Multi-Goal Multi-Agent Path Finding via Decoupled and Integrated Goal Vertex Ordering

Pavel Surynek *

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

Abstract

We introduce multi-goal multi agent path finding (MG-MAPF) which generalizes the standard discrete multi-agent path finding (MAPF) problem. While the task in MAPF is to navigate agents in an undirected graph from their starting vertices to one individual goal vertex per agent, MG-MAPF assigns each agent multiple goal vertices and the task is to visit each of them at least once. Solving MG-MAPF not only requires finding collision free paths for individual agents but also determining the order of visiting agent's goal vertices so that common objectives like the sum-of-costs are optimized. We suggest two novel algorithms using different paradigms to address MG-MAPF: a heuristic search-based algorithm called Hamiltonian-CBS (HCBS) and a compilation-based algorithm built using the satisfiability modulo theories (SMT), called SMT-Hamiltonian-CBS (SMT-HCBS).

Introduction

Mutli-agent path finding (MAPF) (Silver 2005; Ryan 2008; Surynek 2009; Luna and Bekris 2011; Wang and Botea 2011; Ma and Koenig 2017) is an abstraction for many real-life problems where agents, both autonomous or passive, need to be moved (see (Felner et al. 2017; Ma and Koenig 2017) for a survey). The environment in MAPF is modeled as an undirected graph where vertices represent positions and edges define the topology [1].

The standard variant of MAPF assumes that each agent starts in a specified starting vertex and its task is to reach a specified goal vertex. While such formalization encompass many real-life navigation tasks (Cáp et al. 2013; Ma et al. 2017a) there still exist problems especially in logistic domain where the standard MAPF lacks expressiveness.

Such problems that cannot be expressed as MAPF include situations where agents have multiple goals so that instead of reaching single goal location agents need to perform a round-trip to service a set of goals. Many real-life applications requires that agents perform certain task at each of multiple goal locations such as performing maintenance or pickup operation (Pansart, Catusse, and Cambazard 2018; Briant et al. 2020).

Having multiple goals per agent adds a significant new challenge to the problem consisting of determining the order of visiting agent's goal vertices. Hence the ordering of goals as well as non-conflicting path finding are subject to decision making which in addition to this aims on the optimization of various objectives such as commonly adopted sum-of-costs (Standley 2010; Sharon et al. 2013)

We introduce a problem we call a *multi-goal multi-agent path finding* (MG-MAPF). The problem shares movement rules with the standard MAPF but generalizes it by allowing each agent to have multiple goal vertices. Each of the agent's goal vertices must be visited at least once to successfully solve MG-MAPF.

Contribution

We introduce two novel solving algorithms for MG-MAPF: a search-based Hamiltonian CBS (HCBS), a derivative of the CBS algorithm (Sharon et al. 2015), and a compilation-based algorithm, called SMT-Hamiltonian-CBS, derived from SMT-CBS (Surynek 2019b).

The important feature of HCBS is that it decouples the goal vertex ordering from collision free path finding, making HCBS a three level search algorithm where at the high-level, the standard CBS conflict resolution search is performed and the low level search for collision-free circuit is further divided into two levels. The higher level of circuit-finding determines the goal vertex ordering while the lower level finds a collision free circuit visiting all goals following the order determined by the higher level.

The compilation-based approach uses the ideas from solvers for *satisfiability modulo theories* (SMT) (Barrett, de Moura, and Stump 2005; Barrett et al. 2009) namely lazy construction of formulae. A significant feature of this approach is that collision avoidance and goal vertex ordering are integrated and solved at once.

The paper is organized as follows: basic definitions from MAPF and its properties are introduced first. Then search-based HCBS and related MG-MAPF specific heuristics are developed. Compilation-based SMT-HCBS follows in the next section. Finally both new algorithms are experimentally evaluated and compared.

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

[1] Similar setup is used in *graph pebbling* (Kornhauser, Miller, and Spirakis 1984; Parberry 2015; Ratner and Warmuth 1990) where however the focus is rather on computational complexity.

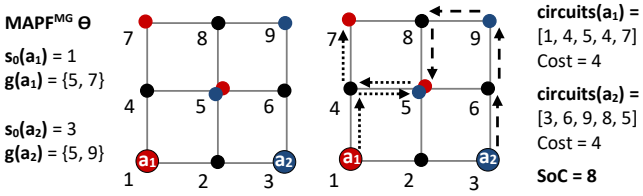

Figure 1: A MG-MAPF instance with and its sum-of-costs optimal solution.

Background and Definitions

We first recall the concepts from the standard multi-agent path finding (MAPF) that we inherit into MG-MAPF. Agents in MAPF are placed in vertices of an undirected graph so that there is at most one agent per vertex. Formally, $s : A \mapsto V$ is a *configuration* of agents in vertices of the graph. A configuration can be transformed instantaneously to the next one by *valid* movements of agents; the next configuration corresponds to the next *time step*. An agent can move into another vertex across an edge provided no collision occurs. A *collision* occurs if two agents are located in the same vertex (*vertex collision*) or if two agents simultaneously cross the same edge in opposite directions (*edge collision*) (Sharon et al. 2015). The configuration at time step t is denoted s_t.

Definition 1 Mutli-goal multi-agent path finding *(MG-MAPF) is a 4-tuple* $\Theta = (G = (V, E), A, s_0, g)$ *where* $G = (V, E)$ *is an undirected graph,* $A = \{a_1, a_2, ..., a_k\}$ *with* $k \in \mathbb{N}$ *is a set of agents where* $k \leq |V|$*,* $s : A \mapsto V$ *represents agents' starting vertices (starting configuration), and* $g : A \mapsto 2^V$ *assigns a set of goal vertices to each agent.*

Each agent in MG-MAPF has the task to visit its goal vertices. Agent's goal vertices can be visited in an arbitrary order but each goal vertex must be visited by the agent at least once. Various objectives can be taken into account. We develop all concepts here for the *sum-of-costs* objective commonly adopted in MAPF but different cumulative objectives can be used as well (Surynek et al. 2016b).

