Formation Change for Robot Groups in Occluded Environments

Wolfgang Hönig, T. K. Satish Kumar, Hang Ma, Sven Koenig, Nora Ayanian

***Abstract*— We study formation change for robot groups in known environments. We are given a team of robots partitioned into groups, where robots in the same group are interchangeable with each other. A formation specifies the locations occupied by each group. The objective is to find collision-free paths that move all robots from a given start formation to a given goal formation. Our algorithm TAPF* has the following features: (a) it incorporates kinematic constraints of robots in form of velocity limits; (b) it maintains a user-specified safety distance between robots; (c) it attempts to minimize the makespan; and (d) it runs efficiently for hundreds of robots and dozens of groups even in dense 3D environments with narrow corridors and other occlusions. We demonstrate the efficiency and effectiveness of TAPF* in simulation and on robots.**

I. INTRODUCTION

We study the formation-change problem for robot groups in known environments. We are given a team of robots partitioned into groups, where robots in the same group are interchangeable with each other. A formation specifies the locations occupied by each group without regard to which robot in a group occupies which location meant for the group. In marching bands, for example, all flute players are interchangeable with each other in the sense that it does not matter which location of the goal formation one of the flute players occupies as long as it is meant for a flute player. The objective of the formation-change problem is to find collision-free paths that move all robots from a given start formation to a given goal formation with the minimal makespan, see Figure 1 for an example. Robot applications include search-and-rescue, robotic convoys, and reconnaissance.

The multi-agent path-finding problem (MAPF) is a specialization of the formation-change problem where all groups have cardinality one (that is, no robots are interchangeable with each other and thus every robot is assigned a specific location in the goal formation). Solving a formation-change problem thus consists of two parts, namely assigning each robot a unique location in the goal formation meant for its group (called target assignment) and multi-agent path finding for the resulting target assignments.

The formation-change problem with one group can be solved in polynomial time with maxflow algorithms [1]. However, solving the formation-change problem with more than one group is NP-hard [2]. Furthermore, robots might not stay in formation during a formation change in occluded environments even if the start and goal formations are identical except for translation because the robots need to split up in order to move around obstacles.

All authors are with the Computer Science Department at the University of Southern California: whoenig@usc.edu, tkskwork@gmail.com, and {hangma,skoenig,ayanian}@usc.edu.

Their research was supported by ARL under grant number W911NF-14-D-0005, ONR under grant numbers N00014-14-1-0734 and N00014-09-1-1031, NASA via Stinger Ghaffarian Technologies, and NSF under grant numbers 1409987 and 1319966.

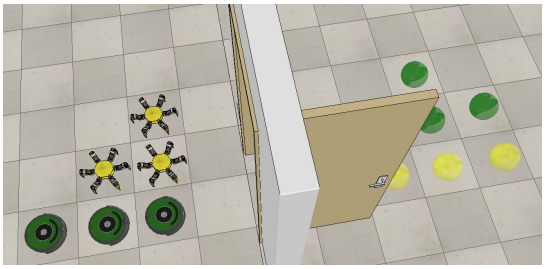
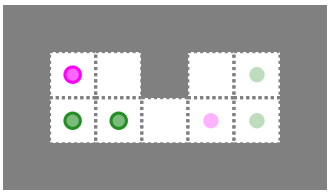


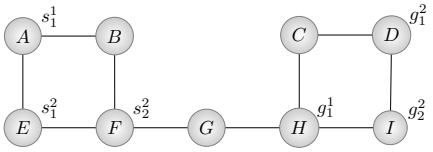
Fig. 1. Formation-change instance with two groups, differentiated by color. The start formation is on the left, and the goal formation is on the right. The open door is a narrow passageway. The walls are obstacles.

We solve the formation-change problem by introducing TAPF*, that combines two of our recent methods sequentially that both minimize the makespan: The first method is the conflict-based min-cost-flow algorithm (CBM) [3], a path-planning method that generalizes the conflict-based search algorithm [4] from solving MAPF to solving the target-allocation and path-finding problem (TAPF), which is a version of the formation-change problem that ignores the physical properties of robots (such as their sizes and kinematic constraints) to search efficiently. CBM uses a two-level search strategy. The low-level search uses a polynomial-time maxflow algorithm for target assignment and multi-agent path finding for all robots from a given group, while the high-level search resolves collisions between robots from different groups. CBM is experimentally up to thirty times faster than a method based on integer linear programming. The second method is MAPF-POST [5], an execution-monitoring method that takes the physical properties of robots into account when using linear programming for a simple temporal network to determine appropriate velocities that allow them to follow the collision-free paths determined by CBM.

