Lazy Compilation of Variants of Multi-robot Path Planning with Satisfiability Modulo Theory (SMT) Approach

Pavel Surynek[1]

***Abstract*— We address variants of multi-robot path planning in graphs (MRPP). We assume robots placed in vertices of an undirected graph with at most one robot per vertex. Robots can move across edges while various problem specific constraints must be satisfied. We introduce a general problem formulation that encompasses known types of robot relocation problems such as multi-robot path planning (MRPP), token swapping (TSWAP), token rotation (TROT), and token permutation (TPERM). We generalize SMT-CBS, a recent solving approach for MRPP based on satisfiability modulo theories (SMT). SMT-CBS compiles MRPP lazily within the SMT framework, starting with the basic model that is refined with a collision resolution constraints whenever collisions between robots occur in the current solution. We show modifications the SMT-CBS algorithm for variants of MRPP and evaluate them experimentally.**

I. INTRODUCTION AND MOTIVATION

Multi-robot path planning in graphs (MRPP) [1], [2] and related problems such as *token swapping* (TSWAP) [3], [4], pebble motion on graphs (PMG) [5] represent important combinatorial problems for motion planning in robotics. We assume that multiple distinguishable robots are placed in vertices of an undirected graph such that at most one robot is placed in each vertex. Robots can move across edges while problem specific rules are observed. For instance, PMG and MRPP usually requires that robots (pebbles/robots) are moved to unoccupied neighbors only. TSWAP on the other hand permits only swaps of pairs of tokens along edges while more complex movements are forbidden. The task in robot path planning problems is to reach a given goal configuration of robots from a given starting configuration using allowed movements.

In this paper we focus on optimal solving of MRPP and its variants with respect to common cumulative objective functions like *sum-of-costs* [6] and *makespan* [7]. Sum-of-costs corresponds to the total cost of all movements performed until the goal configuration in reached - traversal of an edge by an robot has unit cost typically. The makespan calculates the number of time-steps until the goal is reached. In both cases we trying to minimize the objective.

Many practical problems from robotics can be interpreted as MRPP. Examples include discrete multi-robot navigation and coordination, robot rearrangement in automated warehouses [8], ship collision avoidance [9], or formation maintenance of aerial vehicles [10].

[1]FIT, 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.

A. Contributions

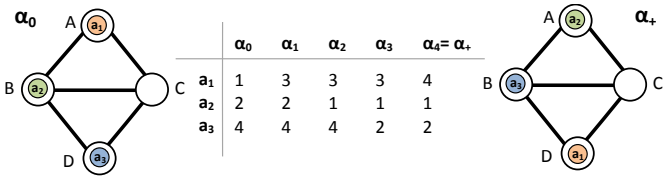
The contribution of this paper consists in suggesting a general framework for defining and solving MRPP based on *satisfiability modulo theories* (SMT) [11]. We used the framework to define two new variants derived from MRPP and TSWAP: *token rotation* (TROT) and *token permutation* (TPERM) where instead of swapping tokens, rotations along non-trivial cycles and arbitrary permutations of tokens are permitted. A recently suggested algorithm called SMT-CBS [12] that combines ideas from CBS [13] and SMT is adapted for all variants of MRPP and experimentally evaluated. Tests on standard benchmarks indicate that SMT-CBS outperforms the previous CBS and the previous state-of-the-art SAT-based algorithm MDD-SAT [14].

We first recall TSWAP and MRPP formally. The SMT-CBS algorithm is recalled in the line with its precursors - the CBS and MDD-SAT algorithms. Together with SMT-CBS we introduce TROT and TPERM. Finally, an experimental evaluation of all concerned algorithms CBS, MDD-SAT, and SMT-CBS on all variants of MRPP is presented.

II. BACKGROUND

We first recall *multi-robot path planning* (MRPP) [15], [16] and *token swapping* (TSWAP) [17] formally. MRPP consists of an undirected graph $G = (V, E)$ and a set of robots $R = \{r_1, r_2, ..., r_k\}$ such that $|R| < |V|$. Each robot is placed in a vertex so that at most one robot resides in each vertex. The configuration of robots is denoted $\alpha : R \rightarrow V$ with α_0 and α_+ denoting the initial and the goal configuration respectively.

