A Complete Algorithm for Generating Safe Trajectories for Multi-Robot Teams

Sarah Tang and Vijay Kumar

Abstract In this paper, we consider the problem of planning collision-free trajectories to navigate a team of labeled robots from a set of start locations to a set of goal locations, where robots have pre-assigned and non-interchangeable goals. We present a solution to this problem for a centralized team operating in an obstacle-free, two-dimensional workspace. Our algorithm allows robots to follow Optimal Motion Plans (OMPs) to their goals when possible and has them enter Circular HOlding Patterns (CHOPs) to safely navigate congested areas. This OMP+CHOP algorithm is shown to be safe and complete, and simulation results show scalability to hundreds of robots.

1 Introduction

Multi-robot systems have become attractive solutions for a wide variety of tasks. One prominent initiative is the Amazon Prime Air project [5], which proposes using autonomous Unmanned Aircraft Systems (UASs) to deliver packages under five pounds to customers within a ten mile radius of a fulfillment center in less than 30 minutes. In this setting, hundreds to thousands of robots could be in the air simultaneously. Each robot is assigned a fixed and non-interchangeable goal, or *labeled*.

While it may seem promising to simply stagger the UASs' altitudes, recent Federal Aviation Administration (FAA) proposed guidelines [4] limit the maximum altitude of these small UASs to 400 feet, essentially confining vehicles to the horizontal plane. Thus, this work focuses on finding safe motion plans for robots operating in a two-dimensional space. This problem is harder than the three-dimensional version, because the latter provides an additional degree of freedom.

There are, broadly speaking, three guarantees of interest for planning algorithms: *safety* — robots will be collision-free with obstacles and each other, *optimality* —

S. Tang and V. Kumar
University of Pennsylvania, e-mail: sytang,kumar@seas.upenn.edu

The final publication is available at Springer via https://doi.org/10.1007/978-3-319-60916-4_34.

the solution is minimum cost, and *completeness* — the planner will always find a solution if one exists and indicate that there is none if one does not.

Approaches to the labeled multi-robot planning problem can be characterized as *coupled* or *decoupled*. *Coupled* planners search for optimal paths in the joint configuration space of all team members, either by directly applying planning algorithms such as A* [7] or with specialized variants [6]. These approaches typically guarantee optimality and completeness. However, as the search space grows exponentially with the number of robots, they quickly become computationally impractical.

Decoupled planners, on the other hand, plan for each robot separately. One approach is to plan each robot's motion in priority order. Lower priority robots must avoid higher priority ones [1] [3]. An alternative is to first find paths that avoid static obstacles, then design velocity profiles that avoid inter-robot collisions [9] [11]. These planners tend to be faster, but are typically not complete.

As a result, algorithms that combine both approaches have been proposed. van den Berg et al. [17] decouple the problem into smaller coupled subproblems, minimizing the dimensionality of the highest-dimensional subproblem. Subdimensional expansion [18] first plans in each robot's individual configuration space and searches a joint state space in regions where collisions occur. These approaches offer significant computational improvements, but can still perform poorly in the worst case.

Other planning approaches include rule-based [2] or network flow [19] algorithms. Alternatively, van den Berg et al. [16] frame the problem as a reciprocal collision avoidance problem. In air traffic control, Tomlin et al. [13] find safe conflict resolution maneuvers in the presence of uncertainties. However, this approach requires computation of solutions to the Hamilton-Jacobi-Isaacs PDE equation, which becomes computationally difficult for large teams.

Other settings allow for robots to be completely interchangeable. Proposed solutions to the *unlabeled* multi-robot planning problem must solve both the task assignment and trajectory generation problems [14]. In particular, Turpin et. al propose an $O(N^3)$ solution to the unlabeled planning problem in obstacle-free environments [14] for teams of N robots.