Formally, an MG-MAPF solution is a sequence of configurations that can be obtained via valid moves from the starting configuration s_0 following the MAPF movement rules such that each agent visits each of its goal vertices at least once:

Definition 2 *A solution to MG-MAPF* $\Theta = (G = (V, E), A, s_0, g)$ *is a sequence of configurations* $S = [s_0, s_1, ..., s_{t_M}]$ *such that* s_t *results from* s_{t-1} $\forall t \in \{1, 2, ..., t_M\}$ *via valid MAPF movements and* $\forall a_i \in A$ *it holds that* $(\forall v_g \in g(a_i))\ (\exists t \in \{1, 2, ..., t_M\})\ (s_t(a_i) = v_g)$.

Given a solution $S = [s_0, s_1, ..., s_{t_M}]$, t_M is the *makespan*, denoted $Make(S)$. We define the *sum-of-costs* as the sum of costs for individual agents: $SoC(S) = \sum_{i=1}^{k} Cost(a_i)$, where the individual cost for agent a_i is defined as: $Cost(a_i) = \min \{t_c \mid (\forall v_g \in g(a_i))\ (\exists t \in 1, 2, ..., t_c)\ (s_t(a_i) = v_g)\}$.

Let us note that the $SoC(S)$ accumulates unit costs of actions including wait actions until all goals are visited. After that we do not care about agent's movements. This is one possible definition of MG-MAPF. We can alternatively require that all agents should return to their starting vertices and/or stay in the final goal. The developed solving algorithms require only minor modifications to be applicable for any of such MG-MAPF variants.

MG-MAPF as well as MAPF is NP-hard (Ratner and Warmuth 1990; Surynek 2010; Yu and LaValle 2013, 2015). A difference can be observed for a special case with one agent. Finding a sum-of-costs optimal solution to MAPF with one agent is in P as it corresponds to finding a shortest path in G. On the other hand sum-of-costs optimal MG-MAPF with one agent is still NP-hard as it corresponds to finding a Hamiltonian path (Rahman and Kaykobad 2005). An example of MG-MAPF is shown in Figure 1.

Hamiltonian Conflict-based Search: HCBS

We suggest a novel algorithm called Hamiltonian Conflict-based Search (Hamiltonian CBS, HCBS) which shares the high level structure with the CBS algorithm (Sharon et al. 2015), one of the most commonly used algorithm for the standard MAPF (Li et al. 2019b,a; Zhang et al. 2020; Li et al. 2020; Boyarski et al. 2020).

When trying to use CBS for MG-MAPF, the significant challenge is represented by the fact that at the low level there is no longer search for a minimum cost path with respect to the set of conflicts, a polynomial-time problem, but rather the search for a minimum cost Hamiltonian path (which even without a set of conflicts is a difficult problem).

Definition 3 *A* Hamiltonian path *(HP) in* G *starting at* $u \in V$ *covering a subset of vertices* $U \subseteq V$ *is a sequence of vertices denoted* $H_P(u, U) = [h_0, h_1, ..., h_{t_H}]$ *such that* $h_1 = u$*,* $\{h_t, h_{t+1}\} \in E$ *for* $t \in \{0, 1, ..., t_H - 1\}$*, and for each* $v \in U$ $\exists t \in \{0, 1, ..., t_H\}$ *such that* $h_t = v$*. The cost of Hamiltonian path corresponds to the number of its edges:* $Cost(H_P(u, U)) = t_H - 1$.

Such CBS hence consists of potentially exponential-sized conflict resolution at the high-level where each conflict leads to exponential-time search for a fresh Hamiltonian path, resulting in an algorithm with high exponential factor. Despite prohibitive theoretical complexity such algorithm could be feasible provided that the low level search is fast enough.

Decoupled Goal Ordering

The key to adapt CBS for MG-MAPF is to decouple the goal vertex ordering from conflict avoidance. The search for a Hamiltonian path going through agent's goal vertices with respect to a set of conflicts is done in two level fashion. At the high-level (of this low level) we are trying to determine optimal ordering of agent's goal vertices. To this purpose we made use of the A* algorithm (Hart, Nilsson, and Raphael 1968) that searches the space of possible **permutations** of agent's goals. After determining the next goal vertex to visit, the algorithm searches for the shortest path observing the conflicts that interconnects the next goal vertex with the current one. The search for the shortest path is done by another instance of A*. Another important factor for the performance of the decoupled approach are the heuristics.

Minimum Spanning Tree Heuristic

We define a variant of *spanning tree* with respect to a subset of vertices of undirected graph $G = (V, E)$.

Definition 4 *A* spanning tree *(ST) of an undirected graph $G = (V, E)$ with respect to a subset of vertices $U \subseteq V$, denoted $T_S(U) = (V_U, E_U)$ is a tree covering U, that is, $U \subseteq V_U \subseteq V$ and $E_U \subseteq E$. The* cost *of a spanning tree is defined as the number of edges included in the tree: $Cost(T_S(U)) = |E_U|$. A* minimum spanning tree *(MST) with respect to U is a spanning tree with minimum cost.*

Observe that $T_S(U)$ may contain other vertices in addition to U to keep it connected. We use the notation $T_S(u, U)$ for $u \in V$ denoting a spanning tree covering $\{u\} \cup U$.

The important property of MST is that it can be found in polynomial time with respect to G (Boruvka 1926; Nesetril, Milková, and Nesetrilová 2001). Another important property is that the cost of MST can serve as the lower bound for the cost of shortest Hamiltonian path:

Proposition 1 *For any $u \in V$ and $U \subseteq V$ it holds that $min\{Cost(T_S(u, U))\} \leq min\{Cost(H_P(u, U))\}$.*

This enables us to use the concept of MST as a basis for a *consistent* A* heuristic (proof omitted). The HCBS algorithm is described using pseudo-code as Algorithm 1.

