Colored Multi-Agent Path Finding: Solving Approaches

Roman Barták, Marika Ivanová, Jiří Švancara
Charles University, Faculty of Mathematics and Physics
Prague, Czech Republic
{bartak,ivanova}@ktiml.mff.cuni.cz, jiri.svancara@mff.cuni.cz

Abstract

Multi-Agent Path Finding (MAPF) deals with the problem of finding collision-free paths for a set of agents moving in a shared environment while each agent has specified its destination. Colored MAPF generalizes MAPF by defining groups of agents that share a set of destination locations. In the paper, we evaluate several approaches to optimally solve the colored MAPF problem, namely, a method based on network flows, an extended version of conflict-based search, and two models using Boolean satisfiability. We also investigate methods for obtaining lower bounds on optimal solutions based on constraint and continuous relaxation techniques.

Introduction

Multi-Agent Path Finding (MAPF) is a rapidly developing area dealing with finding collision-free paths for a set of agents moving in a shared environment. In a classical formulation, each agent has its destination. However, there are applications where groups of agents need to move to specific areas, but the exact locations of agents in these areas are not important. Such generalization of MAPF is called a *Colored MAPF*. Colored MAPF problems can be found in computer games, where armies of bots are moving to locations specified by the player (Ma et al. 2017). Positions of individual bots are not distinguished, but the groups should reach their destinations. There are similar situations in transportation problems, for example, in warehouses (Ma and Koenig 2016). At the end of a working shift, the groups of robots need to move to locations with charging stations. Again, assigning the particular station to a particular robot is not essential, provided that robots reach locations with compatible charging stations. The final example of Colored MAPF is from the area of computer art using mobile robots and drones. Recently, it became popular to use sets of robots to display a picture and to morph it into another picture by moving the robots (Barták and Mestek 2021). Robots may glow different colors, which naturally define sub-groups of robots.

Opposite to classical MAPF, there was not much attention paid to solving Colored MAPF problems. There is only a single work (Ma and Koenig 2016) that proposed a solving technique for Colored MAPF. This technique, Conflict-Based Min-Cost-Flow (CBM), is based on a popular MAPF algorithm Conflict-Based Search (CBS) (Sharon et al. 2012). In this paper, we propose extensions of SAT-based models for MAPF to solve Colored MAPF. One method is a straightforward generalization of the SAT model for classical MAPF. The other method attempts to reduce symmetries that appear in the problem due to the anonymity of agents within groups. We will also formulate the problem as an integer linear program for the multi-commodity network flow problem. We empirically compare these solving approaches to identify the specific instance properties that determine which solving approach to select for a given problem instance.

Colored MAPF

Let $A = \{1, \ldots, n\}$ be a set of n agents and $G = (V, E)$ be an un-directed graph. Agents are initially staying at some vertices, which is described by initial configuration $S : A \rightarrow V$, where $S(a)$ is the initial position of agent a. The final configuration is given by a set of vertices $T \subset V$ such that $|T| = |A|$. *Anonymous MAPF* problem is given by a quadruple (G, A, S, T) and its solution is a set of collision-free plans. A plan π_a for an agent a is a sequence of vertices such that $\pi_a[1] = S(a)$ and for each t either $\pi_a[t] = \pi_a[t+1]$ (agent waits at a vertex) or $(\pi_a[t], \pi_a[t+1]) \in E$ (agent moves to a neighboring vertex). Let m_a be the length of plan for agent a, then we define $\pi_a[t] = \pi_a[m_a]$ for each $t > m_a$ (agents stay in their final vertices). Let $mks = max_{a \in A}\, m_a$ be a makespan of the plans. We require agents to reach the final configuration: $\forall v \in T\ \exists a \in A : \pi_a[mks] = v$, and the plans to be collision free: $\forall a_1, a_2 \in A, a_1 \neq a_2, \forall t : \pi_{a_1}[t] \neq \pi_{a_2}[t]$ (no vertex collision) and $\forall a_1, a_2 \in A, a_1 \neq a_2, \forall t : \pi_{a_1}[t] \neq \pi_{a_2}[t+1] \vee \pi_{a_1}[t+1] \neq \pi_{a_2}[t]$ (no swapping collision).

Colored MAPF (also called Team MAPF or TAPF) with k groups is then given as $(G, (A_1, S_1, T_1), \ldots, (A_k, S_k, T_k))$, where each (G, A_i, S_i, T_i) is anonymous MAPF (Solovey and Halperin 2014), see Figure 1. A solution of Colored MAPF is union of solutions of individual anonymous MAPF problems such that the plans across the groups are also collision-free.