A robot can either *move* to an adjacent empty vertex or *wait* in its current location. The task is to find a sequence of move/wait actions for each robot r_i, moving it from $\alpha_0(r_i)$ to $\alpha_+(r_i)$ such that robots do not *collide*, that is no two robots can enter the same target vertex simultaneously. An example of MRPP instance is shown in Figure 1.



	α_0	α_1	α_2	α_3	$\alpha_4 = \alpha_+$
a_1	1	3	3	3	4
a_2	2	2	1	1	1
a_3	4	4	4	2	2

Fig. 1. An MRPP instance with three robots r_1, r_2, and a_3.

Definition 1: **(move-to-unoccupied rule of MRPP)** Configuration α' *results* from α if and only if the following conditions hold:

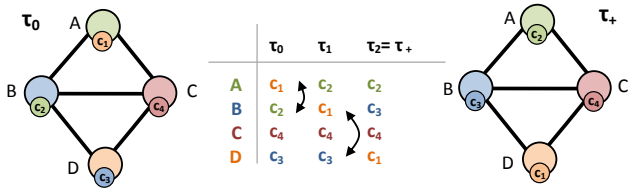


Fig. 2. A TSWAP instance. A solution consisting of two swaps is shown.

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

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

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

The task in MRPP is to find a sequence of configurations $[\alpha_0, \alpha_1, ..., \alpha_\mu]$ such that α_{i+1} results from α_i for $i = 1, 2, ..., \mu - 1$, and $\alpha_\mu = \alpha_+$.

A feasible solution of a solvable MRPP instance can be found in polynomial time [5]; 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 $\mathcal{O}(|V|^3)$) [18]. This is also asymptotically best what can be done as there are MRPP instances requiring $\Omega(|V|^3)$ moves.

In many aspects, *token swapping problem (TSWAP)* (also known as *sorting on graphs*) is similar to MRPP. Using a modified notation from [19] each vertex in G is assigned a color from $C = \{c_1, c_2, ..., c_h\}$ via $\tau_+ : V \to C$. A token of a color from C is placed in each vertex. The task is to transform a given token placement into the one such that colors of tokens and respective vertices of their placement agree. Desirable token placement can be obtained by swapping tokens on adjacent vertices. See Figure 2 for an example instance of TSWAP.

We denote by $\tau : V \to C$ colors of tokens placed in vertices of G. That is, $\tau(v)$ for $v \in V$ is a color of a token placed in v. Starting placement of tokens is denoted as τ_0; the goal token placement corresponds to τ_+. Transformation of one placement to another is captured by the concept of *adjacency* defined as follows [19]:

Definition 2: **(adjacency in TSWAP)** Token placements τ and τ' are *adjacent* if there exists a subset of non-adjacent edges $F \subseteq E$ such that $\tau(v) = \tau'(u)$ and $\tau(u) = \tau'(v)$ for each $\{u, v\} \in F$ and for all other vertices $w \in V \setminus \bigcup_{\{u,v\} \in F} \{u, v\}$ it holds that $\tau(w) = \tau'(w)$.

The task in TSWAP is to find a swapping sequence of token placements $[\tau_0, \tau_1, ..., \tau_m]$ such that $\tau_m = \tau_+$ and τ_i and τ_{i+1} are adjacent for all $i = 0, 1, ..., m - 1$.

It has been shown that for any initial and goal placement of tokens τ_0 and τ_+ respectively there is a swapping sequence transforming τ_0 and τ_+ containing $\mathcal{O}(|V|^2)$ swaps. The proof is based on swapping tokens on a spanning tree of G. We note that the above bound is tight as there are instances consuming $\Omega(|V|^2)$ swaps.

III. RELATED WORK

Although many works studying TSWAP from the theoretical point of view exist [20] practical solving of the problem started only lately [21] .

A. Search for Optimal Solutions

We will use the *sum-of-costs* objective function in all problems studied in this paper. But all presented concepts can be migrated to other cumulative objectives. The following definition introduces the sum-of-costs objective in MRPP. Analogical definition can be introduced for TSWAP too.

Definition 3: **Sum-of-costs** [22] $\xi = \sum_{i=1}^{k} \xi(path(r_i))$ 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.