We do not only combine these methods but also provide several improvements to MAPF-POST: The user can now specify the desired safety distance between robots directly (the distance of two robots in the graph is at least δ and, for four-neighbor square grids and six-neighbor cubic grids, their Euclidean distance is at least $\delta/\sqrt{2}$ for a user-specified parameter $\delta > 0$), and MAPF-POST is more efficient since it no longer uses linear programming. Overall, TAPF* has the following features: (a) it incorporates kinematic constraints of robots in form of velocity limits; (b) it maintains a user-specified safety distance between robots to take their sizes into account; (c) it attempts to minimize the makespan; and



(a) TAPF instance with two groups, differentiated by color. The start formation is on the left, and the goal formation is on the right.



(b) Graphical representation of the TAPF instance. The single pink robot in group 1 (a_1^1) has start and goal locations s_1^1 and g_1^1, respectively.

Robot	$t=1$	$t=2$	$t=3$	$t=4$
1 (a_1^1)	$A \to B$	$B \to F$	$F \to G$	$G \to H$
2 (a_1^2)	$E \to F$	$F \to G$	$G \to H$	$H \to I$
3 (a_2^2)	$F \to G$	$G \to H$	$H \to C$	$C \to D$

(c) An optimal solution of the TAPF instance.

Fig. 2. Running TAPF instance and an optimal solution.

(d) it runs efficiently for hundreds of robots and dozens of groups even in dense 3D environments with narrow corridors and other occlusions. We demonstrate the efficiency and effectiveness of TAPF* in 2D and 3D simulation and on ground robots.

II. EFFICIENT PLANNING

For an instance of the target-assignment and path-finding problem (TAPF) [3], we are given an undirected graph $G = (V, E)$, where V is the set of vertices (corresponding to locations) and E is the set of undirected edges, and K robot groups $\{\text{group}_1, \text{group}_2 \dots \text{group}_K\}$, where group_i consists of K_i robots $\{a_1^i, a_2^i \dots a_{K_i}^i\}$ that are interchangeable with each other – for a total of $N = \sum_{i=1}^{K} K_i$ robots. Each robot a_j^i has a unique start location $s_j^i \in V$, and each group group_i has a set of unique goal locations $g^i = \{g_1^i, g_2^i \dots g_{K_i}^i\}$ (with these sets being pairwise disjoint). A solution to a TAPF instance consists of a target assignment that assigns each robot a unique goal location meant for the same group (given by K one-to-one mappings from robots in a group to targets in the same group, one for each group) and a path (potentially with sequences of the same location, corresponding to wait actions) from its start location to its goal location that avoids collisions with other robots. A solution is optimal iff it minimizes the makespan (that is, the time when the last robot reaches its goal location).

We use the conflict-based min-cost-flow algorithm (CBM) [3] to solve TAPF instances. CBM, like most state-of-the-art MAPF or TAPF solvers, assumes unit-length edges and discrete synchronized robot movements for holonomic robots, where each robot either traverses one edge per timestep or waits at its current location. Two robots collide if they are at the same location at the same timestep or traverse the same edge in opposite directions at the same timestep.

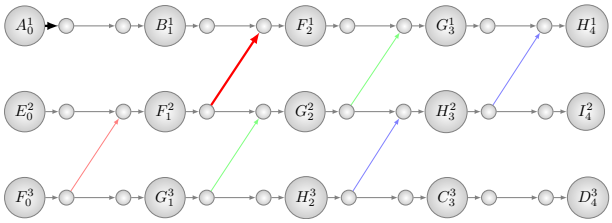


Fig. 3. TPG for the running TAPF instance and the optimal solution from Figure 2. We assume that all edges in G have lengths one and use $\delta = 0.25$. Each vertex labeled l_i^j in the TPG represents the event "robot j arrives at main location l at timestep i." Each edge represents a temporal precedence between the events represented by its incident vertices. The colors of the edges are used in Figure 4.

Figure 2 shows a TAPF instance (with one group of a single robot and one group of two robots) and an optimal solution.

III. EFFICIENT EXECUTION MONITORING

Formulating a formation-change instance as a TAPF instance discretizes the possible arrival times of the robots at locations into integer-valued timesteps, and a solution to a TAPF instance then provides a complete ordering of the arrival times. However, an execution-monitoring method needs leeway to adjust the arrival times to incorporate the kinematic constraints of the robots and maintain a safety distance between them, so should relax these restrictions.

We use MAPF-POST [5] as execution-monitoring method. MAPF-POST first introduces two auxiliary locations for each edge (u,v) in graph G to be able to guarantee a safety distance between robots, splitting the edge into a sequence of three microedges, namely a microedge of length δ from u to the first auxiliary location, a microedge from the first to the second auxiliary location of length $dist(u,v) - 2\delta$ (where $dist(u,v)$ is the length of edge $(u,v) \in E$), and a microedge of length δ from the second auxiliary location to v, for a user-specified parameter $\delta > 0$. Thus, auxiliary locations are at distance δ from the main locations.

