Unifying Search-based and Compilation-based Approaches to Multi-agent Path Finding through Satisfiability Modulo Theories

Pavel Surynek
FIT, Czech Technical University in Prague, Thákurova 9, 160 00 Praha 6, Czechia
pavel.surynek@fit.cvut.cz

Abstract

We unify search-based and compilation-based approaches to multi-agent path finding (MAPF) through *satisfiability modulo theories* (SMT). The task in MAPF is to navigate agents in an undirected graph to given goal vertices so that they do not collide. We rephrase Conflict-Based Search (CBS), one of the state-of-the-art algorithms for optimal MAPF solving, in the terms of SMT. This idea combines SAT-based solving known from MDD-SAT, a SAT-based optimal MAPF solver, at the low-level with conflict elimination of CBS at the high-level. Where the standard CBS branches the search after a conflict, we refine the propositional model with a disjunctive constraint. Our novel algorithm called SMT-CBS hence does not branch at the high-level but incrementally extends the propositional model. We experimentally compare SMT-CBS with CBS, ICBS, and MDD-SAT.

1 Introduction and Background

Multi-agent path finding in graphs (MAPF) [Standley, 2010; Yu and LaValle, 2016] represents an important problem in artificial intelligence with specific applications in planning and robotics. We assume multiple distinguishable agents placed in vertices of an undirected graph such that at most one agent is placed in each vertex. Agents can be moved between vertices across edges. MAPF usually assumes that agents are moved to unoccupied neighbors. The task in MAPF is to reach a given goal configuration of agents from a given starting configuration using valid movements.

We address optimal solving of MAPF with respect to common cumulative objective functions that are minimized - *sum-of-costs* [Sharon *et al.*, 2013; Miltzow *et al.*, 2016] and *makespan* [Yu and LaValle, 2016]. The sum-of-costs corresponds to the total cost of all movements (including wait actions) until the goal configuration in reached - traversal of an edge and wait actions have unit cost. The makespan calculates the total number of time-steps until the goal is reached.

Many practical problems from robotics can be interpreted as MAPF. Examples include discrete multi-robot navigation and coordination, item rearrangement in automated warehouses [Basile *et al.*, 2012], ship collision avoidance [Kim *et al.*, 2014], or formation maintenance and maneuvering of aerial vehicles [Zhou and Schwager, 2015].

This paper contributes by the design and experimental analysis of a novel optimal MAPF algorithm that **unifies** two major approaches to solving MAPF optimally: a **search-based** approach represented by *conflict-based search* (CBS) [Sharon *et al.*, 2015] and a **compilation-based** approach represented by reducing MAPF to propositional satisfiability (SAT) [Biere *et al.*, 2009] in the MDD-SAT algorithm [Surynek *et al.*, 2016]. Our novel algorithm called SMT-CBS rephrases ideas of CBS in the terms of *satisfiability modulo theories* (SMT) [Bofill *et al.*, 2012] at the high-level. While at the low-level we use the SAT encoding from MDD-SAT.

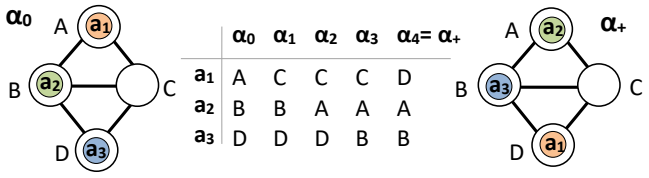
Unlike the original CBS that resolves conflicts between agents by branching the search, SMT-CBS refines the propositional model with a disjunctive constraint to resolve the conflict. SMT-CBS hence does not branch at the high-level but instead incrementally extends the propositional model that is consulted with the external SAT solver similarly as it has been done in MDD-SAT. In contrast to MDD-SAT where the propositional model is fully constructed in a single-shot, the propositional model is being built lazily in SMT-CBS as new conflicts appear.

The hypothesis behind the design of SMT-CBS is that in many cases we do not need to add all constraints to form the *complete propositional model* while still be able to obtain a conflict-free solution. Intuitively we expect that such cases where the *incomplete propositional model* will suffice are represented by sparsely occupied instances with large environments. The expected benefit in contrast to MDD-SAT is that incomplete model can be constructed and solved faster. On the other hand we expect that the superior performance of MDD-SAT in environments densely populated with agents will be preserved as SMT-CBS will quickly converge the model towards the complete one.

We first introduce MAPF formally. Then CBS and MDD-SAT are recalled. On top of this, the combination of CBS and MDD-SAT is developed - the SMT-CBS algorithm. Finally an experimental evaluation of SMT-CBS against CBS , ICBS, and MDD-SAT on various benchmarks is presented.