The **high-level** represented by HCBS$_{conflicts}$ follows the standard CBS. It construct a *constraint tree* (CT) in breadth-first search manner 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 at $v \in V$ at time step t, a solution $N.circuits$ - a set of k Hamiltonian paths for individual agents covering their individual goals, and the sum-of-costs $N.SoC$ of the current solution. Nodes are stored in a priority queue OPEN and processed in the ascending order according to $N.SoC$.

Initially, the shortest Hamiltonian paths for each agent are determined and corresponding node is stored into OPEN (lines 2-5). At a general step, HCBS takes a node N with the minimum sum-of-costs from OPEN and checks whether it represents a valid solution w.r.t. MG-MAPF rules (lines 7-9). If there are no collisions, then the valid solution $N.circuits$ is returned and the algorithm finishes (lines 10-11). Otherwise the search branches by creating two new nodes N_1 and N_2 - successors of N. Assuming a collision (a_i, a_j, v, t) between agents a_i and a_j at vertex v at time step t, this can be avoided if either a_i or a_j does not reside at v at time step t. This requirement corresponds to two conflict avoidance constraints in successor nodes N_1 and N_2 that inherit the set of constraints from N as follows: $N_1.constraints = N.constraints \cup \{(a_i, v, t)\}$ and $N_2.constraints = N.constraints \cup \{(a_j, v, t)\}$ (line 14). In addition to this, fresh Hamiltonian paths for affected agents a_i and a_j are recomputed in respective nodes (lines 15-17).

The **low-level** is represented by HCBS$_{ordering}$ that determines the ordering of agent's goal vertices. This is an A*-based search of the space of goal vertex permutations. Each node N of the search tree consists of $N.u$ a current vertex, starting at $s_0(a_i)$, a set of visited goals $N.finished$, an A*'s g-value and h-value: $N.g$ and $N.h$ where $N.g$ corresponds to the actual cost of partially constructed Hamiltonian path finishing in $N.u$, and $N.h$ is a lower bound estimation of the cost for the remaining part of the Hamiltonian path calculated as the cost of MST starting in $N.u$ and covering the remaining goals. Finally, $N.circuit$ is the partial Hamiltonian path itself.

The very low-level search is done by another instance of A* (line 33) which interconnects the current vertex $N.u$ with the candidate for the next goal v by a shortest path taking into account collision avoidance $constraints$ from the very high-level CBS-style search.

Altogether HCBS consists of **thee levels of search** in three different spaces: (i) space of conflicts, (ii) goal ordering space, and (iii) path space.

Combining soundness and optimality of CBS with properties of MST-based heuristic for goal vertex ordering we can state the following proposition.

Proposition 2 *The HCBS algorithm returns sum-of-costs optimal solution for given input MG-MAPF Θ.*

Compilation-based Approach: SMT-HCBS

The second approach for solving MG-MAPF is based on the reduction of MG-MAPF to a series of Boolean formulae that are decided by an external SAT solver (Audemard and Simon 2018). Our new algorithm called SMT-Hamiltonian-CBS combines existing SMT-CBS (Surynek 2019b) with MG-MAPF specific generation of target Boolean formuale.

Time Expansion and Decision Diagrams

Construction of a Boolean formula corresponding to solvability of MAPF as used in SMT-CBS relies on the time expansion of G. Having SoC, the basic variant of time expansion determines the maximum number of time steps (*makespan*) t_M such that every possible solution with the sum-of-costs less than or equal to SoC fits in t_M timesteps.

The time expansion makes copies of vertices V for each timestep $t = 0, 1, 2, ..., t_M$. 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 $t = 0, 1, ..., t_M - 1$ 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 the trajectory of an agent in G. Hence the modeling task now consists in construction of a formula in which satisfying truth-value assignments correspond to directed paths from $s_0^0(a_i)$ to $g^{t_M}(a_i)$ in the time expansion.

The time expansion is often further improved so that unreachable nodes are removed which reduces the subsequent search effort done on top of the time expansion (Bryant 1995; Sharon et al. 2013; Surynek et al. 2016a). A structure called *multi-valued decision diagram* (MDD) is a subset of the time expansion as described above, that is, it is a directed graph $MDD_i = (V_i, E_i)$ for each agent a_i. A vertex v^t is included in MDD_i if is reachable with respect to the given makespan bound t_M. That is, v^t is included if and only if $Dist(s_0(a_i), v) \leq t$ (agent a_i has enough time to reach v at time step t) and $Dist(v, g(a_i)) \leq t_M - t$ (agent a_i can

Algorithm 1: HCBS algorithm for MG-MAPF.

HCBS$_{conflicts}$ $(\Theta = (G = (V, E), A, s_0, g))$
  $N.constraints \leftarrow \emptyset$
  $N.circuits \leftarrow \{circuit^*(a_i)$ a shortest Hamiltonian path from $s_0(a_i)$ covering $g(a_i) \mid i = 1, 2, ..., k\}$
  $N.SoC \leftarrow \sum_{i=1}^{k} Cost(N.circuits(a_i))$
  insert (SoC, N) into OPEN
  while OPEN $\neq \emptyset$ **do**
    $(key, N) \leftarrow$ min-Key(OPEN)
    remove-Min-Key(OPEN)
    $collisions \leftarrow$ validate$(N.circuits)$
    if $collisions = \emptyset$ **then**
      return $N.circuits$
    let $(a_i, a_j, v, t) \in collisions$
    for *each* $a \in \{a_i, a_j\}$ **do**
      $N'.constraints \leftarrow N.constraints \cup \{(a, v, t)\}$
      $N'.circuits \leftarrow N.circuits$
      $N'.circuits(a_i) \leftarrow$ HCBS$_{ordering}$ $(\Theta, a, N'.constraints)$
      $SoC' \leftarrow \sum_{i=1}^{k} Cost(N'.circuits(a_i))$
      insert (SoC', N') into OPEN