MAPF-POST then converts a given TAPF solution to a temporal plan graph (TPG), which captures only the "essence" of the TAPF solution in form of critical temporal precedences among the arrival times - resulting in a partial ordering of real-valued arrival times. The TPG is a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices (corresponding to events of robots arriving at main or, for vertices that we refer to as safety markers, auxiliary locations) and $\mathcal{E}$ is the set of directed edges (corresponding to critical temporal precedences between events). The edges of the TPG correspond to two types of critical temporal precedences imposed by the given TAPF solution, namely the order in which the same robot has to arrive at two different locations (type 1) and the order in which two different robots have to arrive at the same location (type 2). Type 2 edges always connect two safety markers. For example, Figure 3 shows the TPG for the running TAPF instance and the optimal solution. The horizontal edges are type 1 edges and form chains that have two safety markers between the other vertices and correspond to robots traversing their paths in the

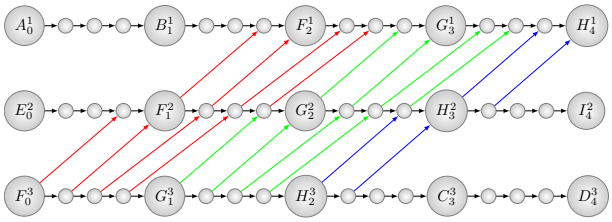


Fig. 4. Improved TPG for the running TAPF instance and the optimal solution from Figure 2. We assume that all edges in G have lengths one and use $\delta = 0.25$. The colored temporal precedence edges in red, green, and blue correspond to the temporal precedence edges of the same color in the TPG from Figure 3.

TAPF solution with sequences of the same location collapsed to the first one. The remaining edges are type 2 edges. In the optimal solution, robot 1 (= a_1^1) moves from A via two auxiliary locations to B, from B via two auxiliary locations to F, and so on, until it eventually arrives at H. Thus, it must arrive at the auxiliary location after A (toward B) after it arrives at A, which corresponds to the thick black type 1 edge in the figure. In the optimal solution, robot 2 (= a_1^2) arrives at F before robot 1. Thus, robot 2 must arrive at the auxiliary location after F (toward G) before robot 1 arrives at the auxiliary location before F (from B), which corresponds to the thick red type 2 edge in the figure. Details can be found in [5].

MAPF-POST then converts the TPG to a simple temporal network (STN) by annotating the (qualitative) temporal precedences with (quantitative) upper and lower time bounds that can encode both different lengths of edges and kinematic constraints of the robots in form of minimum and maximum velocity limits. STNs are widely used for temporal reasoning in artificial intelligence due to their expressive power, simplicity, and tractability. The STN is identical to the TPG but each edge $e = (x, y)$ is now annotated with upper and lower time bounds $[LB(e), UB(e)]$ that specify that event y must be scheduled between $LB(e)$ and $UB(e)$ time units after event x.

MAPF-POST determines arrival times that are consistent with the temporal constraints imposed by the upper and lower bounds. It can use shortest path computations on the distance graph of the STN to minimize the makespan in strongly polynomial time [6] but solves a linear program instead to maximize the safety distance [5], which is slower. The schedule given by the arrival times is then sent to the robots for execution.

IV. IMPROVED EXECUTION MONITORING

MAPF-POST has two disadvantages. First, the user cannot specify the desired safety distance between robots (which typically depends on their size) directly, which forces one to set δ with trial and error to achieve a desired safety distance. The user should ideally be able to specify the safety distance directly. Second, MAPF-POST uses linear programming and can thus be sped up.

A. Improved MAPF-POST

We now present an improved version of MAPF-POST that remedies both disadvantages by using more safety

Algorithm 1: Constructing the improved TPG.

Data: A TAPF solution with makespan T for N robots consisting of path $[u_0^j, \ldots, u_T^j]$ for each robot $j \in \{1 \ldots N\}$.
Result: The improved TPG $\mathcal{G}$.

```
/* add vertices and type 1 edges */
for j ← 1 to N do
    Create a new vertex (referenced by x) and add it to G
    v(0, j) ← x
    t' ← 0
    for t ← 1 to T do
        if u_{t-1}^j ≠ u_t^j then
            for k ← 1 to dist(u_{t-1}^j, u_t^j)/δ − 1 do
                Create a new vertex (referenced by y) and add it to G
                Add edge (x, y) to G
                succ(x) ← y
                x ← y
            succloc(t', j) ← u_t^j
            t' ← t
            Create a new vertex (referenced by y) and add it to G
            v(t, j) ← y
            pred(y) ← x
            Add edge (x, y) to G
            succ(x) ← y
            x ← y
        else
            v(t, j) ← NULL
    succloc(t', j) ← NULL
/* add type 2 edges */
for t ← 0 to T do
    for j ← 1 to N do
        if v(t, j) ≠ NULL then
            for t' ← t + 1 to T do
                for j' ← 1 to N do
                    if j' ≠ j and u_t^j = u_{t'}^{j'} and v(t', j') ≠ NULL then
                        Add edge (v(t, j), pred(v(t', j'))) to G
                        Add edge (succ(v(t, j)), v(t', j')) to G
                        if succloc(t, j) = succloc(t', j') then
                            x ← succ(v(t, j))
                            y ← v(t', j'))
                            for j' ← 1 to dist(u_t^j, succloc(t, j))/δ − 2 do
                                x ← succ(x)
                                y ← succ(y)
                                Add edge (x, y) to G
                        Break out of two loops
```

markers and temporal precedence edges in the TPG. We assume that all edge lengths are a multiple of the user-specified parameter $\delta > 0$. We also assume that each robot traverses each microedge with constant velocity and stays at (main or auxiliary) locations only for an instant before it reaches its goal location. However, its velocity can change instantaneously at each location. Our implementations use controllers that approximate this assumption.

