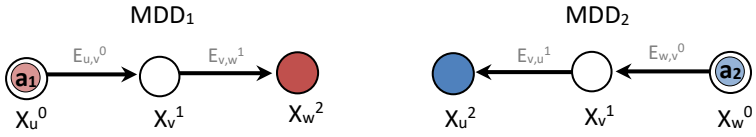

Fig. 4. Mutex propagation simulated by unit propagation.

Proposition 1. *Formula $\mathcal{F}(\xi)$ built using MDDs with mutex propagation allows for strictly stronger Boolean constraint propagation than the formula without mutexes.*

Proof. Consider a corridor with branching (Figure 3 right) consisting of locations u, x, y, (x and y form the first branch) v, x', y', (x' and y' form the second branch), and w. Two agents a_1 and a_2 start to move in opposite directions from u and w respectively leading to a conflict at location v at timestep 2. After building MDDs, we obtain the Boolean variables representing agents' locations $\mathcal{X}_u^0(a_1)$, $\mathcal{X}_x^1(a_1)$, $\mathcal{X}_y^1(a_1)$, $\mathcal{X}_v^2(a_1)$, $\mathcal{X}_{x'}^3(a_1)$, $\mathcal{X}_{y'}^3(a_1)$, and $\mathcal{X}_w^4(a_1)$ for agent a_1 and $\mathcal{X}_w^0(a_2)$, $\mathcal{X}_{x'}^1(a_2)$, $\mathcal{X}_{y'}^1(a_2)$, $\mathcal{X}_v^2(a_2)$, $\mathcal{X}_x^3(a_2)$, $\mathcal{X}_y^3(a_2)$, and $\mathcal{X}_u^4(a_2)$ for agent a_2 (see Figure 5).

Mutex propagation starts at the conflict at timestep 2 from which it discovers mutexes between agent a_1 being at locations x or y and agent a_2 being at locations x' or y' at time step 3, expressed by mutex clauses $\neg\mathcal{X}_{x'}^3(a_1) \vee \neg\mathcal{X}_x^3(a_2)$, $\neg\mathcal{X}_{x'}^3(a_1) \vee \neg\mathcal{X}_y^3(a_2)$, $\neg\mathcal{X}_{y'}^3(a_1) \vee \neg\mathcal{X}_x^3(a_2)$, and $\neg\mathcal{X}_{y'}^3(a_1) \vee \neg\mathcal{X}_y^3(a_2)$. Then, due to having no pair of mutex free actions delivering agents to their goal locations at timestep 4, mutex propagation infers a mutex between agent a_1 being at location w and agent a_2 being at location u at timestep 4, expressed by a mutex clause $\neg\mathcal{X}_w^4(a_1) \vee \neg\mathcal{X}_u^4(a_2)$.

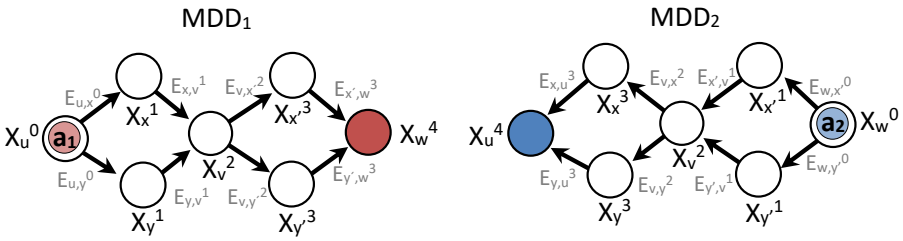

Fig. 5. MDDs consisting of 4 timesteps for agents a_1 and a_2 from Figure 3.

If $\mathcal{X}_w^4(a_1)$ is set to *TRUE*, then UP sets $\mathcal{X}_u^4(a_2)$ to *FALSE* due to the last mutex clause. Now assume again that $\mathcal{X}_w^4(a_1)$ is set to *TRUE* without having mutex clauses. $\mathcal{X}_w^4(a_1)$ occurs in clauses: $\neg\mathcal{X}_w^4(a_1) \vee \mathcal{E}_{x',w}^3(a_1) \vee \mathcal{E}_{y',w}^3(a_1)$ (3), $\neg\mathcal{E}_{x',w}^3(a_1) \vee \mathcal{X}_w^4(a_1)$ (5), and $\neg\mathcal{E}_{y',w}^3(a_1) \vee \mathcal{X}_w^4(a_1)$ (5). UP however does not propagate from this point towards timestep 3 since none of the affected clauses becomes unit. ■

3.2 Experimental Evaluation

We integrated mutex propagation into the existing C++ implementation of MDD-SAT, which uses the Glucose 3.0 SAT solver [1]. MDD-SAT with and without mutex propagation (denoted MDD-SAT-MTX and MDD-SAT respectively) were compared with respect to their runtime on a number of MAPF scenarios from movingai.com [14].