Anonymous MAPF can be solved makespan-optimally

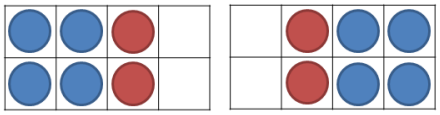

Figure 1: An instance of Colored MAPF. Initial (left) and goal (right) positions of two groups of agents.

in polynomial time (Yu and LaValle 2012), but finding a makespan-optimal solution to Colored MAPF with at least 2 groups is NP-hard (Surynek 2010) (as Colored MAPF is a generalization of classical MAPF, where each agent has its destination). There exists an optimal algorithm for solving Colored MAPF, called Conflict-Based Min-Cost-Flow (CBM) (Ma and Koenig 2016) which is based on a search algorithm Conflict-Based Search (CBS) (Sharon et al. 2012). Briefly speaking, on the high level, CBM performs a classical CBS search over conflicts among the groups; on the low level, each team of agents is navigated by a polynomial-time algorithm that is compliant with the restrictions of the CBS search. If there is a collision between two groups, new restrictions are applied, and the search continues until a collision-free solution is found.

Solution Methods

In addition to the existing CBM algorithm, we propose two reduction-based techniques for determining a makespan-optimal solution for a given problem instance utilizing available solvers. Reduction-based techniques use layered graphs for a given plan length, where each layer describes nodes reachable by agents at a given time step. Let $R_t(i,c)$, $i = 1,\ldots,t$, $c = 1,\ldots,k$ consist of vertices $v \in V$ that lie on a path p from $S(a)$ for some $a \in A_c$ to some vertex in T_c of length at most t, the distance from $S(a)$ to v along p is at most i, and the distance from v to some vertex in T_c along p is at most $t-i$. Vertices in $R_t(i,c)$ are relevant in time step i, that is, for each vertex $v \in R_t(i,c)$ there is a possibility that it will be used by some agent as $\pi_a[i] = v$ for some $a \in A_c$. Vertices that are not in $R_t(i,c)$ cannot be reached in time step i, and therefore cannot be part of the solution.

Both methods have a common pre-processing phase that involves the construction of $R_t(i,c)$, which aims to reduce the number of variables entering a solver. Furthermore, if $T_c \neq R_t(t,c)$, for some group c, the problem is infeasible, and the deadline t has to be increased. The first t for which $T_c = R_t(t,c)$ for each $c \in \{1\ldots,k\}$ gives a straightforward lower bound, which allows to skip several initial executions of a solver.

SAT Models

To solve Colored MAPF via reduction to a Boolean satisfiability problem (SAT), we define the following two sets of variables: $\forall v \in V, \forall a \in \cup_{c=1}^{k} A_c, \forall i \in \{0,\ldots,t\}$: $At(v,a,i)$ meaning that agent a is in vertex v at time step i and $\forall (v,u) \in E, \forall a \in \cup_{c=1}^{k} A_c, \forall i \in \{0,\ldots,t-1\}$: $Pass(v,u,a,i)$ meaning that agent a goes through an edge (v,u) at time step i. Now, we introduce the following constraints:

$$\forall a \in \cup_{c=1}^{k} A_c : At(S(a),a,0) = 1 \quad (1a)$$

$$\forall c \in \{1,\ldots k\}, \forall a \in A_c : \sum_{v \in T_c} At(v,a,t) = 1 \quad (1b)$$

$$\forall v \in V, \forall i \in \{0,\ldots,t\} : \sum_{a \in \cup_{c=1}^{k} A_c} At(v,a,i) \leq 1 \quad (1c)$$

$$\forall (v,u) \in E, \forall i \in \{0,\ldots,t-1\} : \sum_{a \in \cup_{c=1}^{k} A_c} Pass(v,u,a,i) + Pass(u,v,a,i) \leq 1 \quad (1d)$$

$$\forall v \in V, \forall a \in \cup_{c=1}^{k} A_c, \forall i \in \{0,\ldots,t-1\} : At(v,a,i) = \sum_{(v,u) \in E} Pass(v,u,a,t) \quad (1e)$$

$$\forall v \in V, \forall a \in \cup_{c=1}^{k} A_c, \forall i \in \{1,\ldots,t\} : At(v,a,i) = \sum_{(u,v) \in E} Pass(u,v,a,i-1) \quad (1f)$$