We note that one swap in TSWAP corresponds to the cost of 2 as two tokens traverses single edge. Finding optimal MRPP and TSWAP solutions with respect to various cumulative objectives is NP-hard [7].

B. Conflict-based Search

CBS uses the idea of resolving conflicts lazily; that is, a solution of MRPP instance 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 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 $N.constraints$. This is 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 [23].

CBS stores nodes of CT into priority queue OPEN sorted according to ascending $N.\xi$. At each step CBS takes node N with lowest $N.\xi$ from OPEN and checks if $N.paths$ represent collision free paths. 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. Assuming that a collision occurred between robots r_i and r_j in vertex v at time step t it 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 the 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 [13].

C. SAT-based Approach

An alternative approach to optimal MRPP solving as well as to TSWAP solving is represented by reduction of MRPP to propositional satisfiability (SAT) as done in MDD-SAT [14]. The idea is to construct a propositional formula $\mathcal{F}(\xi)$ such that it is satisfiable if and only if a solution of a given MRPP of sum-of-costs ξ exists.

Being able to construct such formula $\mathcal{F}$ one can obtain optimal MRPP 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, where ξ_0 is the sum of lengths of shortest paths serving as the lower bound for the sum-of-costs. This is possible due to monotonicity of MRPP solvability with respect to increasing values of common cumulative objectives such as the sum-of-costs.

D. Details of MDD-SAT Encoding

Construction of $\mathcal{F}(\xi)$ relies on the *time expansion* of 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 with the sum-of-costs less than or equal to ξ fits within μ timesteps.

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 $TEG_i = (V_i, E_i)$ for robot r_i. Propositional variable $\mathcal{X}_v^t(r_j)$ 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. Analogously the meaning of $\mathcal{E}_{u,v}^t(r_i)$ is that it is $TRUE$ if and only if robot r_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 MRPP. For the detailed list of constraints we refer the reader to [14]. We here illustrate the encoding by showing a representative constraints saying that the target vertex of any movement except wait action must be empty. This is ensured by the following propositional expression for every $(u^t, v^{t+1}) \in E_i$:

$$\mathcal{E}_{u,v}^t(a_i) \Rightarrow \bigwedge_{a_j \in A \mid a_j \neq a_i \wedge v^t \in V_j} \neg \mathcal{X}_v^t(a_j) \qquad (1)$$

IV. GENERALIZATIONS OF ROBOT RELOCATION

We define two problems derived from MRPP and TSWAP: *token rotation* (TROT) and *token permutation* (TPERM) [1].

A. Token Rotation and Token Permutation

A swap of pair of tokens can be interpreted as a rotation along a trivial cycle consisting of a single edge. We can generalize this towards longer cycles. TROT permits rotations along longer cycles but forbids trivial cycles; that is, rotations along triples, quadruples, ... of vertices are permitted but swaps along edges are forbidden.

Definition 4: **(adjacency in TROT)** Token placements τ and τ' are *adjacent* in TROT if there exists a subset of edges $F \subseteq E$ such that components $C_1, C_2, ..., C_p$ of induced subgraph $G[F]$ satisfy following conditions:

(i) $C_j = (V_j^C, E_j^C)$ such that $V_j^C = w_1^j, w_2^j, ..., w_{n_j}^j$ with $n_j \leq 3$ and
$E_j^C = \{\{w_1^j, w_2^j\}; \{w_2^j, w_3^j\}; ...; \{w_{n_j}^j, w_1^j\}\}$
(components are cycles of length at least 3)

(ii) $\tau(w_1^j) = \tau'(w_2^j)$, $\tau(w_3^j) = \tau'(w_3^j)$, ..., $\tau(w_{n_j}^j) = \tau'(w_1^j)$
(colors are rotated in the cycle one position forward/backward)

The rest of the definition of a TROT instance is analogous to TSWAP. Similarly we can define TPERM by permitting all lengths of cycles. The formal definition of *adjacency* in TPERM is almost the same as in TROT except relaxing the constraint on cycle length, $n_j \leq 2$.

We omit here complexity considerations for TROT and TPERM for the sake of brevity. Again it holds that a feasible solution can be found in polynomial time but the optimal cases remain intractable in general.