Success rate results are shown in Figures 6 and 7. We used 25 MAPF instances per number of agents having random start and goal locations. The success rate corresponds to the percentage of MAPF instances (out of 25) solved within the time limit of 300 seconds.

Mutex propagation is beneficial across all tested MAPF scenarios. Although a significant amount of time is spent on the mutex propagation itself, the resulting runtime savings during SAT solving phase will eventually result in overall shorter runtimes. Although the improvement cannot be considered a breakthrough in these preliminary experiments, our results demonstrate the promise of mutex propagation for SAT-based MAPF solvers.

4 Conclusion

We integrated *mutex propagation* into MDD-SAT, a sum-of-costs optimal SAT-based solver for MAPF. Our experimental evaluation showed that mutex propagation, despite it is substantial computing overhead, can improve the efficiency of MDD-SAT for several MAPF scenarios. Out theoretical analysis showed however, that mutex propagation in some cases is entailed by *unit propagation*, a built in Boolean constraint propagation in most modern SAT solvers. Hence, we plan to investigate in future research how to adapt MAPF encodings to simulate mutex propagation via unit propagation.

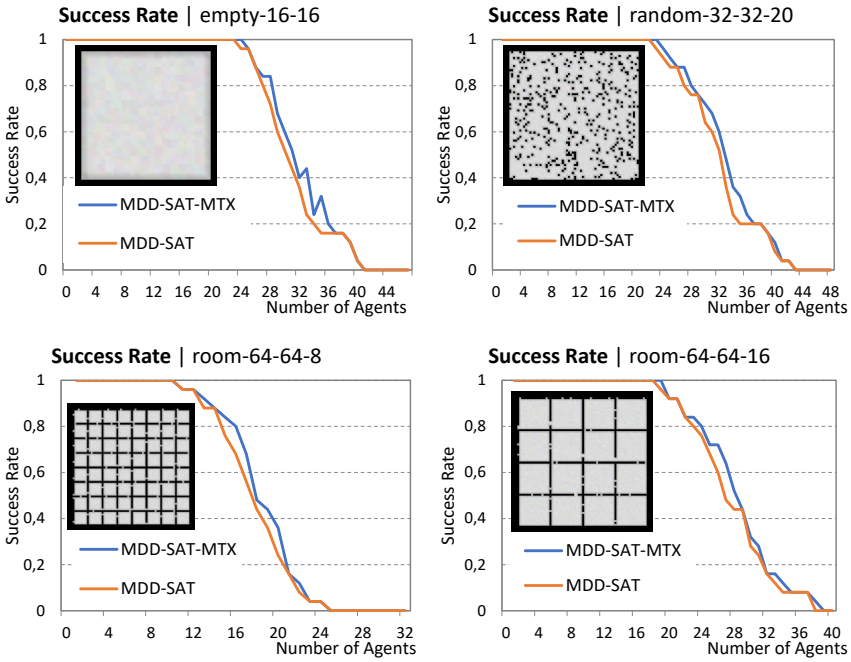

Fig. 6. Success rate experiments on small maps.

Acknowledgements

The research at the Czech Technical University in Prague was supported by GAČR - the Czech Science Foundation, under grant number 19-17966S. The research at the University of Southern California was supported by National Science Foundation (NSF) under grant numbers 1409987, 1724392, 1817189, 1837779, and 1935712 as well as a gift from Amazon. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the sponsoring organizations, agencies, or the U.S. government.

We would like to thank anonymous reviewers for their valuable comments.

References

1. Audemard, G., Lagniez, J., Simon, L.: Improving glucose for incremental SAT solving with assumptions: Application to MUS extraction. In: Theory and Applications of Satisfiability Testing - SAT 2013 - 16th International Conference, Helsinki, Finland, July 8-12, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7962, pp. 309–317. Springer (2013)
2. Audemard, G., Simon, L.: Predicting learnt clauses quality in modern SAT solvers. In: IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009. pp. 399–404 (2009)