The constraints (1a) and (1b) ensure that the starting and goal positions of all agents are valid. Since all of the agents from a group c may arrive at any goal vertex $v \in T_c$, we require each agent $a \in A_c$ to reach exactly one of those vertices. The constraint (1c) ensures that each vertex is occupied by at most one agent to prevent vertex collision, while constraint (1d) does the same for all pairs of opposite edges, thus forbidding swapping collision. The correct movement in the graph is ensured by constraints (1e) and (1f). They ensure that an agent is in a node, if and only if it leaves by one of the outgoing edges (1e). Similarly, an agent is in a node, if and only if it entered by one of the incoming edges in the previous time step (1f). We assume that edges (v,v) exist for each vertex v, allowing agents to wait at a vertex.

To find the optimal makespan, we iteratively increase the upper bound on makespan t until a satisfiable formula is generated. The pre-processing mentioned at the beginning of the section is applicable. We construct the variables only for the vertices that are in $R_t(i,c)$. That is, if a vertex v can not be reached in timestep i, we do not create variable $At(v,a,i)$. Similarly, if an agent can not be present at a vertex at a given time, it can not traverse the incident edges at that time.

This model is a direct alteration of the SAT-based model used to solve classical MAPF problem (Surynek 2014; Stern et al. 2019). The only difference is in the constraint that specifies the agents' goal locations. Therefore, we shall refer to this model as *SAT-basic*.

Another approach to modeling the Colored MAPF problem is to realize that all of the agents in a group are interchangeable. When two agents from a single group swap their positions, it is equivalent to both of them waiting in their current locations. Therefore, we do not need to define the variables $At(v,a,t)$ and $Pass(v,u,a,t)$ for each agent a but rather for each group c ($At(v,c,t)$ and $Pass(v,u,c,t)$). The constraints dealing with starting and goal positions then change as follows:

$$\forall c \in \{1,\ldots k\}, \forall v \in \{S(a) : a \in A_c\} : At(v,c,0) = 1 \quad (2a)$$

$$\forall c \in \{1,\ldots k\}, \forall v \in T_c : At(v,c,t) = 1 \quad (2b)$$

The other constraints remain the same, except that instead of iterating over all agents, we iterate over all groups. This approach

saves many variables entering the SAT solver and, therefore, may prove to be more efficient. We shall refer to this model as *SAT-grouped*.

It is possible to model the allowed movement of the agents over the defined variables in many ways. We tried several other encodings; for example, constraints (1e) and (1f) can be replaced by conditions stating that if an agent is in a vertex, it must leave by precisely one outgoing edge, if an agent is traversing an edge, it must enter the appropriate vertex in the next time step and that each agent occupies exactly one vertex at a time ($|A_c|$ vertices in case of *SAT-grouped*). It was determined in preliminary experiments that the models *SAT-basic* and *SAT-grouped* are the most promising and will be used in the final experiments.

All of our SAT-based models are modeled in Picat programming language (Zhou and Kjellerstrand 2016), which provides an automatic translation of arithmetic constraints (such as constraints (1b) – (1f)) into a CNF formula that is solvable by an off-the-shelf SAT solver. We present the constraints in the arithmetic form as we believe it is more comprehensible.

ILP Model

Colored MAPF has been shown to be equivalent to the multi-commodity network flow problem (MCF) in (Ma and Koenig 2016), where the authors use a directed layered graph constructed in a way that allows formulating the problem using standard MCF constraints. We utilize a more straightforward layered graph that requires several additional constraints because this model's solution time is advantageous according to our preliminary experiments.

Like in the SAT-based models, the makespan optimal solution is obtained by an iterative execution of an ILP solver over the model with an increasing deadline until the problem becomes feasible. The agent groups in Colored MAPF represent different commodities flowing through the extended spatial-temporal layered graph with unit edge capacities.

Given a deadline t, we consider k directed layered graphs $G_t^c = (\cup_{i=0}^{t} V_i^c, \cup_{i=1}^{t} E_i^c)$, $c = 1, \ldots, k$, with $V_i^c = \{v_i : v \in V \cap R_t(i, c), i = 0, \ldots, t\}$, and E_i^c connecting nodes in consecutive layers $i-1$ and i. The set E_i^c consists of arcs (u_{i-1}, u_i) corresponding to waiting at $u \in V$ in time step i, and arcs (u_{i-1}, v_i), representing a transition from node u to an adjacent node v in the original graph G. The ILP formulation of Colored MAPF is built upon variables $f_{uv}^{ic} \in \{0,1\}^{\sum_{c\in\{1,\ldots,k\}} |\cup_{i=1}^{t} E_i^c|}$ indicating whether or not a flow of commodity c passes through $(u, v) \in E_j^c$. The ILP formulation of Colored MAPF is in the form