In many real-life scenarios it happens that robots enter positions being simultaneously vacated by other robots (for example mobile robots often move formations like snake etc.) which is the property captured in above definitions.

B. Adapting CBS and MDD-SAT

Both CBS and MDD-SAT can be modified for optimal solving of TSWAP, TROT, and TPERM. Different movement rules can be reflected in CBS and MDD-SAT algorithms without modifying their high level framework.

1) Different Conflicts in CBS: In CBS, we need to modify the understanding of conflict between robots/tokens. To define conflicts in variants of MRPP we use the concept of *vertex collision* [23] and *edge collision* [25].

TPERM: The easiest case is TPERM as it is least restrictive. We merely forbid simultaneous occurrence of multiple tokens in a vertex - this situation is understood as a collision in TPERM and conflicts are derived from it. If a collision (c_i, c_j, v, t) between tokens c_i and c_j occurs in v at time step t then we introduce conflicts (c_i, v, t) and (c_j, v, t) for c_i and c_j respectively. [2]

[1] These problems have been considered in the literature in different contexts already (for example in [24]). But not from the practical solving perspective focused on finding optimal solutions.

[2] In addition to this MRPP checks vacancy of the target vertex which may cause more colliding situations.

TSWAP: This problem takes conflicts from TPERM but adds new conflicts that arise from doing something else than swapping [21]. Each time edge $\{u, v\}$ is being traversed by token c_i between time steps t and $t+1$, a token residing in v at time step t, that is $\tau_t(v)$, must go in the opposite direction from v to u. If this is not the case, then *edge collision* involving edge $\{u, v\}$ occurs and corresponding *edge conflicts* $(c_i, (u, v), t)$ and $(\tau_t(v), (v, u), t)$ are introduced for robots c_i and $\tau_t(v)$ respectively.

Edge conflicts must be treated at the low level of CBS. Hence in addition to forbidden vertices at given time-steps we have forbidden edges between given time-steps.

TROT: The treatment of conflicts will be complementary to TSWAP in TROT. Each time edge $\{u, v\}$ is being traversed by token c_i between time steps t and $t + 1$, a token residing in v at time step t, that is $\tau_t(v)$, must go anywhere else but not to u. If this is not the case, then we again have edge collision $(c_i, \tau_t(v)$,{u,v}$, t)$ which is treated in the same way as above.

2) Encoding Changes in MDD-SAT: In MDD-SAT, we need to modify encoding of movement rules in propositional formula $\mathcal{F}(\xi)$.

TPERM: This is the easiest case for MDD-SAT. We merely remove all constrains requiring tokens to move into vacant vertices only. That is we remove clauses (1).

TSWAP: Inherits changes from TPERM but in addition to that we need to carry out swaps properly. Edge variables $\mathcal{E}^t_{u,v}(c_i)$ will hence be utilized. The following constraint will be introduced for every $\{u^t, v^{t+1}\} \in E_i$ (intuitively, if token c_i traverses $\{u, v\}$ some other token c_j traverses $\{u, v\}$ in the opposite direction):

$$\mathcal{E}^t_{u,v}(c_i) \Rightarrow \bigvee_{j=1,2,\ldots,k \mid j \neq i \wedge (u^t, v^{t+1}) \in E_j} \mathcal{E}^t_{v,u}(c_j) \quad (2)$$

TROT: Is treated in a complementary way to TSWAP. Instead of adding constraints (2) we add constraints forbidding simultaneous traversal in the opposite direction as follows:

$$\mathcal{E}^t_{u,v}(c_i) \Rightarrow \bigwedge_{j=1,2,\ldots,k \mid j \neq i \wedge (u^t, v^{t+1}) \in E_j} \neg\mathcal{E}^t_{v,u}(c_j) \quad (3)$$

V. COMBINING SAT-BASED APPROACH AND CBS

A close look at CBS reveals that it operates similarly as problem solving in *satisfiability modulo theories* (SMT) [11]. 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 conjunctive fragment of T. A general T-formula Γ is transformed to a *propositional skeleton* by replacing atoms with propositional 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.