The improved MAPF-POST first splits all edges of graph G into sequences of microedges of length δ each, connected via auxiliary locations. It then converts a given TAPF solution (with auxiliary locations added) to the (now improved) TPG as before, except that the type 2 edges are different. MAPF-POST iterates over every pair of vertices x and x' in the TPG where two different robots a and a' arrive at the same (main or auxiliary) location. The robots cannot arrive at the location at the same time in the TAPF solution since the paths in the TAPF solution are collision-free. (For simplicity, one can define the arrival time at an auxiliary

location between two main locations as the minimum of the arrival times at these main locations.) Assume, without loss of generality, that robot a arrives at the location before robot a' in the TAPF solution. MAPF-POST then adds two type 2 edges to the TPG, one from x to the (unique) vertex y such that (y, x') is a type 1 edge and one from the (unique) vertex z to vertex x' such that (x, z) is a type 1 edge.

The improved MAPF-POST then converts the improved TPG to an STN, as before, and finally determines arrival times that minimize the makespan and are consistent with the temporal constraints imposed by the upper and lower bounds with shortest path computations on the distance graph of the STN [5] rather than a linear program since the temporal constraints guarantee a safety distance between robots.

The improved MAPF-POST typically runs much faster than the original MAPF-POST even though its STN is larger. We can decrease the runtime of the improved MAPF-POST by reducing the number of temporal precedence edges. We can remove some or all of those type 2 edges that are in the transitive closure of the remaining temporal precedence edges and thus implied by them. Algorithm 1 shows a way of converting a given TAPF solution (without auxiliary locations added) to the improved TPG efficiently. For example, the optimal solution of the running TAPF instance is provided as follows to Algorithm 1: $p^1 = [u_0^1, u_1^1, u_2^1, u_3^1, u_4^1] = [A, B, F, G, H]$, $p^2 = [u_0^2, u_1^2, u_2^2, u_3^2, u_4^2] = [E, F, G, H, I]$, and $p^3 = [u_0^3, u_1^3, u_2^3, u_3^3, u_4^3] = [F, G, H, C, D]$. Figure 3 shows the resulting improved TPG, which could be simplified further by contracting all safety markers that have only one incoming and one outgoing edge.

B. Properties of Improved MAPF-POST

We now prove how large the safety distance between robots is that the improved MAPF-POST guarantees for the user-specified parameter δ. The calculation of the safety distance depends on how the distance between two robots is measured. In particular, their Euclidean distance (that is, their straight-line distance in the continuous environment) can be smaller (but not larger) than their distance in the graph.

Lemma 4.1: Two robots cannot be at the same main or auxiliary location nor traverse the same microedge in opposite directions at the same time.

Proof: Consider two vertices x and x' in the TPG that correspond to two robots arriving at the same main location. Two robots cannot be at that main location at the same time in the TAPF solution since the paths in the TAPF solution are collision-free. Thus, one robot arrives at the main location after the other robot in the TAPF solution and the other robot has left the main location again at that time. This means that there is a path of temporal precedence edges from one of the vertices x or x' to the other one that starts with a type 1 edge. Thus, one of the robots must have left the main location and arrived at the auxiliary location after it before the other robot arrives at the main location, and they are not at the main location at the same time.

Now consider two safety markers x and x' in the TPG that correspond to two robots arriving at the same auxiliary location. Let the auxiliary location be somewhere between main locations u and v. Assume that robot a moves from u to v and robot a' moves from u (v) to v (u). Both robots cannot be at the same main location nor move from u to v and v to u at the same time in the TAPF solution. Thus, robot a' arrives at location u (v) no earlier (later) than robot a arrives at location v or robot a' arrives at location v (u) no later (earlier) than robot a arrives at location u. This means that there is a path of temporal precedence edges from one of the vertices x or x' to the other one that starts with a type 1 edge. Thus, both robots are not at the auxiliary location at the same time.