$$\max \sum_{c=1}^{k} \sum_{(u,v)\in E_t^c} f_{uv}^{tc} \qquad \text{s. t.} \tag{3a}$$

$$\forall c \in \{1, \ldots, k\}, \forall u \in V_0^c : \sum_{(u,v)\in E_1^c} f_{uv}^{1c} = 1 \tag{3b}$$

$$\forall c \in \{1, \ldots, k\}, \forall i \in \{1, \ldots, t-1\}, \forall v \in V_i^c : \sum_{(u,v)\in E_i^c} f_{uv}^{ic} = \sum_{(v,w)\in E_{i+1}^c} f_{vw}^{i+1,c} \tag{3c}$$

$$\forall c \in \{1, \ldots, k\}, \forall v \in V_t^c : \sum_{(u,v)\in E_t^c} f_{uv}^{tc} = 1 \tag{3d}$$

$$\forall i \in \{1, \ldots, t-1\}, \forall v \in \cup_{c=1}^{k} V_i^c : \sum_{c=1}^{k} \sum_{(u,v)\in E_i^c} f_{uv}^{tc} \leq 1 \tag{3e}$$

$$\forall c, d \in \{1, \ldots, k\}, c \neq d, \forall i \in \{1, \ldots, t\}, \forall (u_{i-1}\, v_i) \in E_i^c, (v_{i-1}\, u_i) \in E_i^d : f_{u_{i-1}\, v_i}^{ic} + f_{v_{i-1}\, u_i}^{id} \leq 1 \tag{3f}$$

$$\forall c \in \{1, \ldots, k\}, \forall i \in \{1, \ldots, t\}, \forall (u, v) \in \cup_{i=1}^{t} E_i^c : f_{uv}^{ic} \in \{0, 1\} \tag{3g}$$

Equations (3b)-(3d) model the flow conservation constraints applied on the extended graph implying that each agent $a \in A_c$ moves from $S(a)$ gradually through layers of G_t^c until it reaches exactly one node in T_c. By (3e) we impose unit node capacities that ensure that agents do not collide at nodes and imply edge capacities. Finally, (3f) prevents two agents from different groups from exchanging their positions. Note that two agents from the same group may swap, as they are anonymous within the group. The objective function (3a) maximizes the number of agents that reach a relevant target node. The objective function is superfluous because the agent groups' sizes give the flow sizes of individual commodities. The objective function value is thus known beforehand. The iterative nature of the algorithm accomplishes the optimality. However, empirical comparison of the model's performances with and without the objective function indicates that the presence of (3a) in the model is beneficial.

The ILP model is implemented in AMPL (Fourer, Gay, and Kernighan 2002) utilizing the CPLEX solver with default settings except for the option `lowercutoff` set to $|\cup_{c=1}^{k} T_c|$ which tells to the solver to skip any branch whose continuous relaxation's optimal value is less than `lowercutoff`.

Lower bounds

The importance of lower bounding methods for Colored MAPF is twofold. Firstly, knowing a lower bound for makespan allows skipping several initial executions of a solver. Secondly, when studying heuristic methods, the optimal solution is typically unknown, and sufficiently tight lower bounds help assess a heuristic algorithm's quality. In the following, we describe four strategies for determining lower bounds pursued in this paper. The function $\ell(s, t)$ returns the length of a shortest path between s and t in G.

1) Simple strategy. A straightforward lower bound for classic MAPF is obtained by disregarding interactions between the agents, that is, allowing both edge and vertex collisions. The lower bound is then given by the maximum of the shortest paths between the agents' initial and target positions. In Colored MAPF, this approach is generalized by taking the maximum of the shortest possible path lengths between nodes in S_c and T_c for each team c. For an instance Γ of Colored MAPF, we have ,

$$\lambda_{\text{sim}}(\Gamma) = \max_{c\in\{1,\ldots,k\}} \max_{s\in S_c} \min_{t\in T_c} \ell(s, t). \tag{4}$$

2) Degree strategy. A drawback of the simple strategy is that (4) does not take into account that the agents have to be distributed in all of the targets, and so the lower bound can be based on two agents aiming for the same target. The lower bound can therefore be strengthened by imposing that each agent is sent to a different target. For each team c, let us consider a complete weighted bipartite graph B_c with partitions corresponding to S_c and T_c, in which edge costs are defined by ℓ. We then iteratively remove edges ordered decreasingly by the cost as long as none of the nodes in B_c is isolated. This condition ensures that each vertex is reachable since they have to be used in the solution. The cost of the last removed edge is the desired lower bound λ_{deg}. Alg. 1 formalizes this greedy process. The condition *conditionHolds* (line 8) is satisfied if there exists an isolated node in B.