HCBS$_{ordering}$ $(\Theta, a_i, constraints)$
  let $\Theta = (G = (V, E), A, s_0, g)$
  $N.u \leftarrow s_0(a_i)$; $N.finished \leftarrow \emptyset$
  $T_S(N.u, g(a_i)) \leftarrow$ construct-MST$(s_0(a_i), g_R, G)$
  $N.g \leftarrow 0$; $N.h \leftarrow Cost(T_S(N.u, g_R))$
  $N.circuit \leftarrow []$
  insert $(N.g + N.h, N)$ into OPEN
  while OPEN $\neq \emptyset$ **do**
    $(key, N) \leftarrow$ min-Key(OPEN)
    remove-Min-Key(OPEN)
    if $N.finished = g(a_i)$ **then**
      return $N.circuit$
    else
      for *each* $v \in g(a_i) \setminus N.finished$ **do**
        $path \leftarrow$ A*$(N.u, v, constraints, N.g)$
        if $path \neq Fail$ **then**
          $N'.u \leftarrow v$
          $N'.finished \leftarrow N.finished \cup \{v\}$
          $g'_R \leftarrow g(a_i) \setminus N'.finished$
          $T_S(N'.u, g'_R) \leftarrow$ construct-MST$(N'.u, g'_R, G)$
          $N'.g \leftarrow N.g + Cost(path)$
          $N'.h \leftarrow Cost(T_S(N'.u, g'_R))$
          $N'h \leftarrow N.circuit \cdot path$
          insert $(N'.g + N'.h, N)$ into OPEN
  return *Fail*

reach its goal in the remaining time) where $Dist(u, v)$ is the length of the shortest path between u and v in G.

Hamiltonian Multi-value Decision Diagram

The idea of MDD can be adapted for MG-MAPF. We introduce a *Hamiltonian MDD* for individual agents denoted MDD_i^H which is almost identical to MDD_i in the terms of expanding G. However, the reachability of vertices at given time steps in MDD_i^H is treated in a different way to reflect that each agent in MG-MAPF has multiple goal vertices.

A node v^t is included in MDD_i^H iff the following conditions hold:

1. $Dist(s_0(a_i), v) \leq t$ and
2. a Hamiltonian path $H_P = H_P(s_0(a_i), g(a_i) \cup \{v\})$ exists in G such that $Cost(H_P) \leq t_M$.

In other words, time t must be enough to reach v from the start and there must be a Hamiltonian path starting at $s_0(a_i)$ visiting both v and all agent's goals of the cost not exceeding the number of levels of expansion t_M. Without proof let us summarize important property of MDD^{H_P}.

Proposition 3 *A sequence of vertices visited by an agent a_i within any solution to MG-MAPF Θ with the sum-of-costs SoC corresponds to a directed path in $MDD_i^{H_P}$.*

Computing costs of Hamiltonian paths for every vertex in G requires substantial computational effort as finding Hamiltonian path is an NP-hard problem (Garey, Johnson, and Tarjan 1976) and also confirmed by our preliminary tests. Hence we introduce a relaxation of MDD^{H_P} denoted MDD^{T_S} in which the requirement of the existence of a Hamiltonian path is replaced with the requirement of existence of a **spanning tree** $T_S(s_0(a_i), g(a_i) \cup \{v\})$ of cost at most t_M. Computing a spanning tree can be done in polynomial time. The direct corollary of Proposition 1 is that using MDD^{T_S} is sound for construction of the formula.

Corollary 1 *A sequence of vertices visited by an agent a_i within any solution to MG-MAPF Θ with the sum-of-costs SoC corresponds to a directed path in $MDD_i^{T_S}$.*

See figure 2 for illustration how MDD^{T_S} can be used to reduce the number of nodes in the time expansion. A spanning tree of cost 3 including vertices v_1, v_3 and v_5 exists so these vertices reachable at time step 3 from v_1 are included but v_2 not because no spanning tree of cost less than 4 including v_1, v_2, v_3 and v_5 exists in G (similarly for v_6).

As solutions to MG-MAPF are typically longer than those in the standard MAPF such pruning of the time expansion is crucial to keep the resulting formula reasonably small.

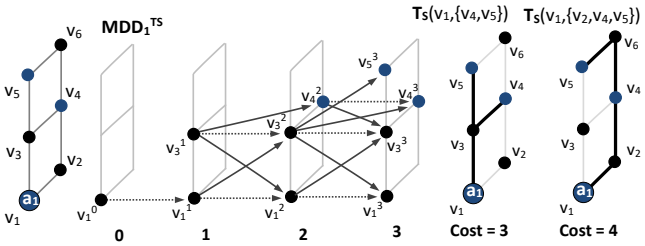

Figure 2: MDD^{T_S} expanded for $t_M = 3$. Spanning trees including various vertices are shown too.

Incomplete Boolean Model

We construct a formula $\mathcal{H}(SoC)$ representing an *incomplete model* for MG-MAPF. Assume we have $MDD_i^{T_S} = (V_i, E_i)$

for a_i from which we derive the formula as follows. A Boolean 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 with the meaning that it is $TRUE$ if and only if agent a_i traverses edge $\{u, v\}$ between time steps t and $t+1$.

Constraints are added on top of these variables to encode the movement rules of MG-MAPF that are exactly the standard MAPF rules such as that agents move across edges (do not jump) and preserve basic physical properties (do not disappear and do not spawn). The detailed list of constraints is given in (Surynek et al. 2016a) and (Surynek 2019b), for the sake of brevity we show here only illustrative examples:

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

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

These constraints state that if a_i is in $u \in V$ at time step t then it has to leave via exactly one edge (u^t, v^{t+1}).