1.1 MAPF Formally

Multi-agent path finding (MAPF) [Silver, 2005; Ryan, 2008] consists of an undirected graph $G = (V, E)$ and a set of



	α_0	α_1	α_2	α_3	$\alpha_4 = \alpha_+$
a_1	A	C	C	C	D
a_2	B	B	A	A	A
a_3	D	D	D	B	B

Figure 1: A MAPF instance with three agents a_1, a_2, and a_3.

agents $A = \{a_1, a_2, ..., a_k\}$ such that $|A| < |V|$. At most one agent resides in each vertex. The configuration of agents is denoted $\alpha : A \rightarrow V$. Starting configuration of agents α_0 and goal configuration α_+ are specified too.

At each time step, an agent can either *move* to an adjacent vertex or *wait* in its current vertex. The task is to find a sequence of move/wait actions for each agent a_i that moves the agent from $\alpha_0(a_i)$ to $\alpha_+(a_i)$ such that agents do not *collide*, i.e., do not occupy the same vertex at the same time. Typically, an agent can move into adjacent unoccupied vertex provided no other agent enters the same target vertex. However different rules for movements are used in variants of MAPF. An example of MAPF instance is shown in Figure 1.

We develop all new concepts in this paper for the *move-to-unoccupied* variant formalized in the following definition.

Definition 1 (move-to-unoccupied MAPF). *Configuration α' is a valid successor of α if and only if the following conditions hold:*

(i) $\alpha(a) = \alpha'(a)$ or $\{\alpha(a), \alpha'(a)\} \in E$ for all $a \in A$ (agents wait or move along edges);

(ii) for all $a \in A$ it holds that if $\alpha(a) \neq \alpha'(a)$ then $\alpha'(a) \neq \alpha(a')$ for all $a' \in A$ (target vertex is empty);

(iii) and for all $a, a' \in A$ it holds that if $a \neq a'$ then $\alpha'(a) \neq \alpha'(a')$ (no two agents enter the same target).

Solving MAPF is to find a sequence of configurations $[\alpha_0, \alpha_1, ..., \alpha_\mu]$ such that α_{i+1} results using valid movements from α_i for $i = 1, 2, ..., \mu - 1$, and $\alpha_\mu = \alpha_+$. A *feasible solution* of a solvable MAPF instance can be found in polynomial time [Wilson, 1974; Kornhauser *et al.*, 1984]; precisely the worst case time complexity of most practical algorithms for finding feasible solutions is $\mathcal{O}(|V|^3)$ [Luna and Bekris, 2011; de Wilde *et al.*, 2014].

1.2 Cumulative Objectives in MAPF

We are often interested in optimal solutions. In case of the *makespan* [Surynek, 2017] we just need to minimize μ in the aforementioned solution sequence. For introducing the *sum-of-costs* objective [Dresner and Stone, 2008; Standley, 2010; Sharon *et al.*, 2013] we need more notation as follows:

Definition 2 Sum-of-costs objective *is the summation, over all k agents, of the number of time steps required to reach the goal vertex. Denoted ξ, where $\xi = \sum_{i=1}^{k} \xi(path(a_i))$ and $\xi(path(a_i))$ is an individual path cost of agent a_i connecting $\alpha_0(a_i)$ and $\alpha_+(a_i)$ calculated as the number of edge traversals and wait actions disregarding any final wait actions.* [1]

[1]The notation $path(a_i)$ refers to path in the form of a sequence of vertices and edges connecting $\alpha_0(a_i)$ and $\alpha_+(a_i)$ while ξ assigns the cost to a given path.

We note that finding a solution that is optimal (minimal) with respect to either the makespan or the sum-of-costs objective is NP-hard [Ratner and Warmuth, 1986; Surynek, 2010].

2 Unifying Search and Compilation

A necessary step before introducing the unification between the search-based and the compilation-based approach is to briefly discuss both approaches themselves.

2.1 Conflict-based Search

CBS is a representative of **search-based approach**. CBS uses the idea of resolving conflicts lazily; that is, a solution of MAPF instance is not searched against the complete set of movement constraints that forbids collisions between agents but with respect to initially empty set of collision forbidding constraints that gradually grows as new conflicts appear. The advantage of CBS is that it can find a valid solution before all constraints are added.

The high-level of CBS searches a *constraint tree* (CT) using a priority queue in breadth first manner. CT is a binary tree where each node N contains a set of collision avoidance constraints $N.constraints$ - a set of triples (a_i, v, t) forbidding occurrence of agent a_i in vertex v at time step t, a solution $N.paths$ - a set of k paths for individual agents, and the total cost $N.\xi$ of the current solution.

The low-level process in CBS associated with node N searches paths for individual agents with respect to set of constraints $N.constraints$. For a given agent a_i, this is a standard single source shortest path search from $\alpha_0(a_i)$ to $\alpha_+(a_i)$ that avoids a set of vertices $\{v \in V | (a_i, v, t) \in N.constraints\}$ whenever working at time step t. For details see [Sharon *et al.*, 2015].

