2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Bounded Sub-optimal Multi-Robot Path Planning Using Satisfiability Modulo Theory (SMT) Approach

Pavel Surynek[1]

Abstract— **Multi-robot path planning (MRPP) is a task of planning collision free paths for a group of robots in a graph. Each robot starts in its individual starting vertex and its task is to reach a given goal vertex. Existing techniques for solving MRPP optimally under various objectives include search-based and compilation-based approaches. Often however finding an optimal solution is too difficult hence sub-optimal algorithms that trade-off the quality of solutions and the runtime have been devised. We suggest eSMT-CBS, a new bounded sub-optimal algorithm built on top of recent compilation-based method for optimal MRPP based on satisfiability modulo theories (SMT). We compare eSMT-CBS with ECBS, a major representative of bounded sub-optimal search-based algorithms. The experimental evaluation shows significant advantage of eSMT-CBS across variety of scenarios.**

I. INTRODUCTION AND MOTIVATION

Multi-robot path planning in graphs (MRPP) [1]–[4] and related pebble motion on graphs (PMG) [5], [6] represent important concepts for understanding motion planning in robotics from the combinatorial perspective. The MRPP problem consists a graph, $G = (V, E)$ and a set $R = \{r_1, r_2, \ldots r_k\}$ of k robots where each robot is placed in a vertex so that at most one robot resides in a vertex.

The task is to find a sequence of move/wait actions for each robot r_i, moving it from its initial position to a given individual goal such that robots do not *collide*, i.e., do not occupy the same location at the same time while moving towards their goals following the plan.

Although MRPP assumes discretization of both time and space it has many applications in continuous physical environments (see [7] for a survey). Examples include multi-robot navigation and coordination [8], robot reconfiguration in automated warehouses [9], ship collision avoidance [10], or formation maintenance and maneuvering of aerial vehicles [11]. The scope of this paper is limited to the setting of **fully cooperative** robots that are centrally controlled.

The discretization of space and time in MRPP is important for simplifying the problem - solving MRPP is often easier than solving the corresponding problem in the continuous space and time [12]. Moreover discrete plans can be executed on physical robots if robots' movements are synchronized with the discrete plan. The synchronization can be achieved for example by reflex capabilities of robots as we show in our simulations with OZOBOT Evo robots [13].

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

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

MRPP is usually solved aiming to minimize one of commonly-used global cumulative cost functions such as: **(1) sum-of-costs** is the summation, over all robots, of the number of time steps required to reach the goal location [3], [14]–[16] or **(2) makespan** - the time until the last robot reaches its goal vertex [17]–[19].

Optimal solvers for MRPP can be divided to two classes. **(1) Search-based solvers.** These algorithms consider MRPP as a graph search problem. Some of these algorithm are variants of the A* algorithm that search [3], [20]. Others algorithms such as ICTS [15] and CBS [7], [21] employ novel (non-A*) search tree.

(2) Compilation-based solvers. By contrast, some optimal solvers compile MRPP to known problems such as constraint satisfaction (CSP) [22], Boolean satisfiability (SAT) [23], Inductive Logic Programming [24] and Answer Set Programming [25]. These solvers employ a polynomial-time reduction from MRPP to the target formalism. In this paper we further widen this direction and introduce **SMT-based bounded sub-optimal** solver built using previous state-of-the-art compilation-based solver SMT-CBS [26].

Many suboptimal solvers were developed for MRPP to trade-off the quality of solutions and the runtime. Some suboptimal solvers aim to to quickly find paths for all robots while paying no attention to the quality of the solution, i.e., how far it is from the optimal solution. We refer to such algorithms as **any solution** MRPP solvers. Many any solution MRPP solvers were proposed [1], [22], [27]–[29], and there are even polynomial time any solution MRPP solvers such as PUSH-AND-SWAP [30], PUSH-AND-ROTATE [31], and BIBOX [32]. Polynomial time algorithms are usually used when k is large, but they guarantee the quality of solutions only roughly.

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

Despite the large number of works devoted to optimal or to suboptimal solutions, there are only few approaches that provide bounded suboptimal solutions: ECBS [33] and CBS with highways [27], both are modifications of the *conflict based search* (CBS) algorithm, and eMDD-SAT [34], a sub-optimal variant of the SAT-based solver MDD-SAT [35].

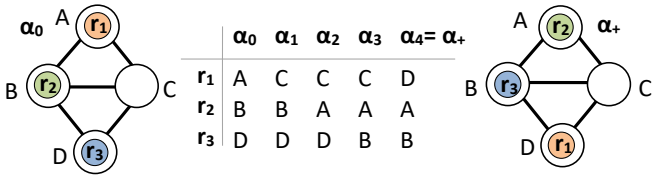

Fig. 1. An MRPP instance with three robots r_1, r_2, and r_3.

A. Contributions

We introduce a new satisfiability modulo theory-based solver eSMT-CBS. The new solver is built on top of the existing optimal SMT-based solver SMT-CBS [26]. The SMT paradigm [36], [37] is similar to SAT-based solvers but integrates the underlying SAT solver [38] in a more active way into the solving process for the target problem.