MG-MAPF differs from MAPF in the encoding of the goal condition. An agent a_i in MG-MAPF must visit each of its goals at least once. Hence we add for each $v \in g(a_i)$ the following constraint, that is we take all copies of v in $MDD_i^{T_S}$ and require that at least one of the corresponding Boolean variables must be $TRUE$:

$$\sum_{v^t \mid v^t \in V_i} \mathcal{X}_v^t(a_i) \geq 1 \tag{3}$$

The incompleteness of the model as suggested in (Surynek 2019b) consists in omitting certain constraints. Specifically collision avoidance constraints are omitted. Hence instead of the equivalence between the satisfiability of $\mathcal{H}(SoC)$ and the solvability of MG-MAPF under given SoC we only establish an **implication** as follows:

Definition 5 (incomplete Boolean model). *A Boolean formula $\mathcal{H}(SoC)$ is an* incomplete Boolean model *of MG-MAPF Θ if the following condition holds:*

$\mathcal{H}(SoC) \in SAT \Leftarrow \Theta$ *has a solution of sum-of-costs SoC.*

The Algorithm and Integrated Goal Ordering

After obtaining a satisfiable formula we cannot immediately declare it to represent a valid solution but first the extracted candidate for MG-MAPF solution must be verified for collisions. If there are no collisions then we are finished and the candidate is a valid MG-MAPF solution. If not then a collision avoidance constraint must be added to refine $\mathcal{H}(SoC)$ and solving process continues with the next iteration. Such a tight integration of lazy formula construction and its solving using the SAT solver is often used in the *satisfiability modulo theories* (SMT) paradigm (Katz et al. 2016).

Assume that a collision occurs at time step t at vertex v between agents a_i and a_j (denoted (a_i, a_j, v, t)). Eliminating this collision is to forbid that a_i or a_j appears at v at time step t which naturally corresponds to a binary clause: $\neg\mathcal{X}_v^t(a_i) \vee \neg\mathcal{X}_v^t(a_j)$. Being aware of the construction of modern CDCL SAT solvers (Audemard and Simon 2018) we can see that this is quite fortunate as such short clauses promote *Boolean constraint propagation* (resolution, unit propagation) that significantly reduces the search effort.

Algorithm 2: SMT-based MG-MAPF solver

```
1  SMT-HCBS (Θ = (G = (V, E), A, s_0, g))
2      conflicts ← ∅
3      circuits ← {circuit*(a_i) a shortest Hamiltonian path
         from s_0(a_i) covering g(a_i) | i = 1, 2, ..., k}
4      SoC ← Σ_{i=1}^{k} Cost(circuits(a_i))
5      while TRUE do
6          (circuits, conflicts) ←
             SMT-HCBS-Fixed(conflicts, SoC, Θ)
7          if circuits ≠ UNSAT then
8              return circuits
9          SoC ← SoC + 1
10 SMT-HCBS-Fixed(conflicts, Soc, Θ)
11     H(SoC) ← encode-Hamiltonian(conflicts, SoC, Θ)
12     while TRUE do
13         assignment ← consult-SAT-Solver(H(SoC))
14         if assignment ≠ UNSAT then
15             circuits ← extract-Solution(assignment)
16             collisions ← validate(circuits)
17             if collisions = ∅ then
18                 return (circuits, conflicts)
19             for each (a_i, a_j, v, t) ∈ collisions do
20                 H(ξ) ← H(ξ) ∪ {¬X_v^t(a_i) ∨ ¬X_v^t(a_j)}
21                 conflicts ←
                     conflicts ∪ {[(a_i, v, t), (a_j, v, t)]}
22         else
23             return (UNSAT,conflicts)
```

The resulting algorithm called SMT-HCBS is described using pseudo-code as Algorithm 2. It is a modification of existing SMT-CBS, the major difference is the use of specific Boolean encoding for MG-MAPF (line 11) as described above. The algorithm tests the existence of a solution of the input instance Θ for increasing sum-of-costs until a positive answer is obtained (lines 5-9). The lower bound for sum-of-costs is obtained as the sum-of-costs of individual Hamiltonian paths for agents (lines 3-4).

The advantage against *complete models* (Surynek et al. 2016a) is that a valid solution can be obtained before the problem is fully specified in terms of constraints which often leads to faster solving process. Observe that an unsatisfiable formula in the case of incomplete model according to Definition 5 means that the input instance Θ is not solvable under the given cost SoC, leading to its incrementing of SoC and the next iteration of the algorithm.

Let us note that both goal vertex ordering and conflict resolution are integrated at the same conceptual level - both problems are encoded in the target Boolean formula are decided by the SAT solver.

Experimental Evaluation

We evaluated HCBS and SMT-HCBS on standard benchmarks from `movingai.com` (Sturtevant 2012). Representative part of the results is presented in this section.

Benchmarks and Setup

HCBS and SMT-HCBS were implemented in C++. The SMT-HCBS solver is built on top of the Glucose 3.0 SAT solver (Audemard and Simon 2018) that ranks among the high performing SAT solvers according to recent SAT solver competitions (Balyo, Heule, and Järvisalo 2017).

The experimental evaluation has been done on diverse instances consisting of 4-connected *grid* maps ranging in sizes from small to large. Random MAPF scenarios from `movingai.com` are used to generate MG-MAPF instances. To obtain instances of various difficulties we varied the number of agents while the number of goal vertices per agent was set as constant. As defined in the benchmark set, 25 different instances are generated per number of agents.

Starting positions of agents are taken directly from the scenario. Since only one goal is defined per agent in each MAPF scenario the set of goals for an agent is generated by making a specified number of random picks among goal positions of all agents in the scenario. This results in instances whose goal vertices are equally distributed across the map.

The tests we focused on comparing HCBS and SMT-HCBS in the terms of success rate and runtime. All experiments were run on system consisting of 200 Xeon 2.8 GHz cores, 1TB RAM, running Ubuntu Linux 18. [2] The success rate is the ratio of the number of instances out of 25 per number of agents that the solver managed to solver under the time limit of 5 minutes.

Runtime Results

We divided the tests into three categories with respect to the size of maps. Results for **small** instances derived from the `empty-16-16` map are shown in Figure 3. Three different cases with the number of goal vertices per agent: 1 (corresponds to MAPF), 2, 4, and 8 are tested.