CBS stores nodes of CT into priority queue OPEN sorted according to the ascending costs of solutions. At each step CBS takes node N with the lowest cost from OPEN and checks if $N.paths$ represent paths that are valid with respect to MAPF movements rules - that is, $N.paths$ are checked for collisions. If there is no collision, the algorithms returns valid MAPF solution $N.paths$. Otherwise the search branches by creating a new pair of nodes in CT - successors of N. Assume that a collision occurred between agents a_i and a_j in vertex v at time step t. This collision can be avoided if either agent a_i or agent a_j does not reside in v at timestep t. These two options correspond to new successor nodes of N - N_1 and N_2 that inherit the set of conflicts from N as follows: $N_1.conflicts = N.conflicts \cup \{(a_i, v, t)\}$ and $N_2.conflicts = N.conflicts \cup \{(a_j, v, t)\}$. $N_1.paths$ and $N_1.paths$ inherit paths from $N.paths$ except those for agents a_i and a_j respectively. Paths for a_i and a_j are recalculated with respect to extended sets of conflicts $N_1.conflicts$ and $N_2.conflicts$ respectively and new costs for both agents $N_1.\xi$ and $N_2.\xi$ are determined. After this, N_1 and N_2 are inserted into the priority queue OPEN.

The pseudo-code of CBS is listed as Algorithm 1. One of crucial steps occurs at line 16 where a new path for colliding agents a_i and a_j is constructed with respect to the extended set of conflicts. $N.paths(a)$ refers to path of agent a.

The CBS algorithm ensures finding sum-of-costs optimal solution (see detailed proofs in [Sharon *et al.*, 2015]).

Algorithm 1: CBS algorithm for MAPF solving

```
1  CBS (G = (V, E), A, α_0, α_+)
2      R.constraints ← ∅
3      R.paths ← {shortest path from α_0(a_i) to
         α_+(a_i) | i = 1, 2, ..., k}
4      R.ξ ← Σ_{i=1}^{k} ξ(N.paths(a_i))
5      insert R into OPEN
6      while OPEN ≠ ∅ do
7          N ← min(OPEN)
8          remove-Min(OPEN)
9          collisions ← validate(N.paths)
10         if collisions = ∅ then
11             return N.paths
12         let (a_i, a_j, v, t) ∈ collisions
13         for each a ∈ {a_i, a_j} do
14             N'.constraints ← N.constraints ∪ {(a, v, t)}
15             N'.paths ← N.paths
16             update(a, N'.paths, N'.constraints)
17             N'.ξ ← Σ_{i=1}^{k} ξ(N'.paths(a_i))
18             insert N' into OPEN
```

Algorithm 2: Framework of SAT-based MAPF solver

```
1  MAPF-SAT (G = (V, E), A, α_0, α_+)
2      paths ← {shortest path from α_0(a_i) to
         α_+(a_i) | i = 1, 2, ..., k}
3      ξ ← Σ_{i=1}^{k} ξ(N.paths(a_i))
4      while TRUE do
5          F(ξ) ← encode(ξ, G, A, α_0, α_+)
6          assignment ← consult-SAT-Solver(F(ξ))
7          if assignment ≠ UNSAT then
8              paths ← extract-Solution(assignment)
9              return paths
10         ξ ← ξ + 1
```

Implications for SMT-CBS

A deterministic implementation of the non-deterministic selection of collision at line 12 has a great impact on the performance. A considerable research effort has been devoted to select collisions with respect to their importance, to this end *cardinal* and *semi-cardinal* collisions have been defined [Boyarski *et al.*, 2015; Felner *et al.*, 2018].

There are two important observations about the CBS algorithm that we further hypothetise to be important in SMT-CBS. Assume fixed time-step t:

Observation 1 *All potential conflicts considering all agents $a_1, a_2, ..., a_k$, and all vertices $v_1, v_2, ..., v_n$ that can appear within CBS together form an* **at-most-one** *occupation constraint: $|\{a_i \mid \alpha_t(a_i) = v_j, i = 1, 2, ...k\}| \leq 1$ for each $j = 1, 2, ..., n$.*

In other words, the above at-most-one constraint, let us denote it as $OCC_{\leq 1}$, says there is at most one agent per vertex.

Observation 2 *It may happen that a set of paths found by CBS at node N such that $N.contraints \subset OCC_{\leq 1}$ is consistent with respect to $OCC_{\leq 1}$.*

The observation formalizes what we generally hope for when using CBS. It can discover a solution before all potential constraints are added which leads to faster solving.

2.2 Compilation to Propositional Satisfiability

The major alternative to CBS is represented by **compilation** of MAPF to propositional satisfiability (SAT) [Surynek *et al.*, 2016; Surynek, 2017]. The idea follows SAT-based planning [Kautz and Selman, 1999] where the existence of a plan for a fixed number time steps is modeled as SAT. We similarly construct propositional formula $\mathcal{F}(\xi)$ such that it is satisfiable if and only if a solution of a given MAPF of sum-of-costs ξ exists. Moreover, the approach is constructive; that is, $\mathcal{F}(\xi)$ exactly reflects the MAPF instance and if satisfiable, solution of MAPF can be reconstructed from satisfying assignment of the formula. We say $\mathcal{F}(\xi)$ to be a *complete propositional model* of MAPF.