Our work proposes a centralized algorithm for finding collision-free trajectories for a team of labeled robots operating in an obstacle-free two-dimensional workspace. In essence, each robot pursues its own optimal motion plan until an impending collision is detected. This causes the affected robots to enter a holding pattern, similar to the racetrack patterns used in civilian aviation in congested airspace. Our approach is similar to subdimensional expansion, however, collisions are resolved through analytically constructed maneuvers as opposed to a high-dimensional graph search. While this is suboptimal, our algorithm offers completeness guarantees and allows for greater scalability to large teams.

The remainder of this paper will proceed as follows. Section 2 presents terminology and Section 3 discusses a known solution to the unlabeled multi-robot planning problem. Section 4 details our algorithm for the labeled problem and discusses its safety and completeness guarantees. Section 5 characterizes the algorithm's performance through simulation experiments and Section 6 presents conclusions and directions for future work.

2 Problem Definition

Let $\mathscr{I}_Z = \{1,2,...,Z\}$ denote the set of integers between 1 and Z, inclusive. Let N denote the number of robots in the team. We represent the start and goal positions of robot $i \in \mathscr{I}_N$ with $\mathbf{s}_i \in \mathbb{R}^2$ and $\mathbf{g}_i \in \mathbb{R}^2$, respectively, and the sets of all start and goal positions with S and G, respectively. $\mathbf{x}$ denotes a position in $\mathbb{R}^2$ and $\mathbf{x}_i$ denotes the state of robot i. Each robot has identical first-order dynamics:

$$\dot{\mathbf{x}}_i(t) = \mathbf{u}_i(t), \quad \|\mathbf{u}_i(t)\|_2 \leq v_{max} \tag{1}$$

In the centralized paradigm, each robot knows the states and goals of all robots.

We define a trajectory as a piecewise smooth function of time, $\gamma(t) : [t_0, t_f] \rightarrow \mathbb{R}^2$. Let $\gamma^{(\mathbf{x}_0,\mathbf{x}_1)}(t)$ denote an optimal trajectory between $\mathbf{x}_0$ and $\mathbf{x}_1$ and $\gamma_i(t)$ denote robot i's trajectory between $\mathbf{s}_i$ and $\mathbf{g}_i$. Let $\boldsymbol{\gamma}(t)$ denote the set of all trajectories $\gamma_i(t)$.

We model each robot as a disk of radius R. We use $\mathscr{B}(\mathbf{x}_i(t))$ to denote the area robot i occupies at $\mathbf{x}_i(t)$ and $\mathscr{B}(\gamma_i)$ to denote the area it sweeps out traversing $\gamma_i(t)$.

The goal of the labeled planning problem is to plan a trajectory $\gamma_i(t)$ for each robot such that $\gamma_i(0) = \mathbf{s}_i$, $\gamma_i(t_{f,i}) = \mathbf{g}_i$. All robots' trajectories start simultaneously but each robot can reach its goal at a unique $t_{f,i}$. We assume robots remain stationary at their goals for all $t > t_{f,i}$, and we require $\mathscr{B}(\mathbf{x}_i(t)) \cap \mathscr{B}(\mathbf{x}_j(t)) = \emptyset$ for all $t \in [0, \max_{i \in \mathscr{I}_N} t_{f,i}]$, $j \neq i \in \mathscr{I}_N$.

3 Concurrent Assignment and Planning of Trajectories (CAPT)

First, consider the unlabeled planning problem: given N robots and M goals, plan a trajectory $\gamma_i(t)$ for each robot such that each goal is visited by one robot. When $M > N$, some goals will remain unvisited while when $M < N$, some robots will not visit any goals. In this section, we outline the Concurrent Assignment and Planning of Trajectories (CAPT) algorithm [14] to solve this problem.