Algorithm 1: Finding a lower bound

Data: An instance $(G, (A_1, S_1, T_1), \dots (A_k, S_k, T_k))$

```
maxLB← 0;
for c = 1, ..., k do
    (S_c, T_c, E_c, ℓ) ← complete bipartite graph with
      partitions S_c and B_c, edges E_c and edge costs
      defined by ℓ(s,t) for each s ∈ S_c and t ∈ T_c;
    L ← sort E_c by decreasing cost ;
    for (s,t) ∈ L do
        lb← ℓ(s,t);
        E_c ← E_c \ (s,t);
        If conditionHolds() break;
    if lb>maxLB then
        maxLB← lb;
return maxLB;
```

3) Matching strategy. The lower bound obtained by Degree strategy can be improved by strengthening the condition *conditionHolds* in Alg. 1 by checking whether E_c still contains a complete matching in B_c as a subset. This condition can be violated earlier than the degree test, and the last removed edge can therefore have a higher cost and yield a tighter lower bound λ_{match}. The reasoning behind the correctness is similar to the Degree strategy – all of the agents have to reach some goal; therefore, each start has to be matched to some goal. Removal of the longest edges (which means ignoring the longest path) provides a lower bound for makespan as opposed to finding a matching with the smallest cost, which would provide a lower bound for the sum of costs (another cost function often used in classical MAPF (Stern et al. 2019)).

4) LP strategy. A typical approach for determining lower bounds for an ILP model is to compute its continuous relaxation. By replacing $f_{uv}^{ic} \in \{0, 1\}$ with $f_{uv}^{ic} \in [0, 1]$ in the integrality constraints (3g) we obtain a linear program that can be solved by, e.g., the simplex method in polynomial time. Another valuable property is that an integral optimal solution to the continuous relaxation is an optimal solution to the original ILP.

Observation 1. *Let Γ be an instance of Colored MAPF. Then,*

$$\lambda_{sim}(\Gamma) \leq_1 \lambda_{deg}(\Gamma) \leq_2 \lambda_{match}(\Gamma) \leq_3 \lambda_{LP}(\Gamma).$$

Proof. $\leq_1$: Let $\ell(s, t)$ be the value returned by Simple strategy. If the Degree strategy removes all edges with length $\ell(s, t)$ (and greater) from the bipartite graph, there has to be a vertex with 0 degree, since $\ell(s, t)$ is the minimal value of all outgoing edges for one of the vertices.

$\leq_2$: Once an edge is removed such that there is a vertex with degree 0, there is no matching in the bipartite graph.

$\leq_3$: LP relaxation finds flows from starts to destinations. From these paths we can reconstruct the matching in B_c. □

An example where Degree and Matching strategies outperform Simple strategy can be seen in Figure 2.

Experimental Evaluation

Initially, we investigate the tightness of the lower bounding strategies. Consequently, we study the proposed models' performance utilizing the best lower bounds and compare them with the state-of-the-art CBM algorithm introduced in (Ma and Koenig 2016).

Figure 2: Two agents from the same group with two goals. Simple strategy computes lower bound of 3, as both agents are navigated to the closer goal. Both Degree and Matching strategies provide lower bound of 4, as they correctly compute that one of the agents has to go to the farther goal.

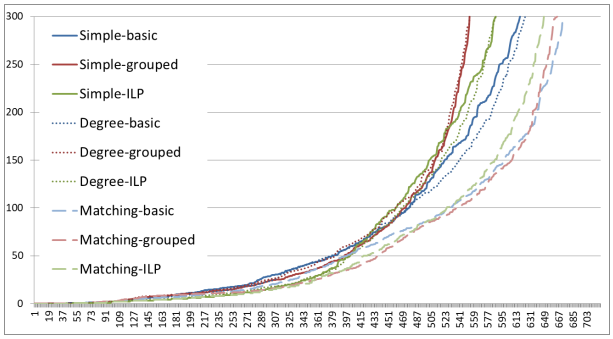

Figure 3: The impact of different lower bounding methods on runtime for each proposed model. The graph shows number of instances (x-axis) solved in a given time limit (y-axis).