Finally, consider a microedge somewhere between main locations u and v. Consider a type 1 edge (x,y) in the TPG that corresponds to robot a traversing the microedge in the direction from u to v and a type 1 edge (y',x') that corresponds to robot a' traversing the microedge in the opposite direction. Both robots cannot be at the same main location nor move from u to v and v to u at the same time in the TAPF solution since the paths in the TAPF solution are collision-free. Thus, robot a' arrives at location v later than robot a arrives at location v or robot a' arrives at location u earlier than robot a arrives at location u. This means that there is a path of temporal precedence edges from x' to x or from y to y' that starts with a type 1 edge. Thus, both robots cannot traverse the microedge in opposite directions at the same time. ■

Lemma 4.2: The distance between two robots in the graph is at least δ if at least one of them is at a main or auxiliary location.

Proof: Robots always stay at locations only for an instant before they reach their goal location. Thus, the times they are at (main or auxiliary) locations coincides with the times they arrive at them. Consider a vertex x in the TPG that corresponds to robot a arriving at location u. Assume, for a proof by contradiction, that the distance in the graph between robot a and another robot a' is less than δ at that time. Since robot a' cannot be at u according to Lemma 4.1, it must be in the interior of a microedge incident on u, which means that it arrived not long ago or will soon arrive at u. Consider the vertex x' in the TPG that corresponds to robot a' arriving at u. We distinguish two cases: First, robot a' moves away from u. Then, robot a' arrived at u before robot a. This means that there is a path of temporal precedence edges from x' to x that starts with a type 1 edge. Thus, their distance is at least δ when robot a arrives at u (and continues to be at least δ in case robot a has reached its goal location), which is a contradiction. Second, robot a' moves toward u. Then, robot a arrived at u before robot a'. This means that there is a path of temporal precedence edges from x to x' that ends in a type 1 edge. Thus, their distance is at least δ when robot a arrives at u and thus also while robot a is at u (since robot a cannot have reached its goal location yet according to Lemma 4.1), which is a contradiction. ■

Theorem 4.1: The distance between robots in the graph is at least δ.

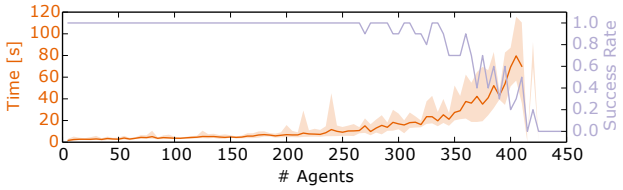
Proof: The theorem follows from Lemma 4.2 if at least one of two robots is at a main or auxiliary location. Otherwise, we distinguish four cases where the distance between two robots could potentially be less than δ in the graph and show that it is not: First, both robots traverse the interior of the same microedge in the same direction or the interiors of two incident microedges toward each other at the same time. One of the robots will arrive at a location incident on its microedge first according to Lemma 4.1. Their distance in the graph will be less than δ at that time, which is a contradiction with Lemma 4.2. Thus, this case is impossible. Second, both robots traverse the interior of the same microedge in opposite directions at the same time, which is a contradiction with Lemma 4.1. Third, both robots traverse the interiors of incident microedges (u, v) and (u, w) away from each other at the same time. One of them left u last and their distance in the graph was less than δ at that time, which is a contraction with Lemma 4.2. Thus, this case is impossible. Fourth, both robots traverse incident microedges (u, v) and (v, w) in the same direction at the same time. Let robot a traverse microedge (u, v) and robot a' traverse microedge (v, w). Robot a' left v no later than robot a left u since otherwise their distance in the graph would be less than δ at that time, which would be a contraction with Lemma 4.2. Robot a' will arrive at w no later than robot a will arrive at v for the same reason. Thus, both robots traverse their microedges from the time when robot a leaves u to the time when robot a' arrives at w. Their distance in the graph is at least δ at those times according to Lemma 4.2. Since both robots move with constant velocities, their distance in the graph is at least δ between those times as well. ■

Theorem 4.2: The Euclidean distance between two robots is at least $\delta/\sqrt{2}$ for four-neighbor square grids G and six-neighbor cubic grids G.

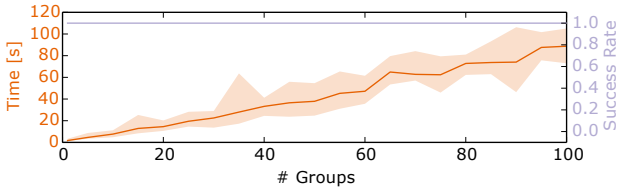
Proof: Consider two robots a and a'. Their distance in the graph is at least δ according to Theorem 4.1. We distinguish three cases and show that their Euclidean distance is at least $\delta/\sqrt{2}$ in each case: First, both robots are on the same microedge. Then, their Euclidean distance is at most δ and thus exactly δ according to Theorem 4.1. Second, both robots are on incident microedges. Let robot a be on microedge (u, v) at distance x from v and robot a' be on microedge (v, w) at distance y from v. Then, their distance in the graph is $x + y \geq \delta$. If both microedges are parallel, then the Euclidean distance of both robots is equal to their distance in the graph and thus $x + y \geq \delta$. If both microedges are orthogonal, then the Euclidean distance of both robots is $\sqrt{x^2 + y^2}$ with $\sqrt{x^2 + y^2} \geq \delta/\sqrt{2}$ since $x^2 - 2xy + y^2 = (x - y)^2 \geq 0 \Rightarrow x^2 + y^2 \geq 2xy \Rightarrow 2(x^2+y^2) = 2x^2+2y^2 \geq x^2+y^2+2xy \geq (x+y)^2 \geq \delta^2 \Rightarrow \sqrt{x^2 + y^2} \geq \delta/\sqrt{2}$. Third, both robots are on non-incident microedges. Then, their Euclidean distance is at least δ. ■