Definition 3 (complete propositional model). *Propositional formula $\mathcal{F}(\xi)$ is a* complete propositional model *of MAPF Σ if the following condition holds:*

$\mathcal{F}(\xi)$ is satisfiable $\Leftrightarrow$ Σ has a solution of sum-of-costs ξ.

Being able to construct such formula $\mathcal{F}$ one can obtain optimal MAPF solution by checking satisfiability of $\mathcal{F}(\xi_0)$, $\mathcal{F}(\xi_0 + 1)$, $\mathcal{F}(\xi_0 + 2)$,... until the first satisfiable $\mathcal{F}(\xi)$ is met (ξ_0 is the lower bound for the sum-of-costs calculated as the sum of lengths of shortest paths). This is possible due to monotonicity of MAPF solvability with respect to increasing values of common cumulative objectives. The framework of SAT-based solving is shown in pseudo-code in Algorithm 2.

The advantage of the SAT-based approach is that state-of-the-art SAT solvers can be used for determining satisfiability of $\mathcal{F}(\xi)$ [Audemard and Simon, 2009].

Details of MDD-SAT Encoding

Construction of $\mathcal{F}(\xi)$ as used in MDD-SAT relies on the time expansion of underlying graph G. Having ξ, the basic variant of time expansion determines the maximum number of time steps μ (*makespan*) such that every possible solution of with the sum-of-costs less than or equal to ξ fits in μ timesteps.

The time expansion makes copies of vertices V for each timestep $t = 0, 1, 2, ..., \mu$. That is, we have vertices v^t for each $v \in V$ and time step t. Edges from G are converted to directed edges interconnecting timesteps in the time expansion. Directed edges (u^t, v^{t+1}) are introduced for $t = 1, 2, ..., \mu - 1$ whenever there is $\{u, v\} \in E$. Wait actions are modeled by introducing edges (u^t, u^{t+1}). A directed path in the time expansion corresponds to trajectory of an agent in time. Hence the modeling task now consists in construction of a formula in which satisfying assignments correspond to directed paths from $\alpha_0^0(a_i)$ to $\alpha_+^{\mu}(a_i)$ in the time expansion.

Assume that we have time expansion $TEG_i = (V_i, E_i)$ for agent a_i. Propositional variable $\mathcal{X}_v^t(a_j)$ is introduced for every vertex v^t in V_i. The semantics of $\mathcal{X}_v^t(a_i)$ is that it is $TRUE$ if and only if agent a_i resides in v at time step t. Similarly we introduce $\mathcal{E}_{u,v}^t(a_i)$ for every directed edge (u^t, v^{t+1}) in E_i. Analogously the meaning of $\mathcal{E}_{u,v}^t(a_i)$ is: it is $TRUE$ if

and only if agent a_i traverses edge $\{u, v\}$ between time steps t and $t + 1$.

Finally constraints are added so that truth assignment are restricted to those that correspond to valid solutions of a given MAPF. Added constraints together ensure that $\mathcal{F}(\xi)$ is a *complete propositional model* for given MAPF.

We here illustrate the model by showing few representative constraints. We omit here constraints that concern objective function. For the detailed list of constraints we refer the reader to [Surynek *et al.*, 2016].

We already know that collisions among agents are eliminated by the $OCC_{\leq 1}$ constraint. $OCC_{\leq 1}$ can be expressed on top of $\mathcal{X}_v^t(a_i)$ variables by the following constraint for every $v \in V$ and timestep t:

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

There are various ways how to translate the constraint using propositional clauses. One efficient way is to introduce $\neg\mathcal{X}_v^t(a_i) \vee \neg\mathcal{X}_v^t(a_j)$ for all possible pairs of a_i and a_j.

Next, there is a constraint stating that if agent a_i appears in vertex u at time step t then it has to leave through exactly one edge (u^t, v^{t+1}):

$$\mathcal{X}_u^t(a_i) \Rightarrow \bigvee_{(u^t, v^{t+1}) \in E_i} \mathcal{E}_{u,v}^t(a_i), \tag{2}$$

$$\sum_{v^{t+1} \mid (u^t, v^{t+1}) \in E_i} \mathcal{E}_{u,v}^t(a_i) \leq 1 \tag{3}$$

Similarly, the target vertex of any movement except wait action must be empty. This is ensured by the following constraint for every $(u^t, v^{t+1}) \in E_i$:

$$\mathcal{E}_{u,v}^t(a_i) \Rightarrow \bigwedge_{a_j \in A \mid a_j \neq a_i \wedge v^t \in V_j} \neg\mathcal{X}_v^t(a_j) \tag{4}$$

Other constraints ensure that truth assignments to variables per individual agents form paths. That is if agent a_i enters an edge it must leave the edge at the next time step.