Our experimental scenarios are inspired by existing benchmarks for MAPF (Stern et al. 2019). The experiments are performed on 4-connected grid maps *empty* (no obstacles are present in the graph) and *random* (20% of vertices chosen at random are marked as an impassable obstacle) of sizes 8×8, 16×16 and 32×32, with the number of groups $k = 5$ and $k = 10$. In every instance, the groups' size is consistent $|A_1| = \dots = |A_k|$, and the total number of agents increases from k to 100 (40 for map sizes 8×8) with the increment of $k - |A| = k, 2k, \dots, 100$. A time limit of 5 minutes is imposed on the runtime of each instance.

Lower bounding methods

For each tested instance, we determine the lower bounds yielded by the four strategies. Simple, Degree, and Matching strategy can be computed in virtually no time on the pursued instances. The LP strategy, however seemingly strong, takes more time, particularly in large instances. There are also cases when even the linear relaxation cannot be calculated to optimality, and only a weaker lower bound is obtained.

The experimental results summarized in Table 1 provide a closer insight into individual strategies' strength. The Matching strategy always reaches the optimal makespan in 3 out of 6 instance classes. It tends to be more successful in less crowded instances, which is following the intuition. The LP lower bound is as strong as the Matching lower bound in instances in which the LP relaxation is solved to optimality within the given time limit. Particularly in some of the large instances (map size 32×32), solving the LP relaxation is interrupted before the optimum is found. Thus, the bounds are weaker than those obtained by Matching. Therefore, the Matching strategy is the most practical approach and a clear winner of this comparison, despite, theoretically, the LP strategy provides no worse lower bounds, but with the trade-off of larger computational complexity. Figure 3 shows the effect of better lower bounds

		λ_{sim}	λ_{deg}	λ_{match}	λ_{LP}
Empty	8x8	92.33	96.70	**99.77**	**99.77**
	16x16	85.64	93.72	**100.00**	**100.00**
	32x32	91.00	95.11	**100.00**	99.06
Random	8x8	77.83	86.15	89.25	**95.41**
	16x16	85.91	93.54	99.84	**99.96**
	32x32	91.77	94.14	**100.00**	98.74

Table 1: Average proportion to optimal makespan for different instance types obtained by individual lower bounding strategies. Instances with unknown optimum are excluded.

on the efficiency of studied methods. There is clear evidence that the Matching strategy brings the most significant efficiency improvement. The Matching strategy's lower bounds are used as a starting point in the experimental evaluation of individual models in the next section.

Exact algorithms

The growth of runtime of each algorithm in all tested instances is depicted in Fig. 4a-4c. Both ILP and SAT models fall behind CBM on the largest maps in virtually all instances (Fig 4c). In the medium-sized maps (Fig 4b), the difference is much less noticeable, particularly in the map random 16×16. On the other hand, in the smallest maps, CBM is outperformed by all other methods as indicated in Fig 4a).

Next, Fig. 4d-4f ordered by increasing map size show the growth of runtime of individual algorithms with increasing number of agents. We observe that some of the curves are not very smooth, suggesting that there are factors other than the number of agents that influence the runtime in a given map. We can observe that adding more agents to the instance makes it easier to solve in some cases. This observation is counterintuitive since, in classical MAPF, adding more agents makes the problem harder. In Colored MAPF, additional agents are assigned to some group, and with them, new goal locations are added to that group. These goal locations may be used by other agents as well, which may yield a shorter and easier to compute plan.

The ILP model performs well on instances with a small number of agents regardless of map size, which is the reason why this method often has a high number of fastest computations (Tab 2). However, with the increasing number of agents, this margin is quickly eliminated and surpassed.

The SAT-based models perform well on smaller instances with the highest success ratio on the 8×8 and 16×16 maps. As for the largest maps, the success ratio and performance fall behind for both of the SAT-based models. We can see that on the largest maps, *SAT-basic* outperforms *SAT-grouped*. This observation is surprising, considering that *SAT-grouped* uses on average only half as many variables as *SAT-basic* across all sizes of maps. Comparing the two models based on the number of teams shows that *SAT-grouped* performs better when the agents are divided into fewer teams; therefore, each team's size is larger.

The CBM algorithm performs the best on the 32×32 maps with both the highest success ratio and the best performance. On the other hand, the algorithm performs poorly on the smallest maps, especially on the map random 8×8, where it was able to solve only half of the instances (it was more successful on empty 8×8, which increases the ratio in Table 2).