V. EXPERIMENTS

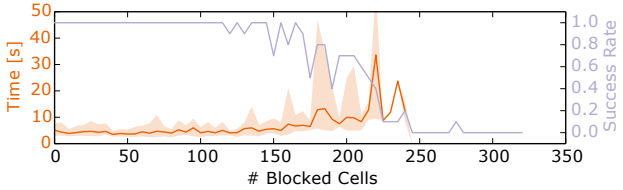
We first evaluate the runtime of CBM on 3D TAPF instances. We then evaluate the makespan, safety distance, and runtime of the original and improved MAPF-POST on 2D TAPF instances. Both of these experiments are run on a laptop computer (i7-4600U 2.1 GHz and 12 GB RAM). We finally demonstrate the efficiency and effectiveness of TAPF*, the combination of CBM and the improved MAPF-POST, on formation-change instances for robot groups in 2D and 3D simulation and on ground robots. The supplemental material contains videos. While CBM produces paths with wait actions, MAPF-POST does not use them and robots are thus not required to stay in place.



(a) Varying the number of robots split into five groups of equal size with no blocked cells.



(b) Varying the number of groups with 100 robots split into groups of equal size and no blocked cells.



(c) Varying the number of blocked cells with 100 robots split into five groups of equal size.

Fig. 5. Success rate and runtime of CBM. The shaded area indicates the runtime range for the solved TAPF instances.

A. Experiments with CBM

We use CBM to solve TAPF instances on $10 \times 10 \times 5$ six-neighbor cubic grids with random start and goal cells, blocked cells, and assignments of robots to groups. We vary the number of robots, groups, and blocked cells. We consider an instance to be solved iff CBM finds an optimal solution within a runtime limit of 120 s.

Figure 5 shows the success rate and runtime of CBM for the solved TAPF instances averaged over ten trials each. The success rate begins to drop at about 300 robots (60 percent robot density), remains at 100 percent independent of the number of groups, and begins to drop at 150 blocked cells (30 percent blocked cell density). The runtime of CBM for the solved TAPF instances increases approximately linearly with the number of robots until about 250 robots (50 percent robot density), increases approximately linearly with the number of groups (being smallest for a single group), and remains approximately constant independent of the number

TABLE I. Makespan, safety distance, and runtime of the original and improved MAPF-POST.

Instance	MAPF-POST	δ	Makespan	Safety Distance	Runtime
Corridor3	Original	0.40	106.0 s	0.53 m	0.4 s
	Improved	1.00	111.0 s	0.70 m	0.1 s
	Improved	0.50	97.0 s	0.35 m	0.2 s
Kiva100	Original	0.40	74.4 s	0.14 m	134.6 s
	Improved	1.00	75.6 s	0.70 m	3.8 s
	Improved	0.50	72.5 s	0.35 m	7.8 s
	Improved	0.25	70.8 s	0.17 m	22.0 s
	Improved	0.20	70.4 s	0.14 m	32.0 s

of blocked cells until the success rate begins to drop at about 150 blocked cells (30 percent blocked cell density).

B. Experiments with Improved MAPF-POST

We implement the improved MAPF-POST in C++ using the `BOOST GRAPH` library [7] for the shortest path computations. We use the original and improved MAPF-POST to solve TAPF instances on four-neighbor square grids since the original MAPF-POST implementation does not support cubic grids. Table I shows the makespan, empirical safety distance (measured as the minimum distance of any two robots during an experiment), and runtime for different values of δ for two TAPF instances with virtual differential-drive robots. Since the user cannot specify the desired safety distance directly for the original MAPF-POST, we tuned its value of δ to result in a reasonable safety distance. The first TAPF instance ("Corridor3") has twenty robots swap sides in equal numbers in a 7×17 grid with a narrow corridor. This TAPF instance is challenging to solve for CBM, but the runtimes of the original and improved MAPF-POST are less than one second. The second TAPF instance ("Kiva100") has 100 robots in a 24×55 grid that models a warehouse domain [8]. The runtime of the original MAPF-POST is larger than two minutes since it uses linear programming. Furthermore, the resulting safety distance of $0.14\,\mathrm{m}$ is too small for deployment on medium-sized robots. In contrast, the improved MAPF-POST runs faster and achieves a larger safety distance.