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

A common measure how to reduce the number of decision variables derived from the time expansion is the use of *multi-value decision diagrams* (MDDs) [Sharon *et al.*, 2013]. The basic observation that holds for MAPF is that an agent can reach vertices in the distance d (distance of a vertex is measured as the length of the shortest path) from the current position of the agent no earlier than in the d-th time step. Analogous observation can be made with respect to the distance from the goal position.

Above observations can be utilized when making the time expansion of G. For a given agent, we do not need to consider all vertices at time step t but only those that are reachable in t timesteps from the initial position and that ensure that the goal can be reached in the remaining $\mu - t$ timesteps.

The combination of SAT-based approach and MDD time expansion led to the MDD-SAT algorithm described in [Surynek *et al.*, 2016] that currently represents state-of-the-art in SAT-based MAPF solving.

3 Combining SMT and CBS

A natural relaxation from the complete propositional model is an *incomplete propositional model* where instead of the equivalence between solving MAPF and the formula we require an implication only.

Definition 4 (incomplete propositional model). *Propositional formula $\mathcal{H}(\xi)$ is an* incomplete propositional model *of MAPF Σ if the following condition holds:*

$\mathcal{H}(\xi)$ is satisfiable $\Leftarrow$ Σ has a solution of sum-of-costs ξ.

A close look at CBS reveals that it operates similarly as problem solving in *satisfiability modulo theories* (SMT) [Bofill *et al.*, 2012]. SMT divides satisfiability problem in some complex theory T into an abstract propositional part that keeps the Boolean structure of the decision problem and a simplified decision procedure $DECIDE_T$ that decides fragment of T restricted on *conjunctive formulae*. A general T-formula Γ is transformed to a *propositional skeleton* by replacing atoms with propositional variables. The SAT solver then decides what variables should be assigned $TRUE$ in order to satisfy the skeleton - these variables tells what atoms hold in Γ. $DECIDE_T$ then checks if the conjunction of atoms assigned $TRUE$ is valid with respect to axioms of T. If so then satisfying assignment is returned. Otherwise a conflict from $DECIDE_T$ (often called a lemma) is reported back and the skeleton is extended with a constraint forbidding the conflict. This is the basic SMT solving process; more advanced schemes exist where the SAT solver and T are integrated more tightly [Nieuwenhuis *et al.*, 2006].

The above observation led us to the idea to rephrase CBS in terms of SMT. The abstract propositional part working with the skeleton will be taken from MDD-SAT provided that only constraints ensuring that assignments form valid paths interconnecting starting positions with goals will be preserved. Other constraints for collision avoidance will be omitted initially. This will result in an *incomplete propositional model*.

The paths validation procedure will act as $DECIDE_T$ and will report back the set of conflicts found in the current solution. Hence axioms of T will be represented by the movement rules of MAPF. We call the resulting algorithm SMT-CBS and it is shown in pseudo-code as Algorithm 3. The algorithm simulates the aforementioned SMT solving scheme.

The algorithm is divided into two procedures: SMT-CBS representing the main loop and SMT-CBS-Fixed solving the input MAPF for fixed cost ξ. The major difference from the standard CBS is that there is no branching at the high-level. The high-level SMT-CBS roughly correspond to the main loop of MDD-SAT. The set of conflicts is iteratively collected during the entire execution of the algorithm. Procedure *encode* from MDD-SAT is replaced with *encode-Basic* that produces encoding that ignores specific movement rules (collisions between agents) but in contrast to *encode* it encodes collected conflicts into $\mathcal{H}(\xi)$.

The conflict resolution in the standard CBS implemented as high-level branching is here represented by refinement of $\mathcal{H}(\xi)$ with disjunction (line 20). The presented SMT-CBS can eventually build the same formula as MDD-SAT but this is done lazily in SMT-CBS.

Algorithm 3: SMT-CBS algorithm for MAPF solving

```
1  SMT-CBS (Σ = (G = (V, E), A, α0, α+))
2      conflicts ← ∅
3      paths ← {path*(ai) a shortest path from α0(ai) to
         α+(ai) | i = 1, 2, ..., k}
4      ξ ← Σ_{i=1}^{k} ξ(paths(ai))
5      while TRUE do
6          (paths, conflicts) ←
             SMT-CBS-Fixed(conflicts, ξ, Σ)
7          if paths ≠ UNSAT then
8              return paths
9          ξ ← ξ + 1
10 SMT-CBS-Fixed(conflicts, ξ, Σ)
11     H(ξ) ← encode-Basic(conflicts, ξ, Σ)
12     while TRUE do
13         assignment ← consult-SAT-Solver(H(ξ))
14         if assignment ≠ UNSAT then
15             paths ← extract-Solution(assignment)
16             collisions ← validate(paths)
17             if collisions = ∅ then
18                 return (paths, conflicts)
19             for each (ai, aj, v, t) ∈ collisions do
20                 H(ξ) ← H(ξ) ∪ {¬X_v^t(ai) ∨ ¬X_v^t(aj)}
21                 conflicts ←
                     conflicts ∪ {[(ai, v, t), (aj, v, t)]}
22         return (UNSAT, conflicts)
```

In the line with Observations 1 and 2 such approach may save resources as a solution may be found or its existence can be ruled out prior to adding all constraints.