We can observe that for 1, 2 and 4 goals per agent SMT-HCBS dominates. This is an expectable result since similar behavior can be observed for SMT-CBS vs. CBS for the standard MAPF (Surynek 2019b). However the clearly visible trend it that HCBS scales better for increasing number of goals which eventually resulted in reversed situation with 8 goals where HCBS performs better.

Results for **medium-sized** instances shown in Figure 4, where 8 goals per agent were used, indicate that HCBS perform significantly better than SMT-HCBS. Only relatively closer performance to HCBS was achieved by SMT-HCBS on instances derived from the `room-64-64-8` map.

Finally, results for **large** instances shown in Figure 5 indicate clear dominance of HCBS over SMT-HCBS.

[2]For the full reproducibility of the presented results we provide a complete source code of our solvers and detailed experimental data on author's web: `http://users.fit.cvut.cz/~surynpav/research` and in git repository: `https://github.com/surynek`.

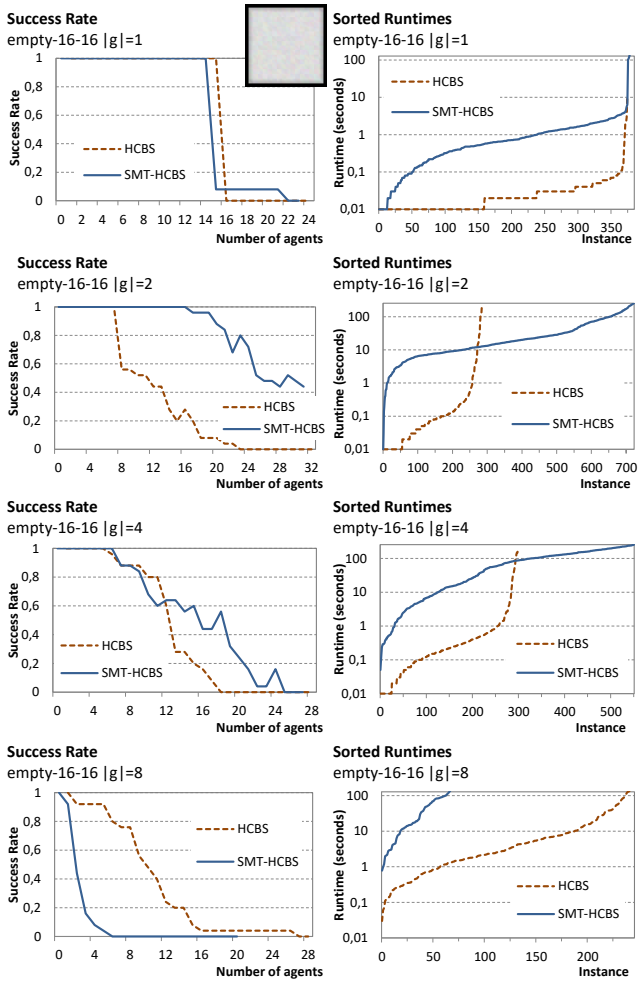

Figure 3: Success rate and runtime comparison on small-sized maps.

Explanation of Results

There are multiple factors explaining the observed results. SMT-HCBS often needs to iterate through many unsatisfiable sum-of-costs before the optimal sum-of-cost is met on instances with greater number of goal vertices. This is due to the fact that agents interact more in such cases which results in a greater difference between the true optimal sum-of-costs and its lower bound estimation. Moreover node pruning in the spanning tree MDDs is less efficient for greater number of goals since the cost of spanning tree deviates more from the true Hamiltonian path cost.

In contrast to this, the effect of domain specific heuristics is more direct in HCBS which can more effectively determine the optimal ordering of goal vertices for individual agents - the choice of the next vertex can be immediately assessed by the heuristic. Similar effect cannot be easily achieved in SMT-HCBS since during MDD generation phase we need to represent all possible goal vertex orderings and the SAT solver itself has no domain specific information.

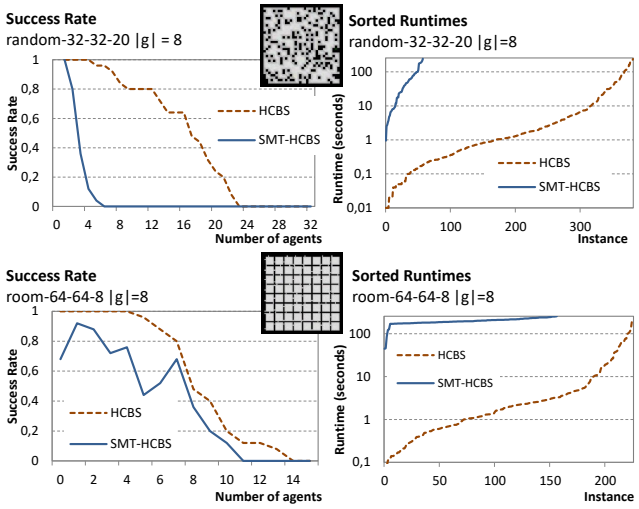

Figure 4: Success rate and runtime comparison on medium-sized maps.

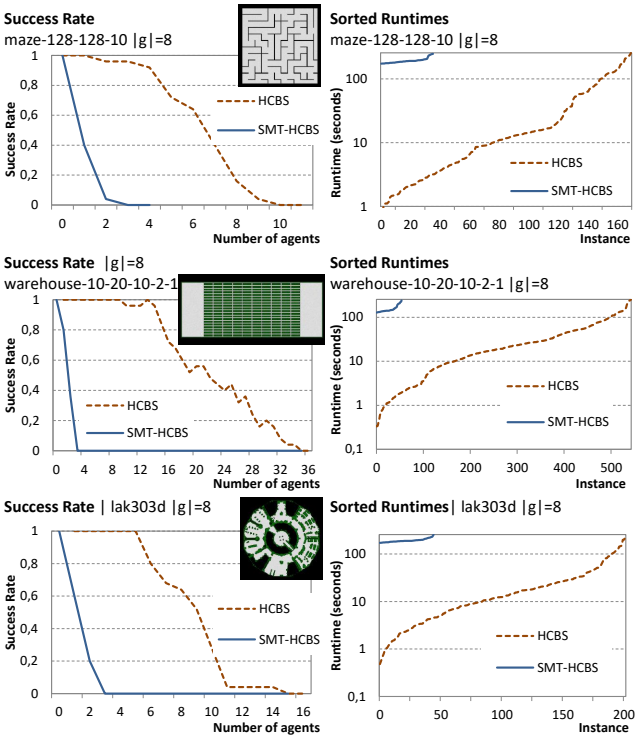

Figure 5: Success rate and runtime comparison on large-sized maps.