Suppose the start and goal locations are at least $2\sqrt{2}R$ away from each other:

$$\|\mathbf{s}_i - \mathbf{s}_j\|_2 > 2\sqrt{2}R \quad \forall i \neq j \in \mathscr{I}_N, \quad \|\mathbf{g}_i - \mathbf{g}_j\|_2 > 2\sqrt{2}R \quad \forall i \neq j \in \mathscr{I}_M \tag{2}$$

Define the assignment mapping robots to goals as $\phi : \mathscr{I}_N \rightarrow \mathscr{I}_M \cup 0$, where $\phi_i = j$ indicates that robot i is assigned to goal j and $\phi_i = 0$ if robot i is unassigned. The CAPT algorithm finds the assignment and trajectories that solve:

$$\min_{\phi, \boldsymbol{\gamma}(t)} \sum_{i=1}^{N} \int_0^{t_f} \dot{\mathbf{x}}_i(t)^T \dot{\mathbf{x}}_i(t) dt \tag{3}$$

The solution to this problem consists of straight-line trajectories that minimize the sum of the squares of the distances traveled. In other words, the optimal assignment is given by:

$$\phi^\star = \underset{\phi}{\operatorname{argmin}} \sum_{i=1}^{N} \|\mathbf{s}_i - \mathbf{g}_{\phi_i}\|_2^2 \tag{4}$$

This assignment can be found in $O(N^3)$ time using the Hungarian Algorithm [10].

Denote the assigned goal of robot i with $\mathbf{g}_i^\star$, where $\mathbf{g}_i^\star = \mathbf{s}_i$ if robot i is unassigned and $\mathbf{g}_i^\star = \mathbf{g}_{\phi_i^\star}$ otherwise. The optimal trajectories are the constant velocity straight-line trajectories from $\gamma_i(0) = \mathbf{s}_i$ to $\gamma_i(t_f) = \mathbf{g}_i^\star$. We want all robots to arrive at their goals simultaneously at t_f, which can be found with:

$$t_f = \max_{i \in \mathscr{I}_N} \frac{\|\mathbf{s}_i - \mathbf{g}_i^\star\|_2}{v_{max}} \tag{5}$$

We will refer to such trajectories as *synchronized*. Turpin et al. show these trajectories are collision-free [14].

4 Optimal Motion Plans + Circular HOlding Patterns (OMP+CHOP) for the Labeled Planning Problem

We now present our algorithm to solve the labeled planning problem. First, we discuss the best-case scenario, where all robots can move directly to their goals following an Optimal Motion Plan (OMP). Next, we consider the worst-case scenario, where all robots must enter a single Circular HOlding Pattern (CHOP). Finally, we describe the full algorithm, which combines these two strategies.

We again assume the start and goal positions satisfy the separation assumptions given in Eq. 2, however, in the labeled setting, $M = N$.

4.1 Optimal Motion Plans (OMPs)

Given any two waypoints and times of arrival at these points, we can design an optimal trajectory taking robot i from $\mathbf{x}_i(t_0) = \mathbf{x}_0$ to $\mathbf{x}_i(t_f) = \mathbf{x}_f$ by solving:

$$\begin{aligned} \gamma^{(\mathbf{x}_0, \mathbf{x}_f)}(t) = \underset{\gamma(t)}{\operatorname{argmin}} & \int_{t_0}^{t_f} \dot{\mathbf{x}}_i(t)^T \dot{\mathbf{x}}_i(t) dt \\ \text{subject to: } & \gamma(t_0) = \mathbf{x}_0,\ \gamma(t_f) = \mathbf{x}_f \end{aligned} \tag{6}$$

As before, the optimal trajectory is the constant-velocity straight-line path:

$$\gamma^{(\mathbf{x}_0, \mathbf{x}_f)}(t) = \left(\mathbf{x}_f - \mathbf{x}_0\right) \frac{t - t_0}{t_f - t_0} + \mathbf{x}_0 \tag{7}$$

The Optimal Motion Plan (OMP) for a robot is the optimal trajectory from its current position to its goal. In the best case, all robots' OMPs from their start positions are collision-free. Then, for each robot, $\gamma_i(t) = \gamma^{(\mathbf{s}_i, \mathbf{g}_i)}(t)$, $t_0 = 0$, and $t_{f,i} = \frac{\|\mathbf{s}_i - \mathbf{g}_i\|_2}{v_{max}}$. Trajectories are *unsynchronized*: all robots travel at v_{max} to arrive at different times.