The organization of the paper is as follows. We first give the background in conflict-based search. Then prerequisites for constructing SMT framework for robot path planning are recalled, the SAT-based MRPP solving and the satisfiability modulo theory-based SMT-CBS algorithm. On top of this, the bounded sub-optimal algorithm eSMT-CBS is developed. Finally, an experimental evaluation comparing eSMT-CBS with major search-based sub-optimal algorithm ECBS [39] is presented showing that eSMT-CBS achieves significantly better or comparable performance across diverse scenarios and hence represents a viable alternative to search-based solvers

II. BACKGROUND

We first recall *multi-robot path planning* (MRPP) [1], [40] formally and related solvers as they appear in the literature.

A. Multi-robot Path Planning

In MRPP, the time is discretized into time steps. The configuration of robots at time-step t is denoted as $\alpha_t : R \mapsto V$. Formally, an MRPP instance is a tuple $\Sigma = (G = (V, E), R, \alpha_0, \alpha_+)$ where $\alpha_0 : R \mapsto V$ is an initial configuration of robots and $\alpha_+ : R \mapsto V$ is a goal configuration of robots. A *solution* for Σ is a sequence of configurations $\mathcal{S}(\Sigma) = [\alpha_0, \alpha_1, ..., \alpha_\mu]$ such that α_{t+1} results from valid movements from α_t for $t = 1, 2, ..., \mu - 1$, and $\alpha_\mu = \alpha_+$. Orthogonally, the solution can be represented as a set of paths for individual robots.

At each time step a robot can either *move* to an adjacent location or *wait* in its current location. Robots must not *conflict*, i.e., must not occupy the same location at the same time. Typically, a robot can move into adjacent unoccupied vertex provided no other robot enters the same target vertex but other rules for movements are used as well [1].. An example of MRPP instance and its solution is shown in Figure 1.

Our bounded sub-optimal solver will use the *sum-of-costs* as the objective function formally introduced as follows:

[1]Different movement rules such as train-like movements where only the leading robot needs to enter a vacant vertex while other robots can immediately follow it are also used.

Definition 1: **Sum-of-costs** denoted ξ is the summation, over all robots, of the number of time steps required to reach the goal. Formally, $\xi = \sum_{i=1}^{k} \xi(path(r_i))$, where $\xi(path(r_i))$ is an *individual path cost* of robot r_i connecting $\alpha_0(r_i)$ calculated as the number of edge traversals and wait actions. [2]

Observe that in the sum-of-costs we accumulate the cost of wait actions for robots not yet reaching their goal vertices. A feasible solution of a solvable MRPP instance can be found in polynomial time [5], [6]; precisely the worst case time complexity of most practical algorithms for finding feasible solutions is $\mathcal{O}(|V|^3)$ (asymptotic size of the solution is also $\mathcal{O}(|V|^3)$) [30], [31], [41], [42]. This is also asymptotically best what can be done as there are MRPP instances requiring $\Omega(|V|^3)$ moves.

Finding an optimal solution with respect to the sum-of-costs objective is NP-hard [17], [43] and also determining the existence of a solution that differs from the optimum by a factor less than $4/3$ in case of the makespan optimization is NP-hard too [44]. Therefore designing algorithms based on search and Boolean satisfiability (SAT) [45] for MRPP is justifiable.

B. Conflict-based Search

Conflict-based search (CBS) [16] is a major search-based algorithm for MRPP. CBS resolves conflicts lazily; that is, a solution of MRPP is not searched against the complete set of movement constraints that forbids collisions between robots 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 (r_i, v, t) forbidding occurrence of robot r_i in vertex v at time step t, a solution $N.paths$ - a set of k paths for individual robots, and the total cost $N.\xi$ of the current solution.

The low level process in CBS associated with node N searches paths for individual robots with respect to set of constraints $N.constraints$. For a given robot r_i, this is a standard single source shortest path search from $\alpha_0(r_i)$ to $\alpha_+(r_i)$ that avoids a set of vertices $\{v \in V | (r_i, v, t) \in N.constraints\}$ whenever working at time step t. For details see [16].

CBS stores nodes of CT into priority queue OPEN sorted according to ascending costs of solutions. At each step CBS takes node N with lowest cost from OPEN and checks if $N.paths$ represent paths that are valid with respect to movements rules in MRPP. That is, $N.paths$ are checked for collisions. If there is no collision, the algorithms returns valid MRPP solution $N.paths$. Otherwise the search branches by creating a new pair of nodes in CT - successors of N.

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

Assume that a collision occurred between robots r_i and r_j in vertex v at time step t. This collision can be avoided if either robot r_i or robot r_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 set of conflicts from N as follows: $N_1.conflicts = N.conflicts \cup \{(r_i, v, t)\}$ and $N_2.conflicts = N.conflicts \cup \{(r_j, v, t)\}$. $N_1.paths$ and $N_1.paths$ inherit path from $N.paths$ except those for robot r_i and r_j respectively. Paths for r_i and r_j are recalculated with respect to extended sets of conflicts $N_1.conflicts$ and $N_2.conflicts$ respectively and new costs for both robots $N_1.\xi$ and $N_2.\xi$ are determined. After this N_1 and N_2 are inserted into the priority queue OPEN.