C. Experiments in Simulation

We use TAPF* in conjunction with the robot simulator V-REP [9] to simulate differential drive robots, six-legged robots, and quadcopters. The differential drive robots have a maximum angular velocity of $4.2\,\mathrm{rad/s}$. CBM treats them as holonomic robots, and the improved MAPF-POST then uses a large safety distance to account for their non-holonomic constraints. Figure 6(a) shows an example formation-change instance on simulated quadcopters.

D. Experiments on Robots

We use eight iRobot Create2 robots equipped with single-board computers (such as ODROID-C1+), that interface to the robots via their serial ports. The computers run the improved MAPF-POST on Ubuntu 14.04 with ROS Jade and communicate via WiFi with a single `roscore` on a host computer that runs CBM. Localization is provided by a 12-camera VICON MX optical motion capture system in a space approximately $5\,\mathrm{m} \times 4\,\mathrm{m}$ in size. CBM and MAPF-POST use a grid cell size of $0.75\,\mathrm{m} \times 0.75\,\mathrm{m}$. Our chosen value of $\delta = 0.75\,\mathrm{m}$ guarantees a safety distance of $0.75\,\mathrm{m}/\sqrt{2} \approx 0.53\,\mathrm{m}$, assuming perfect execution on holonomic robots. To prevent collisions of the robots (whose diameter is $0.35\,\mathrm{m}$) due to this wrong assumption, the improved MAPF-POST limits their maximum translational velocity to $0.2\,\mathrm{m/s}$, even though the robots can move with a translational velocity of up to $0.5\,\mathrm{m/s}$. Figures 6(b) and 6(c) show an example formation-change instance on the robots.

VI. RELATED WORK

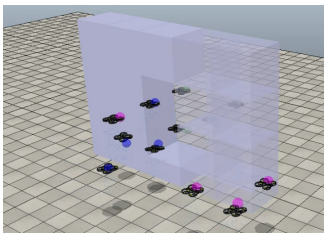
The research reported in [10] is not concerned with formation change for robot groups but shares some ideas with our approach. We now discuss related work in robotics not covered in [5] and [3]. Methods for formation control try to move robots in a given formation and restore the formation in case the robots split up in order to move around obstacles. Examples include behavior-based [11], leader-follower [12], virtual structure [13], potential field [14], and graph-based [15] methods. Formation-control methods, different from TAPF*, are often decentralized, assume that no robots are interchangeable with each other, and provide no guarantees on solution quality. Research done in the context of quadcopters includes [16] and [17].

Simple formation-change methods assume, different from TAPF*, that the robots are not allowed to split up in order to move around obstacles [18]. Many of the more general formation-change methods assume that all robots are interchangeable with each other. Some of these methods are centralized and optimal [19]. Research done in the context of quadcopters includes [20] and [21], but these formation-change methods, different from TAPF*, tend to be very slow since they consider richer kinematic and dynamic constraints. Other methods are, different from TAPF*, decentralized and provide no guarantees on solution quality [22]. Research done in the context of quadcopters includes [21]. One interesting application that makes the latter assumption are the PixelBots, which use formation changes to implement a display. PixelBot implementations exist for swarms of both differential-drive robots [23] and quadcopters [24].

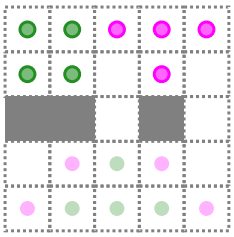
Some methods for formation change, like TAPF*, consider groups of robots. For example, K-color multi-robot motion planning (where K is the number of groups), different from TAPF*, considers robots of polygonal shapes in a continuous environment but provides no guarantees on solution quality and is very slow [25].

VII. CONCLUSIONS

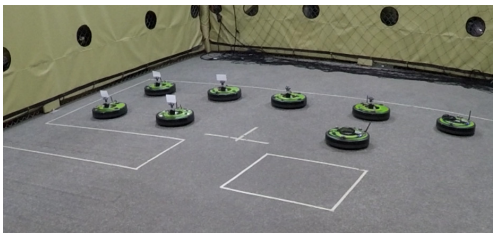
In this paper, we solved the formation-change problem for robot groups in known environments. We are given a team of robots partitioned into groups, where robots in the same group are interchangeable with each other. Our algorithm TAPF* has the following features: (a) it incorporates kinematic constraints of robots in form of velocity limits; (b) it maintains a user-specified safety distance between robots; (c) it attempts to minimize the makespan; and (d) it runs



(a) Formation-change instance with a V-REP simulation of three groups of four quadcopters each, differentiated by color. Some quadcopters fly through and others around the obstacle.



(b) TAPF instance with two groups of four robots each, differentiated by color. The start formation is on top, and the goal formation is at the bottom.



(c) Formation-change instance on Create2 robots that corresponds to the TAPF instance in (b). The four robots with a white flag (on the left) belong to the same group.