These results comply with the observation made for the classical MAPF problem that the CBS-based algorithm performs well on

		solved k = 5	solved k=10	fastest k = 5	fastest k=10
8×8	Sat-b	**100.00**	**97.50**	12.50	17.50
	Sat-g	**100.00**	**97.50**	31.25	**47.50**
	ILP	95.00	82.50	**50.00**	25.00
	CBM	77.50	70.00	6.25	7.50
16×16	Sat-b	**100.00**	**100.00**	9.50	**39.00**
	Sat-g	**100.00**	**100.00**	28.50	11.00
	ILP	**100.00**	88.00	**58.50**	31.00
	CBM	98.00	99.00	3.50	19.00
32×32	Sat-b	85.50	82.00	6.00	18.00
	Sat-g	94.00	57.00	5.50	1.00
	ILP	95.50	59.00	39.00	9.00
	CBM	**99.00**	**97.00**	**49.00**	**69.00**

Table 2: Proportion of instances solved to optimality within a given time limit and instances solved in the shortest runtime by individual methods.

sparse instances with fewer interactions between agents (or, in this case, a group of agents), and reduction-based algorithms perform well on smaller, more dense instances (Svancara and Barták 2019).

Summary and Future Research

We present several reduction models that solve Colored MAPF. First, we study an ILP model based on the multi-commodity network flow problem. Another approach is a reduction to Boolean satisfiability (SAT). We pursue two models – a basic model, which is a direct alteration of the SAT-based model for classical MAPF, and a grouped model that exploits the fact that agents in a single group are interchangeable. The number of variables entering the SAT solver in the grouped model is significantly lower than for the basic model.

We compared these models with the known algorithm CBM on 4-connected grid maps with various sizes and numbers of obstacles. We can observe some trends in the effectiveness of the solvers. The ILP approach performs well on instances with a small number of agents on maps of any size, while the CBM algorithm performs well on the largest sparse maps where the groups of agents do not interact much. On the other hand, small maps with many agents are solved most successfully by the SAT-based models.

Unlike classic MAPF, where extending an instance by a new agent–target pair in an existing team either increases or does not change the minimum makespan, Colored MAPF has the property that such an extension may decrease the minimum makespan of the instance. We observe this behavior on multiple occasions in the experiments. The explanation lies in the fact that a newly added target may be placed in a favorable position, closer to an agent that previously had to pass via a path of length equal to the minimum makespan. If this agent aims for the new target, the new instance's minimum makespan may be shorter than in the original one. A further investigation of circumstances in which adding a new target leads to a decrease of the minimum makespan is a relevant area of future research and is also essential in better understanding Colored MAPF.

It might be interesting to investigate more types of maps to gain a closer insight into which aspects besides the graph size and agent count influence individual methods' runtime.