4.2 Circular HOlding Patterns (CHOPs)

When their OMPs are not collision-free, robots enter a Circular HOlding Pattern (CHOP) to safely maneuver to their goals. Algorithm 1 presents the CHOP construction algorithm and Sections 4.2.1-4.2.4 detail its key steps. Its inputs are the *CHOP start time*, τ_s, the index set of robots involved, $\mathscr{R}_m$, a set of *CHOP start positions*, X_s, from which robots enter the CHOP, and the set of goals $X_g = \{\mathbf{g}_i \mid i \in \mathscr{R}_m\}$. The equality sign denotes the assignment of a value, a left arrow indicates the addition of discrete elements to an existing set, and $X_{a,i}$ denotes element i of set X_a.

For now, assume all robots immediately enter a single CHOP. This represents the worst-case scenario, where robots are densely packed and smaller CHOPs cannot be created. In this case, the algorithm inputs are $\tau_s = 0$, $\mathscr{R}_m = \mathscr{I}_N$, $X_s = S$, $X_g = G$.

Algorithm 1 $(m, \mathbf{x}_c, r_c)$ = Create CHOP$(\tau_s, \mathscr{R}_m, X_s, X_g, R, v_{max})$

1: N_m = number of robots in $\mathscr{R}_m$
2: $\mathbf{x}_c = \frac{\sum_{i \in \mathscr{I}_{N_m}} X_{s,i}}{N_m}$ // Define center of the CHOP
3: $n_w = 2N_m$ // Designate number of intermediate waypoints in the CHOP
4: r_c = Find CHOP Radius$(n_w, \mathbf{x}_c, X_g, R)$ //Find minimum safe radius for the CHOP
5: // Find the set of intermediate waypoints
6: $X_m = \{\mathbf{x}_c + r_c[\cos(\theta_i) \;\; \sin(\theta_i)]^T \mid \theta_i = (i-1)\frac{2\pi}{n_w}, i \in \mathscr{I}_{n_w}\}$
7: // Assign entry waypoint for each robot
8: $X_w = \{X_{m,1}, X_{m,3}, ..., X_{m,n_w-1}\}$
9: ϕ^s = CAPT(X_s, X_w)
10: // Define Exit Condition for each robot
11: **for all** $i \in \mathscr{I}_{N_m}$ **do**
12: $\quad \phi_i^g = \text{argmin}_{j \in \mathscr{I}_{n_w}} \|X_{m,j} - X_{g,i}\|_2$ // Assign the exit waypoint
13: **end for**
14: $\mathscr{P}_i = \emptyset \;\; \forall i \in \mathscr{I}_{N_m}$ // Find priority sets
15: **for all** $i \in \mathscr{I}_{N_m}$ **do**
16: $\quad$ **for all** $j \in \mathscr{I}_{N_m} \backslash i$ **do**
17: $\quad\quad$ **if** $\mathscr{B}(X_{g,i}) \cap \mathscr{B}(\gamma^{(X_{m,\phi_j^g}, X_{g,j})}) \neq \emptyset$ **then**
18: $\quad\quad\quad \mathscr{P}_i \leftarrow j$
19: $\quad\quad$ **end if**
20: $\quad$ **end for**
21: **end for**
22: (m) = Construct CHOP$(\tau_s, \mathscr{R}_m, X_s, X_m, X_g, \phi^s, \phi^g, \mathscr{P}, v_{max})$

4.2.1 Define Intermediate Waypoints

First, we find a set of n_w *intermediate waypoints*, X_m, distributed evenly about a circle with center $\mathbf{x}_c$ and radius r_c. These waypoints must satisfy *safety conditions*:

1. The distance between all points in the set X_w, defined in Line 8, is at least $2\sqrt{2}R$.
2. The distance of every goal in X_g from every waypoint in X_m is at least $2\sqrt{2}R$.
3. The distance of every goal in X_g from every path between a pair of consecutive intermediate waypoints in X_m is at least $2R$.