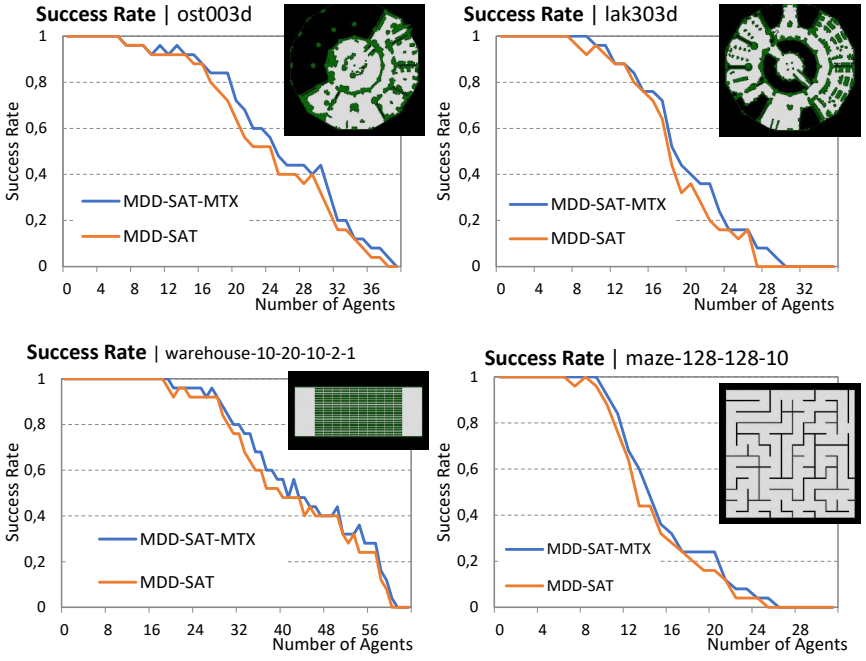

Fig. 7. Success rate experiments on large maps.

3. Bailleux, O., Boufkhad, Y.: Efficient CNF encoding of boolean cardinality constraints. In: Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2833, pp. 108–122. Springer (2003)
4. Blum, A., Furst, M.L.: Fast planning through planning graph analysis. Artificial Intelligence **90**(1-2), 281–300 (1997)
5. Dowling, W.F., Gallier, J.H.: Linear-time algorithms for testing the satisfiability of propositional horn formulae. Journal of Logic Programming **1**(3), 267–284 (1984)
6. Felner, A., Li, J., Boyarski, E., Ma, H., Cohen, L., Kumar, T.K.S., Koenig, S.: Adding heuristics to conflict-based search for multi-agent path finding. In: Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling, ICAPS 2018, Delft, The Netherlands, June 24-29, 2018. pp. 83–87. AAAI Press (2018)
7. Hönig, W., Preiss, J.A., Kumar, T.K.S., Sukhatme, G.S., Ayanian, N.: Trajectory planning for quadrotor swarms. IEEE Trans. Robotics **34**(4), 856–869 (2018)
8. Kautz, H.A., Selman, B.: Pushing the envelope: Planning, propositional logic and stochastic search. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, USA, August 4-8, 1996, Volume 2. pp. 1194–1201. AAAI Press / The MIT Press (1996)
9. Li, J., Surynek, P., Felner, A., Ma, H., Kumar, T.K.S., Koenig, S.: Multi-agent path finding for large agents. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI

2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019. pp. 7627–7634. AAAI Press (2019)
10. Nguyen, X., Kambhampati, S.: Extracting effective and admissible state space heuristics from the planning graph. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA. pp. 798–805. AAAI Press / The MIT Press (2000)
11. Sharon, G., Stern, R., Felner, A., Sturtevant, N.R.: Conflict-based search for optimal multi-agent pathfinding. Artificial Intelligene **219**, 40–66 (2015)
12. Sharon, G., Stern, R., Goldenberg, M., Felner, A.: The increasing cost tree search for optimal multi-agent pathfinding. Artificial Intelligence **195**, 470–495 (2013)
13. Sinz, C.: Towards an optimal CNF encoding of boolean cardinality constraints. In: Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3709, pp. 827–831. Springer (2005)
14. Sturtevant, N.R.: Benchmarks for grid-based pathfinding, http://www.movingai.com. Transactions on Computational Intelligence and AI in Games **4**(2), 144 – 148 (2012)
15. Surynek, P.: Time-expanded graph-based propositional encodings for makespan-optimal solving of cooperative path finding problems. Annals of Mathematics and Artificial Intelligence **81**(3-4), 329–375 (2017)
16. Surynek, P., Felner, A., Stern, R., Boyarski, E.: Efficient SAT approach to multi-agent path finding under the sum of costs objective. In: ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016). Frontiers in Artificial Intelligence and Applications, vol. 285, pp. 810–818. IOS Press (2016)
17. Weld, D.S.: Recent advances in AI planning. AI magazine **20**(2), 93–123 (1999)
18. Wurman, P.R., D'Andrea, R., Mountz, M.: Coordinating hundreds of cooperative, autonomous vehicles in warehouses. AI Magazine **29**(1), 9–20 (2008)
19. Yu, J.: Intractability of optimal multirobot path planning on planar graphs. IEEE Robotics Automation Letters **1**(1), 33–40 (2016)
20. Zhang, H., Li, J., Surynek, P., Koenig, S., Kumar, T.K.S.: 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. pp. 323–332. AAAI Press (2020)