The CBS algorithm ensures finding sum-of-costs optimal solution. Detailed proofs of this claim can be found in [16]. The high-level of CBS is similar to Dijkstra's algorithm for finding shortest path [46] using only g-values for selecting the next node for expansion. It can be modified towards more A*-like algorithm by adding heuristic h-values [47] and also towards bounded sub-optimal algorithm where g values (or $g + h$ values) could differ from the correct value by factor $(1 + \epsilon)$ [33].

III. MRPP COMPILATION

A major alternative to CBS is represented by compilation of MRPP to Boolean satisfiability (SAT) [34], [48] and using the satisfiability modulo (SMT) theory approach [26].

A. SAT-based Approach

The idea is to construct a Boolean formula whose satisfiability corresponds to existence of a solution of sum-of-costs ξ to a given MRPP Σ. Moreover, the approach is constructive; that is, a solution to MRPP can be reconstructed from satisfying assignment of the formula.

There are two ways how to connect satisfiability of the formula and solvability of Σ: using either **equivalence** or **implication**.

Definition 2: **(complete Boolean model)** Boolean formula $\mathcal{F}(\xi)$ is a *complete Boolean model* of MRPP Σ if the following condition holds:

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

Complete Boolean models were the used in makespan optimal SAT-based solvers for MRPP [49] and in MDD-SAT [34], the first sum-of-costs optimal SAT-based solver. A natural relaxation from the complete Boolean model is an *incomplete Boolean model* where instead of the equivalence between solving MRPP and the formula we require an implication only. Incomplete models are inspired from the SMT paradigm are used in the recent sum-of-costs optimal solver SMT-CBS [26].

Definition 3: **(incomplete Boolean model).** Boolean formula $\mathcal{H}(\xi)$ is an *incomplete Boolean model* of MRPP Σ if the following condition holds:

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

Being able to construct formula $\mathcal{F}$ one can obtain optimal MRPP solution by checking satisfiability of $\mathcal{F}(0)$, $\mathcal{F}(1)$, $\mathcal{F}(2)$,... until the first satisfiable $\mathcal{F}(\xi)$ is met. This is possible due to monotonicity of MRPP solvability with respect to increasing values of common cumulative objectives such as the sum-of-costs. In practice it is however impractical to start at 0; lower bound estimation is used instead - sum of lengths of shortest paths can be used in the case of sum-of-costs. The framework of SAT-based solving is shown in pseudo-code in Algorithm 1.

Algorithm 1: Framework of SAT-based MRPP

```
1  SAT-MRPP (G = (V, E), R, α_0, α_+)
2      paths ← {shortest path from α_0(r_i) to α_+(r_i) | i = 1, 2, ..., k}
3      ξ ← Σ_{i=1}^{k} ξ(paths(r_i)); μ ← max_{i=1}^{k} ξ(paths(r_i))
4      while True do
5          F(ξ) ← encode-Complete(μ, ξ, G, R, 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; μ ← μ + 1
```

The advantage of the SAT-based approach is that state-of-the-art SAT solvers can be used for determining satisfiability of $\mathcal{F}(\xi)$ [38] and any progress in SAT solving hence can be utilized for increasing efficiency of MRPP solving.

B. Details of Boolean Encoding

We recall important aspects of Boolean encoding used in MDD-SAT and SMT-CBS. The construction of both $\mathcal{F}(\xi)$ and $\mathcal{H}(\xi)$ 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 (also refered to as a *makespan*) such that every possible solution of the given MRPP with the sum-of-costs less than or equal to ξ fits within μ timesteps (that is, no robot is outside its goal vertex after μ-th timestep if the sum-of-costs ξ is not to be exceeded).

The time expansion itself makes copies of vertices V for each timestep $t = 0, 1, 2, ..., \mu$. That is, we have vertices v^t for each $v \in V$ time step t. Edges from G are converted to directed edges interconnecting timesteps in 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, t^{t+1}). A directed path in time expansion corresponds to trajectory of a robot in time. Hence the modeling task now consists in construction of a formula in which satisfying assignments correspond to directed paths from $\alpha_0^0(r_i)$ to $\alpha_+^\mu(r_i)$ in the time expansion.

Assume that we have time expansion (V_i, E_i) for robot r_i. Boolean variable $\mathcal{X}_v^t(r_i)$ is introduced for every vertex v^t in V_i. The semantics of $\mathcal{X}_v^t(r_i)$ is that it is $TRUE$ if and only if robot r_i resides in v at time step t. Similarly we introduce $\mathcal{E}_{u,v}^t(r_i)$ for every directed edge (u^t, v^{t+1}) in E_i.

Next, constraints are added so that truth assignments are restricted to those that correspond to valid solutions of a given MRPP. Added constraints together ensure that $\mathcal{F}(\xi)$ is a *complete Boolean model* for given MRPP.

We here illustrate the model by showing few representative constraints. We omit here constraints that concern the objec-

tive function. For the detailed list of constraints we refer the reader to [35].