We designate n_w as twice the number of robots in $\mathscr{R}_m$ and $\mathbf{x}_c$ as the mean of the robots' start positions. r_c, the minimum radius that satisfies the safety criteria, can be found analytically. Note that robots' goals can be inside or outside the CHOP.

4.2.2 Define Entry Waypoints

To enter a CHOP, robots move synchronously from their CHOP start positions to an intermediate waypoint designated as their *entry waypoint*. Line 8 chooses every other waypoint from X_m to form the set of candidate entry waypoints, X_w. In Line 9, these waypoints are assigned to robots with the optimal assignment returned by the CAPT algorithm when considering X_s as start positions and X_w as goals.

4.2.3 Define Exit Conditions

Next, robots synchronously and sequentially visit intermediate waypoints in clockwise order until they satisfy their Exit Condition (EC). First, Lines 11-13 assigns the intermediate waypoint closest to each robot's goal as its *exit waypoint*. Robots can only exit the CHOP from this waypoint. Second, Lines 14-21 construct each robot's *priority set*, $\mathscr{P}_i$. A robot can exit via its exit waypoint only if all robots in $\mathscr{P}_i$ have exited. Line 17 ensures that if robot i remaining stationary at its goal will result in a collision with robot j moving towards its goal, robot i cannot exit before robot j.

4.2.4 Construct CHOP

To execute a CHOP, each robot follows optimal trajectories to sequentially visit its CHOP start position, its entry waypoint, a series of intermediate waypoints, and its exit waypoint at the appropriate times. Upon satisfying its EC, it returns to pursuing an OMP starting from its exit waypoint. Thus, we can fully represent the motion of all robots in a CHOP with $m = \{\{X_i \mid i \in \mathscr{R}_m\}, T, T_{goal}\}$. X_i is the series of waypoints robot i visits, starting from its CHOP start position and ending with its exit waypoint. Note that the sets X_i can be different lengths. $T = \{t_1, t_2, ...\}$ indicates arrival times at waypoints, where robot i must be at position $X_{i,j}$, if it exists, at time t_j. T is common to all robots, and $|T| = \max_{i \in \mathscr{R}_m} |X_i|$, where $|\cdot|$ denotes a set's cardinality. Finally, $T_{goal} = \{t_{goal,i} \mid i \in \mathscr{R}_m\}$ is a series of *goal arrival times*. Robot i must reach its goal at time $t_{goal,i}$ after exiting the CHOP.

We already know X_i for each robot. Line 22 additionally defines the series T and T_{goal}. Section 4.4 will show that to guarantee safety, trajectories between waypoints in the CHOP and the OMPs of robots that have exited must all be synchronized. To achieve this while respecting all robots' velocity constraints, we define $t_1 = \tau_s$ and:

$$t_j = t_{j-1} + \max_{i \in \mathscr{R}_{m_j}} \frac{\|\mathbf{x}_{next,i} - X_{i,j-1}\|_2}{v_{max}} \qquad j = 2, ..., j_{max} \tag{8}$$