In the context of MRPP, the abstract propositional part working with the skeleton is taken from the MDD-SAT encoding provided that only constraints ensuring that assignments form valid paths interconnecting starting positions with goalsare kept. Other constraints for collision avoidance are omitted initially. The paths validation procedure will act as $DECIDE_T$ and will report back a set of conflicts found in the current solution. Hence axioms of T are represented by the movement rules of MRPP, TSWAP, TROT, and TPERM respectively.

The SMT-CBS algorithm based on the above idea is shown in pseudo-code as Algorithm 1 (it is formulated for MRPP; but is applicable for TSWAP, TPERM, and TROT after replacing conflict resolution part).

Algorithm 1: Framework of SMT-based MRPP solving

```
1  SMT-CBS (Σ = (G = (V, E), R, α0, α+))
2      conflicts ← ∅
3      paths ← {path*(ri) a shortest path from α0(ri) to
         α+(ri) | i = 1, 2, ..., k}
4      ξ ← Σ_{i=1}^{k} ξ(paths(ri))
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 (ri, rj, v, t) ∈ collisions do
20                 H(ξ) ← H(ξ) ∪ {¬X^t_v(ri) ∨ ¬X^t_v(rj)}
21                 conflicts ←
                     conflicts ∪ {[(ri, v, t), (rj, v, t)]}
22         return (UNSAT,conflicts)
```

The conflict resolution from the standard CBS implemented as high-level branching is here represented by refinement of $\mathcal{F}(\xi)$ with disjunction (line 20). Branching is thus deferred into the SAT solver. The advantage of SMT-CBS in contrast to MDD-SAT 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.

VI. EXPERIMENTAL EVALUATION

We performed an extensive evaluation of all presented algorithms on standard benchmarks [26]. Representative part of results is presented in this section.

We implemented SMT-CBS for all variants of MRPP in C++ on top of the Glucose 4 SAT solver [27]. 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 [23] that we need in our tests. And finally we took existing implementation of MDD-SAT also written in C++ and modified it for all problem variants.

All experiments were run on a system consisting of 120 Xeon 2.0 GHz cores, 512 GB RAM, running Ubuntu Linux 18. [3]

The experimental evaluation has been done on diverse instances consisting of 4-connected *grid* of size 8×8, *random graphs* containing 20% of random edges, *star* graphs, and *cliques*. Initial and goal configurations of tokens/robots have been generated randomly .

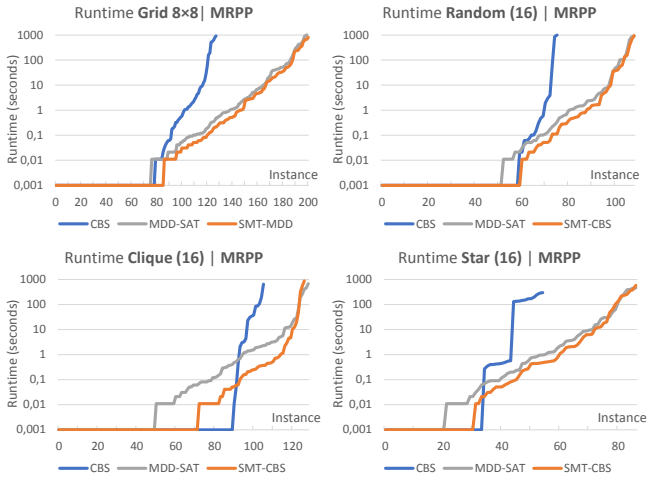


Fig. 3. Runtime comparison of CBS, MDD-SAT, and SMT-CBS on small graph instances of MRPP.

We varied the number of robots to obtain instances of various difficulties; that is, the underlying graph was not fully occupied - which in MRPP has natural meaning while in token problems we use one special color $\perp \in C$ that stands for an empty vertex (that is, we understand v as empty if and only if $\tau(v) = \perp$). For each number of robots/tokens we generated 10 random instances.

The timeout in all test was set to 1000 seconds. Presented results were obtained from instances solved within this timeout.

A. Evaluation on Small Graphs

Our tests were focused on the runtime comparison and evaluation of the size of encodings in case of MDD-SAT and SMT-CBS. Part of results we obtained in small graphs is presented in Figures 3 and 4.