Related Work

The most closely related problem to MG-MAPF is *multi-agent pickup and delivery* (MAPD) (Ma et al. 2017b; Liu et al. 2019), defining a set of tasks $T = \{t_1, t_2, ...t_m\}$ where each task $t_j = (p_j, d_j)$ is characterized by a pickup location $p_j \in V$ and a delivery location $d_j \in V$. Agents can freely select tasks to fulfill. The freedom in choice of tasks and the order of their fulfilling makes the problem very hard. The contemporary solving approaches for MAPD first assign tasks to agents and followed by determining the ordering of tasks per agent ignoring collisions. Then collision free paths are planned according to the task ordering. As there is no feedback between the phases the resulting plan is sub-optimal.

Another related work is represented by order picking problem (OPP) (Pansart, Catusse, and Cambazard 2018) which is a variant of traveling salesman problem in a rectangular warehouse. Various integer programming and flow-based formulations of OPP have been studied. The important difference from MG-MAPF is that it is typically regarded as a single agent problem, hence collisions between agents are not considered in OPP.

A generalization of MAPF where an agent is assigned multiple goal vertices and is successful if it reaches any of its goals is suggested in (Surynek 2013). Similar setup has been studied for *target assignment* in MAPF (Hönig et al. 2018) where each agent selects one goal from multiple choices.

A compilation-based approach to MAPF that uses a similar lazy clause generation scheme as SMT-HCBS but derived from CSP has been proposed in (Gange, Harabor, and Stuckey 2019).

Conclusion

We introduced a novel multi-goal multi agent path finding (MG-MAPF) problem. MG-MAPF generalizes MAPF by assigning each agent a set of goal vertices instead of one goal. The task is to visit each of agent's goal vertices at least once while we aim on finding sum-of-costs optimal solutions. Two algorithms generating sum-of-costs optimal solutions are suggested. A search-based algorithm called Hamiltonian CBS (HCBS) derived from CBS and a compilation-based approach SMT-HCBS that reduces MG-MAPF to Boolean logic and solves the problem in the target formalism by the SAT solver.

HCBS introduces three level search in which conflict resolution is done at the high level and goal vertex ordering and path planning are done at the low level. The key technique is decoupling the vertex ordering from collision-free path planning. CBS provides greater room for integrating heuristics that we made use of when adapting it for MG-MAPF.

In contrast to this, SMT paradigm provides more powerful search but integration of domain specific heuristics is more complicated. We increased the informedness of the SMT-based algorithm SMT-HCBS through the concept of Hamiltonian and spanning tree MDD which eventually led to significant improvements so that SMT-HCBS is able to solve some of the standard benchmarks. However decoupling of the goal vertex ordering from conflict resolution turned out to be the crucial factor behind the better performance of HCBS. The worse support of domain specific heuristics shows limitations of the SMT approach generally.

For future work we plan investigate possibility of generating the formulae lazily not only in the phase of conflict resolution but also in the phase of MDD generation. A possible future direction is also to address MG-MAPF via the DPLL(MG-MAPF) framework (Surynek 2019a).

References

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

Balyo, T.; Heule, M. J. H.; and Järvisalo, M. 2017. SAT Competition 2016: Recent Developments. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, 5061–5063. AAAI Press.

Barrett, C. W.; de Moura, L. M.; and Stump, A. 2005. SMT-COMP: Satisfiability Modulo Theories Competition. In *Computer Aided Verification, 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6-10, 2005, Proceedings*, volume 3576 of *Lecture Notes in Computer Science*, 20–23. Springer.

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.

Boruvka, O. 1926. O jistém problému minimáním. *Práce Moravské přírodovědecké společnosti* 3(3): 7–58.

Boyarski, E.; Harabor, D.; Stuckey, P. J.; Bodic, P. L.; and Felner, A. 2020. F-Cardinal Conflicts in Conflict-Based Search. In Harabor, D.; and Vallati, M., eds., *Proceedings of the Thirteenth International Symposium on Combinatorial Search, SOCS 2020, Online Conference [Vienna, Austria], 26-28 May 2020*, 123–124. AAAI Press.

Briant, O.; Cambazard, H.; Cattaruzza, D.; Catusse, N.; Ladier, A.; and Ogier, M. 2020. An efficient and general approach for the joint order batching and picker routing problem. *Eur. J. Oper. Res.* 285(2): 497–512.

Bryant, R. E. 1995. Binary decision diagrams and beyond: enabling technologies for formal verification. In Rudell, R. L., ed., *Proceedings of the 1995 IEEE/ACM International Conference on Computer-Aided Design, ICCAD 1995*, 236–243. IEEE Computer Society / ACM.

Cáp, M.; Novák, P.; Vokrínek, J.; and Pechoucek, M. 2013. Multi-agent RRT: sampling-based cooperative pathfinding. In Gini, M. L.; Shehory, O.; Ito, T.; and Jonker, C. M., eds., *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2013*, 1263–1264. IFAAMAS.

Felner, A.; Stern, R.; Shimony, S. E.; Boyarski, E.; Goldenberg, M.; Sharon, G.; Sturtevant, N. R.; Wagner, G.; and Surynek, P. 2017. Search-Based Optimal Solvers for the Multi-Agent Pathfinding Problem: Summary and Challenges. In *Proceedings of the Tenth International Symposium on Combinatorial Search, SOCS 2017*, 29–37. AAAI Press.