First, there is a constraint stating that if robot r_i appears in vertex u at time step t then it has to leave through exactly one edge (u^t, v^{t+1}). This can be established by following constraints:

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

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

Collisions between robots can be eliminated by the following constraint over $\mathcal{X}_v^t(r_i)$ variables for every $v \in V$ and timestep t. This constraint is however omitted in $\mathcal{H}(\xi)$ making $\mathcal{H}(\xi)$ essentially **incomplete** as its satisfying assignments may correspond to plans leading to a **collision**.

$$\sum_{r_i\in A \mid v^t\in V_i} \mathcal{X}_v^t(r_i) \leq 1 \tag{3}$$

C. SMT-based Approach

A common approach in *satisfiability modulo theories* (SMT) [37] for deciding the satisfiability problem in some complex theory T is to divide it into an abstract Boolean 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 *Boolean skeleton* by replacing atoms with Boolean variables. The standard SAT-solving procedure 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.

The above observation stands behind rephrasing CBS in terms of SMT, the SMT-CBS algorithm. The paths validation procedure acts as $DECIDE_T$ and reports back a set of conflicts found in the current solution. Hence axioms of T are represented by the movement rules of MRPP. SMT-CBS is listed as Algorithm 2.

The algorithm is divided into two procedures: SMT-CBS representing the main loop and SMT-CBS-Fixed solving the input MRPP for a 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-Complete* from MDD-SAT is replaced with *encode-Incomplete* that produces encoding that ignores specific movement rules (collisions between robots) but in contrast to *encode-Complete* it encodes collected conflicts into $\mathcal{H}(\xi)$.

The conflict resolution in standard CBS implemented as high-level branching is here represented by refinement of $\mathcal{H}(\xi)$ with disjunction (line 20). Branching is thus deferred into the SAT solver. The advantage of SMT-CBS is that it builds the formula lazily; that is, it adds constraints on demand after conflict occurs. Such approach may save resources as solution may be found before all constraint are added.

Algorithm 2: SMT-based MRPP solver

```
1  SMT-CBS (Σ = (G = (V, E), R, α0, α+))
2      conflicts ← ∅
3      paths ← {shortest path from α0(ri) to α+(ri) | i = 1, 2, ..., k}
4      ξ ← Σ_{i=1}^{k} ξ(paths(ri)); μ ← max_{i=1}^{k} ξ(paths(ri))
5      while TRUE do
6          (paths, conflicts) ← SMT-CBS-Fixed(conflicts, μ, ξ, Σ)
7          if paths ≠ UNSAT then
8              return paths
9          ξ ← ξ + 1; μ ← μ + 1
10 SMT-CBS-Fixed(conflicts, μ, ξ, Σ)
11     H(ξ) ← encode-Incomplete(conflicts, μ, 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 (ri, rj, v, t) ∈ collisions do
20                 H(ξ) ← H(ξ) ∪ {¬X_v^t(ri) ∨ ¬X_v^t(rj)}
21                 conflicts ← conflicts ∪ {[(ri, v, t), (rj, v, t)]}
22         return (UNSAT, conflicts)
```

IV. BOUNDED SUBOPTIMAL SMT-BASED SOLVER

The key to the new bounded-suboptimal SMT-based solver is the modification of cost bound ξ used in the construction of $\mathcal{H}(\xi)$. Inside $\mathcal{H}(\xi)$, the ξ bound is used to bound the sum-of-costs using the *cardinality constraint* [50]. Since now the number of time expansions will not be derived directly from ξ parameter we will denote the formula as $\mathcal{H}(\mu, \xi)$ where $\mu = \mu_0 + \Delta$ will be the number of time expansions, $\mu_0 = \max_{i=1}^{k} \xi(path(r_i))$ where $path(r_i)$ is the shortest path for r_i, and the second parameter will be the cost bound used in the cardinality constraint. Instead of incrementing ξ we will increment Δ.

In SMT-CBS, ξ is incremented by one in every iteration. Allowing $\xi = \xi_0 + \Delta$ parameter to be less restrictive, we will replace Δ with $\Delta' = \Delta + \delta$, where $\delta \geq 0$ is an integer value, producing a formula of the same size but representing more solutions [3]. Since $\Delta' > \Delta$, we expect a formula with the sum-of-costs bounded by Δ' to be easier to solve than that with the original Δ.

The following proposition shows that for a solvable MRPP Σ the sum-of-costs of the solution obtained by the above process differs from the optimal one by at most δ.

[3]The change from Δ to Δ' does not affect the number of clauses that represent the cardinality constraint, because we encode the cardinality constraints using a sequential counter, whose size is proportional to the number of Boolean variable involved but not to the value of the bound [51].