Fig. 6. Examples of experiments in simulation and on robots.

efficiently for hundreds of robots and dozens of groups even in dense 3D environments with narrow corridors and other occlusions. TAPF* achieves efficiency by combining two of our recent methods sequentially that both minimize the makespan: The first method is CBM, a path-planning method that solves a version of the formation-change problem that ignores the physical properties of robots (such as their sizes and kinematic constraints) to search efficiently. The second method is MAPF-POST, an execution-monitoring method that takes the physical properties of robots into account when using a simple temporal network to determine appropriate velocities that allow them to follow the collision-free paths determined by CBM. We did not only combine CBM and MAPF-POST but also provided several improvements to MAPF-POST: The user can now specify the desired safety distance directly, and it is more efficient since it no longer uses linear programming.

In the future, we intend to use TAPF* for online re-planning in case one or more robots deviate from their nominal velocities. Many such cases require TAPF* only to add temporal constraint edges to the simple temporal network and re-solve it, which can be done much faster than running CBM.

REFERENCES

[1] J. Yu and S. M. LaValle, "Multi-agent path planning and network flow," in *Workshop on the Algorithmic Foundations of Robotics*, 2012, pp. 157–173.

[2] 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 *AAAI Conference on Artificial Intelligence*, 2016, pp. 3166–3173.

[3] H. Ma and S. Koenig, "Optimal target assignment and path finding for teams of agents," in *International Conference on Autonomous Agents and Multiagent Systems*, 2016, pp. 1144–1152.

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

[5] W. Hönig, T. K. S. Kumar, L. Cohen, H. Ma, H. Xu, N. Ayanian, and S. Koenig, "Multi-agent path finding with kinematic constraints," in *International Conference on Automated Planning and Scheduling*, 2016, pp. 477–485.

[6] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," *Artificial Intelligence*, vol. 49, no. 1-3, pp. 61–95, 1991.

[7] J. G. Siek, L. Lee, and A. Lumsdaine, *The Boost Graph Library - User Guide and Reference Manual*. Pearson / Prentice Hall, 2002.

[8] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI Magazine*, vol. 29, no. 1, pp. 9–20, 2008.

[9] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *International Conference on Intelligent Robots and Systems*, 2013, pp. 1321–1326.

[10] M. Cirillo, F. Pecora, H. Andreasson, T. Uras, and S. Koenig, "Integrated motion planning and coordination for industrial vehicles," in *International Conference on Automated Planning and Scheduling*, 2014, pp. 463–471.

[11] T. R. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 926–939, 1998.

[12] T. D. Barfoot and C. M. Clark, "Motion planning for formations of mobile robots," *Robotics and Autonomous Systems*, vol. 46, no. 2, pp. 65–78, 2004.

[13] M. A. Lewis and K. Tan, "High precision formation control of mobile robots using virtual structures," *Autonomous Robots*, vol. 4, no. 4, pp. 387–403, 1997.

[14] N. E. Leonard and E. Fiorelli, "Virtual leaders, artificial potentials and coordinated control of groups," in *Decision and Control*, vol. 3, 2001, pp. 2968–2973.

[15] J. P. Desai, J. P. Ostrowski, and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 905–908, 2001.

[16] N. Michael and V. Kumar, "Control of ensembles of aerial robots," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1587–1602, 2011.

[17] M. Turpin, N. Michael, and V. Kumar, "Decentralized formation control with variable shapes for aerial robots," in *International Conference on Robotics and Automation*, 2012, pp. 23–30.

[18] A. Krontiris, S. J. Louis, and K. E. Bekris, "Multi-level formation roadmaps for collision-free dynamic shape changes with non-holonomic teams," in *International Conference on Robotics and Automation*, 2012, pp. 1570–1575.

[19] L. Liu and D. A. Shell, "Multi-robot formation morphing through a graph matching problem," in *Distributed Autonomous Robotic Systems*, 2012, pp. 291–306.

[20] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Autonomous Robots*, vol. 35, no. 4, pp. 287–300, 2013.

[21] M. Turpin, N. Michael, and V. Kumar, "Capt: Concurrent assignment and planning of trajectories for multiple robots," *International Journal of Robotics Research*, vol. 33, no. 1, pp. 98–112, 2014.

[22] N. Michael, M. M. Zavlanos, V. Kumar, and G. J. Pappas, "Distributed multi-robot task assignment and formation control," in *International Conference on Robotics and Automation*, 2008, pp. 128–133.

[23] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. A. Beardsley, "Image and animation display with multiple mobile robots," *International Journal of Robotics Research*, vol. 31, no. 6, pp. 753–773, 2012.

[24] J. Alonso-Mora, M. Schoch, A. Breitenmoser, R. Siegwart, and P. A. Beardsley, "Object and animation display with multiple aerial vehicles," in *International Conference on Intelligent Robots and Systems*, 2012, pp. 1078–1083.

[25] K. Solovey and D. Halperin, "k-color multi-robot motion planning," *International Journal of Robotics Research*, vol. 33, no. 1, pp. 82–97, 2014.