4 Experiments

We performed experiments with the new SMT-CBS algorithm on standard benchmarks [Boyarski *et al.*, 2015; Sharon *et al.*, 2013]. Representative part of results is presented.

4.1 Benchmarks and Setup

We implemented SMT-CBS in C++ on top of the Glucose 4 SAT solver [Audemard and Simon, 2009] that ranks among the best SAT solvers according to recent SAT solver competitions [Balyo *et al.*, 2017]. Whenever possible the SAT solver is consulted in the incremental mode. The standard CBS has been re-implemented in C++ from scratch since the original implementation written in Java does support only grids but not general graphs [Sharon *et al.*, 2015] that we need in our tests. We also compared SMT-CBS to one of the most recent C++ versions of ICBS [Boyarski *et al.*, 2015] that implements *rectangle reasoning* [Li *et al.*, 2019]. And finally we took existing implementation of MDD-SAT also written in C++.

In CBS we implemented preference of resolving *cardinal conflicts* [Boyarski *et al.*, 2015]. Without this heuristic, CBS exhibited poor performance. In SMT-CBS we initially tried to resolve against single cardinal conflict too but eventually it turned out to be more efficient to resolve against all discovered conflicts (the presented pseudo-code shows this variant).

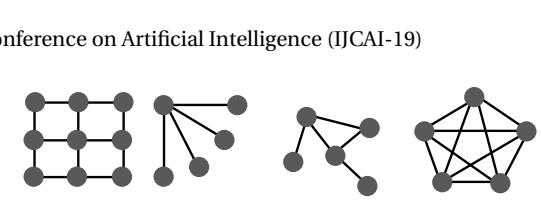


Figure 2: Examples of *grid*, *star*, *random graph*, and *clique*.

All experiments were run on a Ryzen 7 CPU 3.0 Ghz under Kubuntu linux 16 with 16 GB RAM.[2].

We divided the experimental evaluation into two categories of tests. The first part of experimental evaluation has been done on diverse instances consisting of **small** graphs: 4-connected open *grid* of size 8×8, *random graphs* containing 20% of random edges, *star* graphs, and *cliques* (see Figure 2). Cliques, random graphs, and stars consisted of 16 vertices.

The initial and goal configurations of agents have been generated randomly in all tests.

The second part of experimental evaluation took place on large 4-connected maps taken from *Dragon Age* [Sharon *et al.*, 2015; Sturtevant, 2012]. In contrast to small instances, these were only sparsely populated with agents. Initial and goal configuration were generated at random again.

We varied the number of agents in MAPF instances to obtain instances of various difficulties. For each number of agents we generated 10 random instances.

The timeout was set to 1000 seconds. Presented results were obtained from instances finished under this timeout.

4.2 Comparison on Small Graphs

Tests on small graphs were focused on the runtime comparison. Results are shown Figure 3 - sorted runtimes are presented, hence easy instances are to the left while hard instances are to the right.

CBS performs well in easy instances but its performance degrades quickly. Both MDD-SAT and SMT-CBS are faster for instances containing more agents. For hardest instances solvable under the timeout (the hardest 8×8 instance contains 20 agents) the performance of MDD-SAT and SMT-CBS is roughly the same. The most interesting situation can be observed in the middle with instances of medium difficulty; here SMT-CBS dominates over MDD-SAT by factor of 2 to 10.

These observations are in the line with our hypotheses. When agents interacts too much, SMT-CBS produces the same formula as MDD-SAT does, that is why we see similar performance for hardest instances. With less interacting agents SMT-CBS does not need to deal with all potential conflicts and is faster than MDD-SAT.

4.3 Evaluation on Large Maps

The second category of tests was focused on the performance of CBS, MDD-SAT and SMT-CBS on large maps. In the three structurally different maps up to 50 agents were placed randomly. Again we had 10 random instances per each number of agents.

[2]To enable reproducibility of presented results we provide the complete source code and experimental data: http://users.fit.cvut.cz/∼surynpav/research/ijcai2019

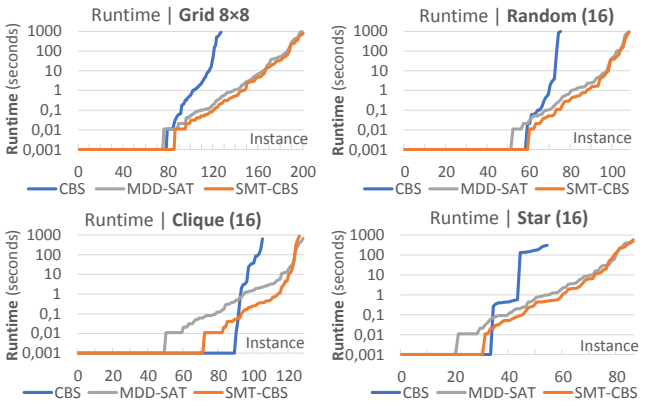


Figure 3: Runtime comparison of CBS, MDD-SAT and SMT-CBS on various small graphs.