Algorithm 3: eSMT-CBS, an $(1+\epsilon)$-bounded suboptimal SMT-based MRPP solver

```
1  eSMT-CBS(Σ = (G = (V, E), R, α_0, α_+))
2      conflicts ← ∅
3      paths ← {shortest path from α_0(r_i) to α_+(r_i) | i = 1, 2, ..., k}
4      ξ_0 ← Σ_{i=1}^{k} ξ(paths(r_i)); μ_0 ← max_{i=1}^{k} ξ(paths(r_i))
5      Δ ← 0
6      while TRUE do
7          Δ' ← Δ + ε · (ξ_0 + Δ)
8          (paths, conflicts) ←
             SMT-CBS-Fixed(conflicts, μ_0 + Δ, ξ_0 + Δ', Σ)
9          if paths ≠ UNSAT then
10             return paths
11         Δ ← Δ + 1
```

Propositon 1: Let δ be a non-negative integer and let $\mathcal{H}(\mu_0 + \Delta, \xi_0 + \Delta + \delta)$ be the first satisfiable formula corresponding to a valid MRPP solusion encountered in the sequence of formulae $\mathcal{H}(\mu_0, \xi_0 + \delta)$, $\mathcal{H}(\mu_0 + 1, \xi_0 + 1 + \delta)$,...,$\mathcal{H}(\mu_0 + \Delta - 1, \xi_0 + \Delta + \delta - 1)$, $\mathcal{H}(\mu_0 + \Delta, \xi_0 + \Delta + \delta)$. Then solution represented by $\mathcal{H}(\mu_0 + \Delta, \xi_0 + \Delta + \delta)$ has sum-of-costs $\xi \leq \xi_{opt} + \delta$ where ξ_{opt} is the optimal sum-of-costs for Σ.

Proof: Formula $\mathcal{H}(\mu_0 + \Delta - 1, \Delta + \delta - 1)$ in the penultimate iteration could not be augmented by adding collision avoidance constraints to represent a valid solution and eventually became unsatisfiable. This means that no solution of makespan at most $\mu_0 + \Delta - 1$ and sum-of-costs at most $\xi_0 + \Delta + \delta - 1$ exists. But we also know that all solutions of sum-of-costs $\xi_0 + \Delta - 1$ fit under the makespan of at most $\mu_0 + \Delta - 1$. Hence unsolvability of formula $\mathcal{H}(\mu_0 + \Delta - 1, \Delta + \delta - 1)$ together with $\delta \geq 0$ implies that there is no solution of sum-of-costs $\xi_0 + \Delta - 1$ at all. Therefore, the optimal sum-of-costs is at least $\xi_0 + \Delta$. The solvability of $\mathcal{H}(\mu_0 + \Delta, \Delta + \delta)$ says that there is a solution of Σ of sum-of-costs $\xi_0 + \Delta + \delta$ which differs from the optimum by at most δ. ■

Observe that the only property of δ we used was that it is a non-negative integer but there is no requirement that it must be constant across individual iterations of the algorithm. Proposition 1 holds even if we use a non-negative δ as a function of Δ instead of a constant. This property can be used to modify the above SAT-based framework to an $(1+\epsilon)$-bounded suboptimal algorithm.

Corollary 1: Given an error $\epsilon > 0$ the SMT-based sub-optimal framework can modified to an $(1 + \epsilon)$-bounded suboptimal algorithm by appropriate setting of $\delta(\Delta)$.

Proof: Let $\delta(\Delta) = \epsilon \cdot (\xi_0 + \Delta)$. Hence the sum-of-costs of the solution returned by the algorithm is at most $(1+\epsilon) \cdot (\xi_0 + \Delta)$ while the optimum is at least $\xi_0 + \Delta$ hence the ratio between the sum-of-costs of returned solution and the sum-of-costs of the optimal one is at most $(1 + \epsilon)$. ■

The pseudo-code of the $(1+\epsilon)$-bounded suboptimal SMT-based algorithm is presented as Algorithm 3. We refer to this algorithm as **eSMT-CBS**.

Observe that in any solution to a MRPP problem it holds that $\mu \leq \xi \leq m \cdot \mu$. Therefore, if $\xi_0 + \Delta + \delta(\Delta) \geq \mu \cdot k$ then there is no need to add any cardinality constraints to $\mathcal{H}(\mu, \xi)$, as the solution is guaranteed to be bounded by $\mu \cdot k$.

This inequality represents a limit of the degree of relaxation achievable by allowing more freedom over the cost bound imposed by the cardinality constraint. Hence the eSMT-CBS algorithm will effectively be an $\left(\frac{k \cdot (\mu_0 + \Delta)}{\xi_0 + \Delta}\right)$-bounded algorithm in the worst case.

V. EXPERIMENTAL EVALUATION

We evaluated eSMT-CBS on standard benchmarks from `movingai.com` [15], [21], [52]. Representative part of results is presented in this section.

A. Benchmarks and Setup

We implemented eSMT-CBS in C++ using the existing implementation of SMT-CBS on top of the Glucose 3.0 SAT solver [53] that still ranks among the best SAT solvers according to recent SAT solver competitions [54]. Across the series of incremental refinements of $\mathcal{H}(\mu, \xi)$ the SAT is consulted in the incremental way. Concerning ECBS, we used existing implementation by [33] written in C#.

All experiments were run on system consisting of Xeon 2.8 GHz cores, 32 GB RAM, running Ubuntu Linux 18 (for testing eSMT-CBS) and Windows 10 (for ECBS tests). [4]

The experimental evaluation has been done on diverse instances consisting of 4-connected *grid* maps ranging in sizes from small to large.