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 2019*, 155–162. AAAI Press.

Garey, M. R.; Johnson, D. S.; and Tarjan, R. E. 1976. The Planar Hamiltonian Circuit Problem is NP-Complete. *SIAM J. Comput.* 5(4): 704–714.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* 4(2): 100–107.

Hönig, W.; Kiesel, S.; Tinka, A.; Durham, J. W.; and Ayanian, N. 2018. Conflict-Based Search with Optimal Task Assignment. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018*, 757–765. IFAAMAS / ACM.

Katz, G.; Barrett, C. W.; Tinelli, C.; Reynolds, A.; and Hadarean, L. 2016. Lazy proofs for DPLL(T)-based SMT solvers. In Piskac, R.; and Talupur, M., eds., *2016 Formal Methods in Computer-Aided Design, FMCAD 2016, Mountain View, CA, USA, October 3-6, 2016*, 93–100. IEEE.

Kornhauser, D.; Miller, G. L.; and Spirakis, P. G. 1984. Coordinating Pebble Motion on Graphs, the Diameter of Permutation Groups, and Applications. In *25th Annual Symposium on Foundations of Computer Science, FOCS 1984*, 241–250. IEEE Computer Society.

Li, J.; Boyarski, E.; Felner, A.; Ma, H.; and Koenig, S. 2019a. Improved Heuristics for Multi-Agent Path Finding with Conflict-Based Search: Preliminary Results. In *Proceedings of the Twelfth International Symposium on Combinatorial Search, SOCS 2019, Napa, California, 16-17 July 2019*, 182–183. AAAI Press.

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.

Li, J.; Harabor, D.; Stuckey, P. J.; Ma, H.; and Koenig, S. 2019b. Disjoint Splitting for Multi-Agent Path Finding with Conflict-Based Search. In *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS 2018, Berkeley, CA, USA, July 11-15, 2019*, 279–283. AAAI Press.

Liu, M.; Ma, H.; Li, J.; and Koenig, S. 2019. Task and Path Planning for Multi-Agent Pickup and Delivery. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*, 1152–1160. International Foundation for Autonomous Agents and Multiagent Systems.

Luna, R.; and Bekris, K. E. 2011. Push and Swap: Fast Cooperative Path-Finding with Completeness Guarantees. In Walsh, T., ed., *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, 294–300. IJCAI/AAAI.

Ma, H.; and Koenig, S. 2017. AI buzzwords explained: multi-agent path finding (MAPF). *AI Matters* 3(3): 15–19.

Ma, H.; Koenig, S.; Ayanian, N.; Cohen, L.; Hönig, W.; Kumar, T. K. S.; Uras, T.; Xu, H.; Tovey, C. A.; and Sharon, G. 2017a. Overview: Generalizations of Multi-Agent Path Finding to Real-World Scenarios. *CoRR* abs/1702.05515.

Ma, H.; Li, J.; Kumar, T. K. S.; and Koenig, S. 2017b. Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, 837–845. ACM.

Nesetril, J.; Milková, E.; and Nesetrilová, H. 2001. Otakar Boruvka on minimum spanning tree problem Translation of both the 1926 papers, comments, history. *Discret. Math.* 233(1-3): 3–36.

Pansart, L.; Catusse, N.; and Cambazard, H. 2018. Exact algorithms for the order picking problem. *Comput. Oper. Res.* 100: 117–127.

Parberry, I. 2015. Solving the (n^2 - 1)-Puzzle with 8/3 n^3 Expected Moves. *Algorithms* 8(3): 459–465.

Rahman, M. S.; and Kaykobad, M. 2005. On Hamiltonian cycles and Hamiltonian paths. *Inf. Process. Lett.* 94(1): 37–41.

Ratner, D.; and Warmuth, M. K. 1990. NxN Puzzle and Related Relocation Problem. *J. Symb. Comput.* 10(2): 111–138.

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.

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

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.

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

Surynek, P. 2009. A novel approach to path planning for multiple robots in bi-connected graphs. In *2009 IEEE International Conference on Robotics and Automation, ICRA 2009, Kobe, Japan, May 12-17, 2009*, 3613–3619. IEEE.

Surynek, P. 2010. An Optimization Variant of Multi-Robot Path Planning Is Intractable. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. AAAI Press.

Surynek, P. 2013. Optimal Cooperative Path-Finding with Generalized Goals in Difficult Cases. In *Proceedings of the Tenth Symposium on Abstraction, Reformulation, and Approximation, SARA 2013, 11-12 July 2013, Leavenworth, Washington, USA*. AAAI.

Surynek, P. 2019a. On the Tour Towards DPLL(MAPF) and Beyond. In *Discussion and Doctoral Consortium papers of AI*IA 2019 - 18th International Conference of the Italian Association for Artificial Intelligence, Rende, Italy, November 19-22, 2019*, volume 2495 of *CEUR Workshop Proceedings*, 74–83. CEUR-WS.org.

Surynek, P. 2019b. 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. 2016a. Efficient SAT Approach to Multi-Agent Path Finding Under the Sum of Costs Objective. In *ECAI 2016 - 22nd European Conference on Artificial Intelligence*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, 810–818. IOS Press.

Surynek, P.; Felner, A.; Stern, R.; and Boyarski, E. 2016b. 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.

Wang, K. C.; and Botea, A. 2011. MAPP: a Scalable Multi-Agent Path Planning Algorithm with Tractability and Completeness Guarantees. *J. Artif. Intell. Res.* 42: 55–90.

Yu, J.; and LaValle, S. M. 2013. Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*. AAAI Press.

Yu, J.; and LaValle, S. M. 2015. Optimal Multi-Robot Path Planning on Graphs: Structure and Computational Complexity. *CoRR* abs/1507.03289.

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.