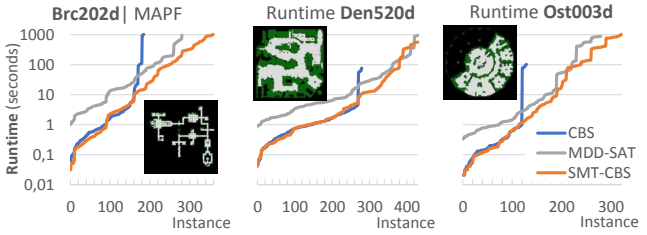


Figure 4: Runtime comparison of CBS, MDD-SAT and SMT-CBS on large maps.

Sorted runtimes are shown in Figure 4. There is no significant difference between CBS and SMT-CBS in easier cases but MDD-SAT lags behind. The situation changes after going into the medium difficulty region where runtimes of CBS go quickly up while SMT-CBS maintains significant advantage over MDD-SAT (factor 2 to 5). The performance of SMT-CBS and MDD-SAT eventually meets in the hard region.

In addition to runtime comparison, we compared the number of clauses generated by MDD-SAT and SMT-CBS. This comparison is focused on verifying if SMT-CBS terminates prior to adding all possible constraints. Sorted numbers of clauses are shown in Figure 5 from which we can clearly see that SMT-CBS generates order of magnitudes fewer clauses than MDD-SAT. This is directly reflected in smaller memory consumption by SMT-CBS.

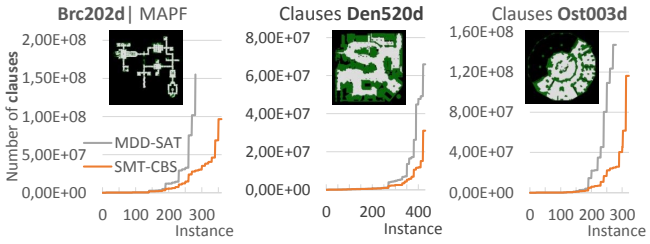


Figure 5: Clauses generated by MDD-SAT and SMT-CBS.

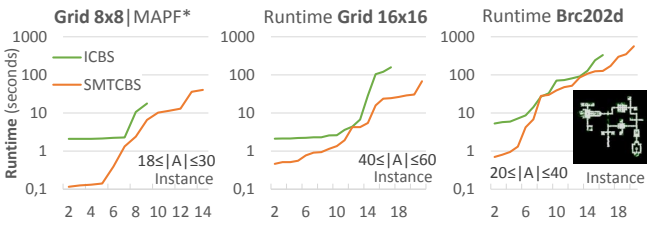


Figure 6: Comparison of SMT-CBS to ICBS implementing rectangular reasoning.

4.4 SMT-CBS and Recent Progress in ICBS

Comparing our implementaiton of SMT-CBS with the recent version of ICBS featuring various heuristics such as rectangular reasoning indicated that SMT-CBS has a significant advantage on small grids[3] (sorted runtimes presented in Figure 6). As instances get harder, runtime of ICBS goes quickly up reaching the timeout of 1000 seconds earlier than SMT-CBS. In large maps, SMT-CBS and ICBS are closer to each other however SMT-CBS was still faster for a larger set of instances.

In this experiment we used a version of MAPF (denoted MAPF*) where agents can enter vertices being simultaneously vacated by other agents. This induced necessity to introduce edge conflicts in SMT-CBS.

The important observation is that SMT-CBS and ICBS differ on individual instances greatly. It often happens that an instance easy for ICBS is hard for SMT-CBS and vice versa.

5 Conclusion

We suggested a new MAPF solving method called SMT-CBS that combines advantages of MDD-SAT - fast solving in highly constrained cases, and advantages of CBS - fast solving in large sparse environments.

The new algorithm outperforms CBS and ICBS and improves the performance of MDD-SAT in terms of the size of generated formulae as well as in runtime. Experiments confirmed our hypotheses that SMT-CBS is able to produce solution well before all constraints are added to the encoding which altogether leads to faster solving.

The performance gap between MDD-SAT and SMT-CBS is not as big as the gap in the number of generated clauses. The explanation of this is that SMT-CBS needs to call the SAT solver more times than MDD-SAT does which represents an extra overhead.

For the future work we plan to further generalize SMT-CBS approach; geometric agents in a continuous environment represent one promising direction.

Acknowledgments

This work has been supported by GAČR - the Czech Science Foundation, grant registration number 19-17966S.