These grid maps were obtained synthetically or as a discretization of real environments. In 4-connected grids, robots traverse the map using orthogonal movements; diagonal movements are not used, hence unit time per move can be realistically assumed. This makes 4-connected grids suitable for finding plans to be executed on physical robots. In a related study [13], we simulated discrete plans found for 4-connected grid graphs on a group of OZOBOT Evo robots. Minor deviations from the discrete plan occurring during execution can be compensated by reflex capabilities of robots so that synchronization of robots' movements with the discrete plan can be achieved.

We varied the number of robots to obtain instances of various difficulties while initial and goal configurations of robots were generated according to scenarios provided on `movingai.com`. Depending on the map size we used 64 to 128 robots and 25 different instances per number of robots. The timeout in all test was set to 128 seconds. Presented results were obtained from instances solved within this timeout. Both tested algorithms were used in 4 different setups with different error: $\epsilon = 1.00$ (optimal MRPP), $\epsilon = 1.01$, $\epsilon = 1.05$, and $\epsilon = 1.10$.

[4]To enable reproducibility of presented results we provide complete source code of our solvers and detailed experimental data on author's web: `http://users.fit.cvut.cz/~surynpav/iros2020`.

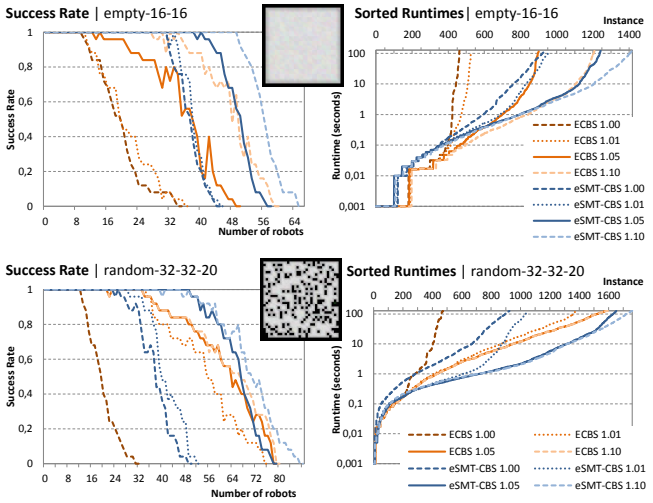

Fig. 2. Success rate and runtime comparison on small-sized maps.

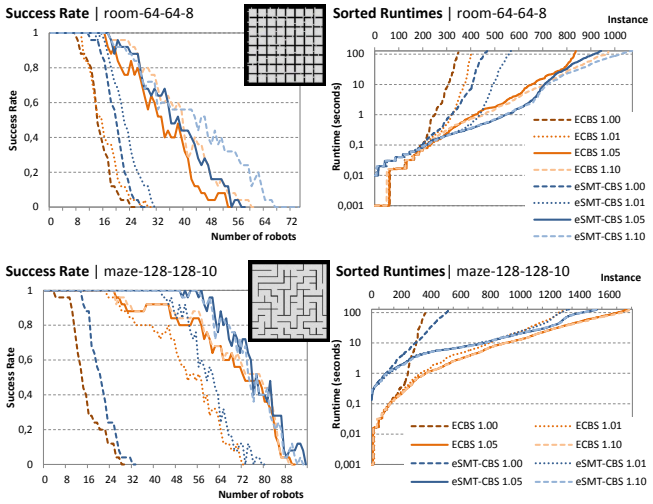

Fig. 3. Success rate and runtime comparison on medium-sized maps.

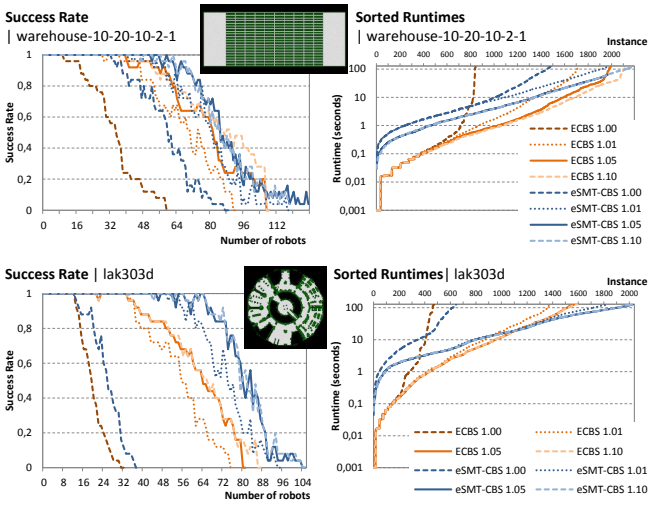

Fig. 4. Success rate and runtime comparison on large-sized maps.

B. Runtime Results

Results are presented in Figures 2, 3, and 4. We present *success rate* and *sorted runtimes*. Success rate shows the ratio of instances solved under the time limit of 128 seconds out of 25 instances per number of robots. Sorted runtimes are inspired by cactus plots from the SAT Competition [54]. We took runtimes of all instances solved under the time limit by a given algorithm and sorted them along x-axis; so the x-th data-point represents the runtime of x-th easiest instance for the given algorithm. The faster algorithm yields to a lower curve in the cactus plot.