CBS performs well in easy instances but its performance degrades quickly. Both MDD-SAT and SMT-CBS are faster for instances containing more robots. For hardest instances solvable under the timeout (in the 8×8 grid the hardest

[3]To enable reproducibility of presented results we provide complete source code of our solvers on author's web: `http://users.fit.cvut.cz/~surynpav/iros2019`.

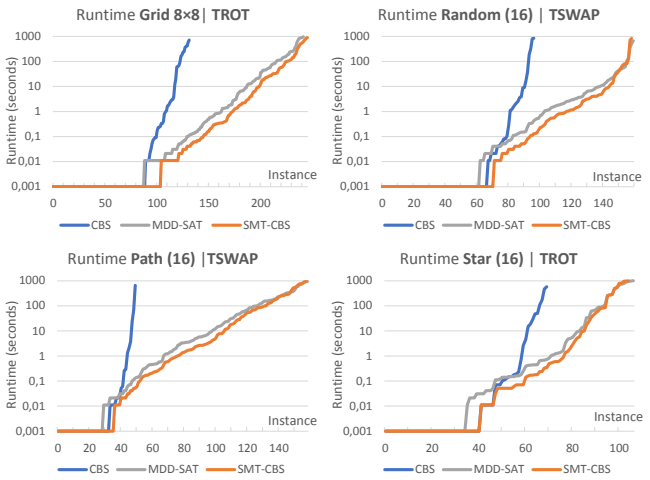


Fig. 4. Runtime comparison of CBS, MDD-SAT, and SMT-CBS on small graph instances of TSWAP and TROT.

instance contains 20 robots) 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 say medium difficulty - here SMT-CBS dominates over MDD-SAT by factor of 2 to 10.

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

B. Evaluation on Large Graphs

The second group of tests was focused on the performance of CBS, MDD-SAT and SMT-CBS on large 4-connected maps taken from *Dragon Age* [23], [28]. In contrast to small instances, these were only sparsely populated with agents. Initial and goal configuration were generated at random again. In the three structurally different maps up to 50 agents were placed randomly. Again we had 10 random instances per each number of agents.

Sorted runtimes are reported in Figures 5 and 6 with MRPP and TSWAP respectively. There is no significant difference between CBS and SMT-CBS in easier cases but MDD-SAT lags behind. The situation changes after going into medium difficulty region where runtimes of CBS go quickly up while SMT-CBS maintains significant advantage (factor 2 to 5) over MDD-SAT. Eventually however the performance of SMT-CBS and MDD-SAT meets in the hard region.

In addition to runtime comparison, we compared the number of clauses generated by MDD-SAT and SMT-CBS. Sorted numbers of clauses are shown in Figure 7 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 when compared to MDD-SAT.

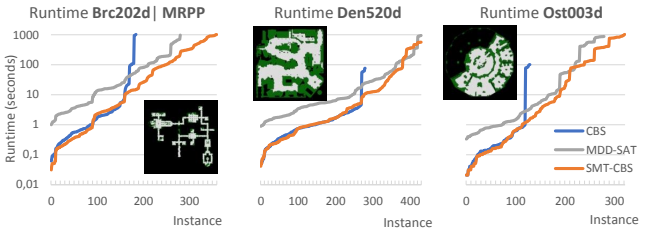


Fig. 5. Runtime comparison of CBS, MDD-SAT and SMT-CBS on large instances of MRPP.

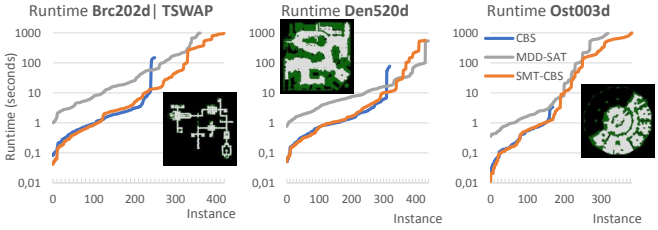


Fig. 6. Runtime comparison of CBS, MDD-SAT and SMT-CBS on large instances of TSWAP.