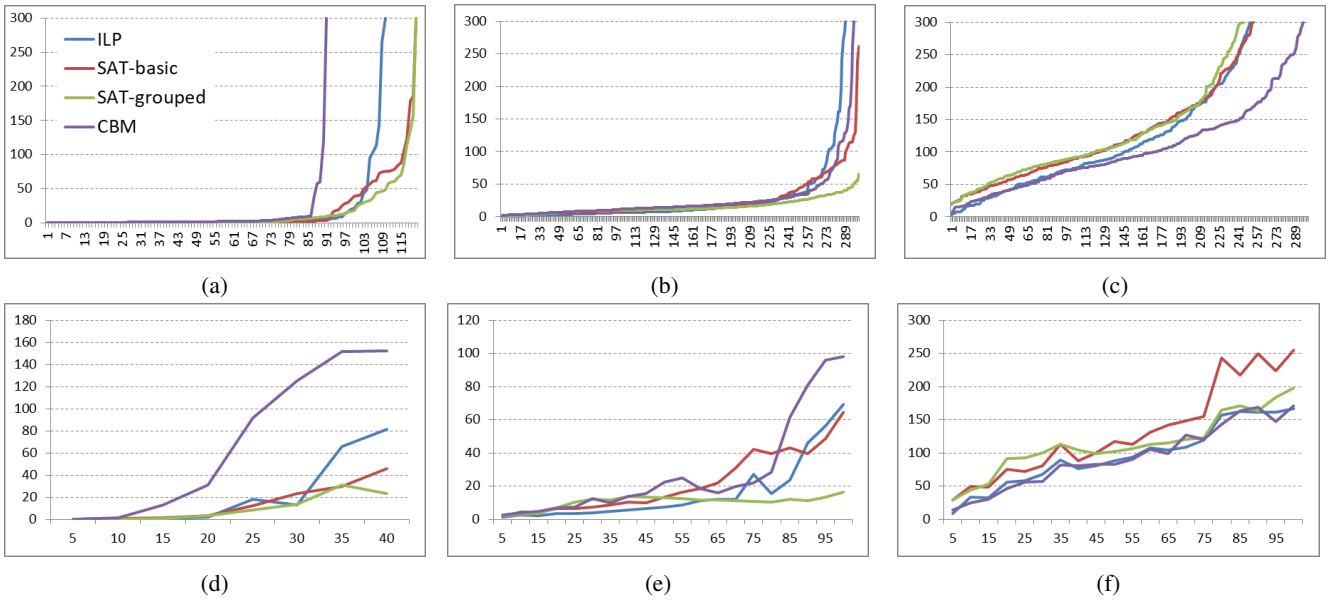

Figure 4: Figures 4a – 4c show number of instances (x-axis) solved in a given time limit (y-axis) by each of the studied solvers. In order they show the results for maps of size 8×8, 16×16 and 32×32. Figures 4d – 4f show the average runtime for instances with a given number of agents. In order they show the results for maps of size 8×8, 16×16 and 32×32.

Our methods for finding lower bounds often yield a value equal to the actual makespan in many tested instances. Some types of real-life instances often involve much larger maps with thousands of agents. It is, therefore, worthwhile to develop algorithms without the optimality guarantee. Experimental results in this work suggest that these lower bounding techniques help assess the quality of inexact methods.

Acknowledgement

This research is supported by the the Czech-Israeli Cooperative Scientific Research Project LTAIZ19014 and by the project 19-02183S of the Czech Science Foundation.

References

Barták, R., and Mestek, J. 2021. Ozomorph: Demonstrating colored multi-agent path finding on real robots. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*. AAAI Press.

Fourer, R.; Gay, D.; and Kernighan, B. 2002. Ampl: A modeling language for mathematical programming. *Management Science - MANAGE SCI* 36.

Ma, H., and Koenig, S. 2016. Optimal target assignment and path finding for teams of agents. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, May 9-13, 2016*, 1144–1152.

Ma, H.; Yang, J.; Cohen, L.; Kumar, T. K. S.; and Koenig, S. 2017. Feasibility study: Moving non-homogeneous teams in congested video game environments. In Magerko, B., and Rowe, J. P., eds., *Proceedings of the Thirteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-17), October 5-9, 2017, Snowbird, Little Cottonwood Canyon, Utah, USA*, 270–272. AAAI Press.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2012. Conflict-based search for optimal multi-agent path finding. In Hoffmann, J., and Selman, B., eds., *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*. AAAI Press.

Solovey, K., and Halperin, D. 2014. k-color multi-robot motion planning. *I. J. Robotics Res.* 33(1):82–97.

Stern, R.; Sturtevant, N. R.; Felner, A.; Koenig, S.; Ma, H.; Walker, T. T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T. K. S.; Barták, R.; and Boyarski, E. 2019. Multi-agent pathfinding: Definitions, variants, and benchmarks. In Surynek, P., and Yeoh, W., eds., *Proceedings of the Twelfth International Symposium on Combinatorial Search, SOCS 2019, Napa, California, 16-17 July 2019*, 151–159. AAAI Press.

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*.

Surynek, P. 2014. Compact representations of cooperative path-finding as SAT based on matchings in bipartite graphs. In *Proceedings of the 26th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 875–882. IEEE Computer Society.

Svancara, J., and Barták, R. 2019. Combining strengths of optimal multi-agent path finding algorithms. In Rocha, A. P.; Steels, L.; and van den Herik, H. J., eds., *Proceedings of the 11th International Conference on Agents and Artificial Intelligence, ICAART 2019, Volume 1, Prague, Czech Republic, February 19-21, 2019*, 226–231. SciTePress.

Yu, J., and LaValle, S. M. 2012. Multi-agent path planning and network flow. In *Algorithmic Foundations of Robotics X - Proceedings of the Tenth Workshop on the Algorithmic Foundations of Robotics, WAFR 2012, 2012*, 157–173.

Zhou, N., and Kjellerstrand, H. 2016. The picat-sat compiler. In *Practical Aspects of Declarative Languages - 18th International Symposium, PADL 2016, St. Petersburg, FL, USA, January 18-19, 2016. Proceedings*, 48–62.