[3]The implementation of ICBS does not support general graphs.

References

[Audemard and Simon, 2009] G. Audemard and L. Simon. Predicting learnt clauses quality in modern SAT solvers. In *IJCAI*, pages 399–404, 2009.

[Balyo *et al.*, 2017] Tomás Balyo, Marijn J. H. Heule, and Matti Järvisalo. SAT competition 2016: Recent developments. In *AAAI*, pages 5061–5063, 2017.

[Basile *et al.*, 2012] Francesco Basile, Pasquale Chiacchio, and Jolanda Coppola. A hybrid model of complex automated warehouse systems - part I: modeling and simulation. *IEEE Trans. Automation Science and Engineering*, 9(4):640–653, 2012.

[Biere *et al.*, 2009] A. Biere, A. Biere, M. Heule, H. van Maaren, and T. Walsh. *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[Bofill *et al.*, 2012] Miquel Bofill, Miquel Palahí, Josep Suy, and Mateu Villaret. Solving constraint satisfaction problems with SAT modulo theories. *Constraints*, 17(3):273–303, 2012.

[Boyarski *et al.*, 2015] E. Boyarski, A. Felner, R. Stern, G. Sharon, D. Tolpin, O. Betzalel, and S. Shimony. ICBS: improved conflict-based search algorithm for multi-agent pathfinding. In *IJCAI*, pages 740–746, 2015.

[de Wilde *et al.*, 2014] B. de Wilde, A. ter Mors, and C. Witteveen. Push and rotate: a complete multi-agent pathfinding algorithm. *JAIR*, 51:443–492, 2014.

[Dresner and Stone, 2008] K. Dresner and P. Stone. A multiagent approach to autonomous intersection management. *JAIR*, 31:591–656, 2008.

[Felner *et al.*, 2018] Ariel Felner, Jiaoyang Li, Eli Boyarski, Hang Ma, Liron Cohen, T. K. Satish Kumar, and Sven Koenig. Adding heuristics to conflict-based search for multi-agent path finding. In *Proceedings of ICAPS 2018*, pages 83–87, 2018.

[Kautz and Selman, 1999] Henry A. Kautz and Bart Selman. Unifying sat-based and graph-based planning. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 1999*, pages 318–325, 1999.

[Kim *et al.*, 2014] D.-G Kim, Katsutoshi Hirayama, and G.-K Park. Collision avoidance in multiple-ship situations by distributed local search. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 18:839–848, 09 2014.

[Kornhauser *et al.*, 1984] Daniel Kornhauser, Gary L. Miller, and Paul G. Spirakis. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *FOCS, 1984*, pages 241–250, 1984.

[Li *et al.*, 2019] Jiaoyang Li, Daniel Harabor, Peter Stuckey, Hang Ma, and Sven Koenig. Symmetry-breaking constraints for grid-based multi-agent path finding. In *AAAI*. AAAI Press, 2019.

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

[Miltzow *et al.*, 2016] Tillmann Miltzow, Lothar Narins, Yoshio Okamoto, Günter Rote, Antonis Thomas, and Takeaki Uno. Approximation and hardness of token swapping. In *ESA 2016*, volume 57 of *LIPIcs*, pages 66:1–66:15. Schloss Dagstuhl, 2016.

[Nieuwenhuis *et al.*, 2006] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT modulo theories: From an abstract davis–putnam–logemann–loveland procedure to dpll(T). *J. ACM*, 53(6):937–977, 2006.

[Ratner and Warmuth, 1986] Daniel Ratner and Manfred K. Warmuth. Finding a shortest solution for the N x N extension of the 15-puzzle is intractable. In *AAAI*, pages 168–172, 1986.

[Ryan, 2008] Malcolm R. K. Ryan. Exploiting subgraph structure in multi-robot path planning. *J. Artif. Intell. Res. (JAIR)*, 31:497–542, 2008.

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

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

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

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

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

[Surynek *et al.*, 2016] 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.

[Surynek, 2010] Pavel Surynek. An optimization variant of multi-robot path planning is intractable. In *AAAI 2010*. AAAI Press, 2010.

[Surynek, 2017] Pavel Surynek. Time-expanded graph-based propositional encodings for makespan-optimal solving of cooperative path finding problems. *Ann. Math. Artif. Intell.*, 81(3-4):329–375, 2017.

[Wilson, 1974] Richard M Wilson. Graph puzzles, homotopy, and the alternating group. *Journal of Combinatorial Theory, Series B*, 16(1):86 – 96, 1974.

[Yu and LaValle, 2016] Jingjin Yu and Steven M. LaValle. Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Trans. Robotics*, 32(5):1163–1177, 2016.

[Zhou and Schwager, 2015] Dingjiang Zhou and Mac Schwager. Virtual rigid bodies for coordinated agile maneuvering of teams of micro aerial vehicles. In *ICRA 2015*, pages 1737–1742, 2015.