VII. CONCLUSIONS

We introduced a general framework for reasoning about robot path planning problems in graphs based on concepts from SMT. In addition to two known problems MRPP and TSWAP we introduced two derived variants TROT and TPERM in this context. We modified the recent algorithm SMT-CBS for TROT and TPERM. Experimental evaluation showed that SMT-CBS significantly outperforms previous state-of-the-art SAT-based solver MDD-SAT in instances of medium difficulty across all variants of the MRPP problem. Comparison of individual variants of the problem indicated that TPERM is the easiest, TROT in the middle, and TSWAP together with MRPP represent the hardest variants.

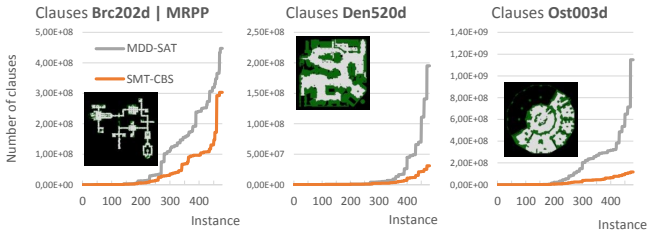


Fig. 7. Clauses generated by MDD-SAT and SMT-CBS on large MRPP instances.

REFERENCES

[1] M. R. K. Ryan, "Graph decomposition for efficient multi-robot path planning," in *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007, pp. 2003–2008.

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

[3] J. Kawahara, T. Saitoh, and R. Yoshinaka, "The time complexity of the token swapping problem and its parallel variants," in *WALCOM 2017 Proceedings.*, ser. LNCS, vol. 10167. Springer, 2017, pp. 448–459.

[4] É. Bonnet, T. Miltzow, and P. Rzazewski, "Complexity of token swapping and its variants," in *STACS 2017*, ser. LIPIcs, vol. 66. Schloss Dagstuhl, 2017, pp. 16:1–16:14.

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

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

[7] J. Yu and S. M. LaValle, "Optimal multi-robot path planning on graphs: Structure and computational complexity," *CoRR*, vol. abs/1507.03289, 2015.

[8] 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.

[9] 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.

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

[11] 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.

[12] P. Surynek, "Unifying search-based and compilation-based approaches to multi-agent path finding through satisfiability modulo theories," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019*, 2019.

[13] 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.

[14] P. Surynek, A. Felner, R. Stern, and E. Boyarski, "Efficient SAT approach to multi-agent path finding under the sum of costs objective," in *ECAI*, 2016, pp. 810–818.

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

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

[17] K. Yamanaka, E. D. Demaine, T. Ito, J. Kawahara, M. Kiyomi, Y. Okamoto, T. Saitoh, A. Suzuki, K. Uchizawa, and T. Uno, "Swapping labeled tokens on graphs," in *FUN 2014 Proceedings*, ser. LNCS, vol. 8496. Springer, 2014, pp. 364–375.

[18] 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.

[19] K. Yamanaka, E. D. Demaine, T. Ito, J. Kawahara, M. Kiyomi, Y. Okamoto, T. Saitoh, A. Suzuki, K. Uchizawa, and T. Uno, "Swapping labeled tokens on graphs," *Theor. Comput. Sci.*, vol. 586, pp. 81–94, 2015.

[20] T. Miltzow, L. Narins, Y. Okamoto, G. Rote, A. Thomas, and T. Uno, "Approximation and hardness of token swapping," in *ESA 2016*, ser. LIPIcs, vol. 57. Schloss Dagstuhl, 2016, pp. 66:1–66:15.

[21] P. Surynek, "Finding optimal solutions to token swapping by conflict-based search and reduction to SAT," in *Proceedings of ICTAI 2018*. IEEE, 2018, in press.

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

[23] 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.

[24] J. Yu and D. Rus, "Pebble motion on graphs with rotations: Efficient feasibility tests and planning algorithms," in *Algorithmic Foundations of Robotics XI - WAFR 2014*, 2014, pp. 729–746.

[25] W. Hönig, T. K. S. Kumar, L. Cohen, H. Ma, H. Xu, N. Ayanian, and S. Koenig, "Summary: Multi-agent path finding with kinematic constraints," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI 2017*, 2017, pp. 4869–4873.

[26] 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.

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

[28] N. R. Sturtevant, "Benchmarks for grid-based pathfinding," *Computational Intelligence and AI in Games*, vol. 4, no. 2, pp. 144–148, 2012.