The general trend observable across all tested values of ϵ and all maps from sorted runtimes is that ECBS is faster for easy instances but its performance quickly degrades as instances gets harder. This is quite expectable since ECBS compared to eSMT-CBS has smaller overhead but on the other hand lacks advanced learning and propagation mechanisms that help the SMT-based solver in harder cases.

As instances get harder, eSMT-CBS starts to perform better than ECBS. In some cases ECBS experience sharp increase in the runtime after crossing certain level of difficulty (this is well observable in `empty-16-16` with $\epsilon = 1.01$ and $\epsilon = 1.05$, and in `room-64-64-8` with $\epsilon = 1.01$ and $\epsilon = 1.05$).

There is significant difference in how relaxing the ϵ parameter reduces the difficulty. The most significant change happens by switching from the optimal MRPP ($\epsilon = 1.00$) to slightly sub-optimal ($\epsilon = 1.01$) which dramatically reduces the difficulty for both tested solvers (this change is the most dramatic in case of ECBS on `random-32-32-20`). On the other hand, changing ϵ from 1.05 to 1.10 sometimes has almost no effect (especially in large instances like `warehouse-10-20-10-2-1` and `lak303d`).

The disadvantage of earlier SAT-based MRPP solvers was worse scalability for large maps compared to CBS/EBCS which was caused by the need to construct a huge formula. This has been largely eliminated in SMT-CBS that constructs incomplete Boolean model that is significantly smaller than the complete model even on large maps. Therefore eSMT-CBS maintains its leads over ECBS even in large maps.

VI. CONCLUSION

We introduced eSMT-CBS, a novel bounded sub-optimal algorithm for MRPP based on satisfiability modulo theories (SMT). The new algorithm combines strengths of SAT-based solving, which due to powerful clause learning mechanism and Boolean constraint propagation can successfully solve combinatorially hard cases of MRPP, with lazy construction of the Boolean formula using the SMT-inspired mechanism that enables scalability of the solver even for large maps.

The advantage of eSMT-CBS appears in harder instances with long runs of the SAT solver where the clause learning mechanism has enough time to prune the search space efficiently. On the other hand the SMT-based approach has an overhead of building formula and communication with the external solver which negatively affects performance in sparsely occupied instances.

One of possible future research directions is to integrate the SAT solver and the high-level MRPP solving scheme more closely. Currently we need to wait for a complete

assignment of Boolean variables before the extracted paths are validated with respect to MRPP rules. Potentially we can validate paths extracted from **partial assignments** as done in DPLL(T) solvers.

REFERENCES

[1] D. Silver, "Cooperative pathfinding," in *AIIDE*, 2005, pp. 117–122.

[2] M. R. K. Ryan, "Graph decomposition for efficient multi-robot path planning," in *Proceedings of IJCAI*, 2007, pp. 2003–2008.

[3] T. Standley, "Finding optimal solutions to cooperative pathfinding problems." in *Proceedings of AAAI*, 2010, pp. 173–178.

[4] J. Yu and S. M. LaValle, "Planning optimal paths for multiple robots on graphs," in *Proceedings of ICRA*, 2013, pp. 3612–3617.

[5] R. M. Wilson, "Graph puzzles, homotopy, and the alternating group," *Journal of Combinatorial Theory, Series B*, vol. 16, no. 1, pp. 86 – 96, 1974.

[6] D. Kornhauser, G. L. Miller, and P. G. Spirakis, "Coordinating pebble motion on graphs, the diameter of permutation groups, and applications," in *Proceedings of FOCS*, 1984, pp. 241–250.

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

[8] R. Luna and K. E. Bekris, "Network-guided multi-robot path planning in discrete representations," in *Proceedings of IROS*, 2010, pp. 4596–4602.

[9] F. Basile, P. Chiacchio, and J. Coppola, "A hybrid model of complex automated warehouse systems - part I: modeling and simulation," *IEEE Trans. Automation Science and Engineering*, vol. 9, no. 4, pp. 640–653, 2012.

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

[11] D. Zhou and M. Schwager, "Virtual rigid bodies for coordinated agile maneuvering of teams of micro aerial vehicles," in *Proceedings of ICRA*, 2015, pp. 1737–1742.

[12] A. Andreychuk, K. S. Yakovlev, D. Atzmon, and R. Stern, "Multi-agent pathfinding with continuous time," in *Proceedings of IJCAI*. ijcai.org, 2019, pp. 39–45.

[13] J. Chudy, N. Popov, and P. Surynek, "Multi-agent path finding simulation with a swarm of physical robots: cooperative behavior via reflex-based control," in *Proceedings of ROBOVIS*. SCITEPRESS, to appear, 2020.

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

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

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

[17] P. Surynek, "An optimization variant of multi-robot path planning is intractable," in *Proceedings of AAAI*. AAAI Press, 2010.

[18] ——, "Compact representations of cooperative path-finding as SAT based on matchings in bipartite graphs," in *Proceedings of ICTAI*, 2014, pp. 875–882.

[19] ——, "Reduced time-expansion graphs and goal decomposition for solving cooperative path finding sub-optimally," in *IJCAI*, 2015, pp. 1916–1922.

[20] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artif. Intell.*, vol. 219, pp. 1–24, 2015.

[21] 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*, 2015, pp. 740–746.

[22] M. Ryan, "Constraint-based multi-robot path planning," in *Proceedings of ICRA*, 2010, pp. 922–928.

[23] P. Surynek, "Towards optimal cooperative path planning in hard setups through satisfiability solving," in *PRICAI*, 2012, pp. 564–576.

[24] J. Yu and S. LaValle, "Planning optimal paths for multiple robots on graphs," in *Proceedings of ICRA*, 2013, pp. 3612–3617.

[25] E. Erdem, D. G. Kisa, U. Oztok, and P. Schueller, "A general formal framework for pathfinding problems with multiple agents," in *Proceedings of AAAI*, 2013.

[26] P. Surynek, "Unifying search-based and compilation-based approaches to multi-agent path finding through satisfiability modulo theories," in *Proceedings of IJCAI*. ijcai.org, 2019, pp. 1177–1183.

[27] L. Cohen, T. Uras, and S. Koenig, "Feasibility study: Using highways for bounded-suboptimal mapf," in *SOCS*, 2015, pp. 2–8.

[28] A. Botea and P. Surynek, "Multi-agent path finding on strongly biconnected digraphs," in *Proceedings of AAAI*, 2015, pp. 2024–2030.

[29] Q. Sajid, R. Luna, and K. Bekris, "Multi-agent pathfinding with simultaneous execution of single-agent primitives," in *Proceedings of SoCS*, 2012.

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

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

[32] P. Surynek, "A novel approach to path planning for multiple robots in bi-connected graphs," in *Proceedings of ICRA*, 2009, pp. 3613–3619.

[33] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Proceedings of SoCS*, 2014.

[34] P. Surynek, A. Felner, R. Stern, and E. Boyarski, "Sub-optimal sat-based approach to multi-agent path-finding problem," in *Proceedings of SoCS*. AAAI Press, 2018, pp. 90–105.

[35] ——, "Efficient SAT approach to multi-agent path finding under the sum of costs objective," in *Proceedings of ECAI 2016*, 2016, pp. 810–818.

[36] R. Nieuwenhuis, A. Oliveras, and C. Tinelli, "Solving SAT and SAT modulo theories: From an abstract davis–putnam–logemann–loveland procedure to dpll(T)," *J. ACM*, vol. 53, no. 6, pp. 937–977, 2006.

[37] M. Bofill, M. Palahí, J. Suy, and M. Villaret, "Solving constraint satisfaction problems with SAT modulo theories," *Constraints*, vol. 17, no. 3, pp. 273–303, 2012.

[38] G. Audemard, J. Lagniez, and L. Simon, "Improving glucose for incremental SAT solving with assumptions: Application to MUS extraction," in *Proceedings of SAT*, 2013, pp. 309–317.

[39] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Proceedings of ECAI*, 2014, pp. 961–962.

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

[41] P. Surynek, "Solving abstract cooperative path-finding in densely populated environments," *Computational Intelligence*, vol. 30, no. 2, pp. 402–450, 2014.

[42] R. Luna and K. Bekris, "Efficient and complete centralized multi-robot path planning," in *Proceedings of IROS*, 2011, pp. 3268–3275.

[43] J. Yu and S. M. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Proceedings of AAAI*, 2013.

[44] H. Ma, C. A. Tovey, G. Sharon, T. K. S. Kumar, and S. Koenig, "Multi-agent path finding with payload transfers and the package-exchange robot-routing problem," in *Proceedings of AAAI 2016*. AAAI Press, 2016, pp. 3166–3173.

[45] 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.

[46] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[47] J. Li, A. Felner, E. Boyarski, H. Ma, and S. Koenig, "Improved heuristics for multi-agent path finding with conflict-based search," in *Proceedings of IJCAI*. ijcai.org, 2019, pp. 442–449.

[48] P. Surynek, "On propositional encodings of cooperative path-finding," in *Proceedings of ICTAI*. IEEE, 2012, pp. 524–531.

[49] ——, "Time-expanded graph-based propositional encodings for makespan-optimal solving of cooperative path finding problems," *Ann. Math. Artif. Intell.*, vol. 81, no. 3-4, pp. 329–375, 2017.

[50] J. Silva and I. Lynce, "Towards robust CNF encodings of cardinality constraints," in *Proceedings of CP*, 2007, pp. 483–497.

[51] C. Sinz, "Towards an optimal CNF encoding of boolean cardinality constraints," in *Proceedings of CP*, 2005.

[52] N. Sturtevant, "Benchmarks for grid-based pathfinding, http://www.movingai.com," *Trans. on Computational Intelligence and AI in Games*, vol. 4, no. 2, pp. 144 – 148, 2012.

[53] G. Audemard and L. Simon, "Predicting learnt clauses quality in modern SAT solvers," in *Proceedings of IJCAI*, 2009, pp. 399–404.

[54] T. Balyo, M. J. H. Heule, and M. Järvisalo, "SAT competition 2016: Recent developments," in *Proceedings of AAAI*, 2017, pp. 5061–5063.