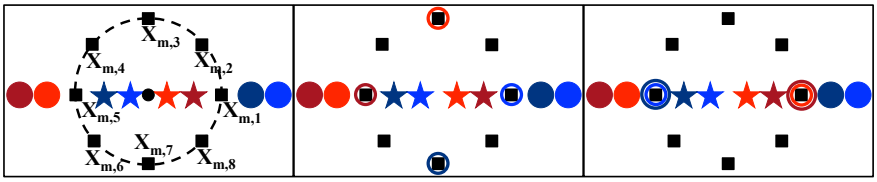
Here, $j_{max} = \max_{i \in \mathscr{R}_m} |X_i|$. $\mathscr{R}_{m_j} \subseteq \mathscr{R}_m$ is the subset of indices for which $|X_i| \geq j-1$, $\mathbf{x}_{next,i}$ refers to $X_{i,j}$ if $|X_i| \geq j$ and $X_{g,i}$ if $|X_i| = j-1$. Then:

$$t_{goal,i} = \begin{cases} t_{|X_i|+1} & \text{if } |X_i| < j_{max} \\ t_{j_{max}} + \max_{i \in \mathscr{R}_{m_{j_{max}}}} \frac{\|G_i - X_{i,j_{max}}\|_2}{v_{max}} & \text{if } |X_i| = j_{max} \end{cases} \tag{9}$$

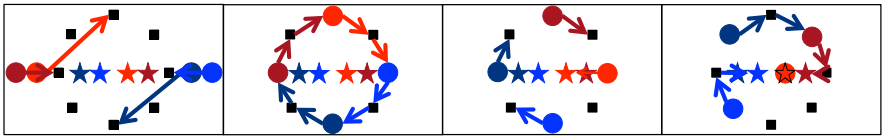
We further define the *CHOP exit time* for each robot, denoted $\tau_{f,i}$, as the time it leaves its exit waypoint, which we will use in later algorithms.

4.2.5 Example Problem

(a) Problem definition: robots begin at start positions indicated by circles and must navigate to assigned goal positions indicated by stars of the same color.



(b) Identify intermediate points (c) Identify entry waypoints (d) Identify exit conditions



(e) Resulting motion (f) (g) (h)

Fig. 1: We design a Circular HOlding Pattern (CHOP) for the problem in Fig. 1a. Figs. 1b-1d show key steps of Algorithm 1. Figs. 1e-1h illustrate the motion plan.

We illustrate Algorithm 1 using the example problem presented in Fig. 1a. Fig. 1b illustrates the placement of the intermediate waypoints, pictured as black squares. Fig. 1c shows the assigned entry waypoint for each start position with a circle of the same color. Fig. 1d shows each robot's assigned exit waypoint with a circle of the same color, with higher priority robots indicated by larger circles. Figs. 1e-1h illustrate the resulting motion plan. As an example, robot 2's planned trajectory is:

$$\gamma_2(t) = \begin{cases} \gamma^{(\mathbf{s}_2, X_{m,3})}(t) & t_0 \leq t \leq t_1 \\ \gamma^{(X_{m,3}, X_{m,2})}(t) & t_1 < t \leq t_2 \\ \gamma^{(X_{m,2}, X_{m,1})}(t) & t_2 < t \leq t_3 \\ \gamma^{(X_{m,1}, \mathbf{g}_2)}(t) & t_3 < t \leq t_{exit,2} \end{cases} \tag{10}$$

4.3 The Motion Planning Algorithm

Now, we combine the previous techniques into a single motion planning algorithm, referred to as OMP+CHOP, that allows robots to follow their OMPs when possible and directs them into appropriately designed CHOPs in congested areas. This algorithm is presented in Algorithm 2 and described in detail in Sections 4.3.1-4.3.3.

4.3.1 Compute Motion Plan

$\mathscr{M}$ contains the set of all CHOPs in the motion plan, from which the set of trajectories γ can be derived. Initially, in Line 2, $\mathscr{M}$ is empty and all robots follow

Algorithm 2 γ = OMP_CHOP(S, G, R, v_{max})

1: $\mathscr{M} = \emptyset$
2: γ = Compute Motion Plan ($S, G, \mathscr{M}$)
3: $\mathscr{C}$ = Find Imminent Collision(γ, R)
4: **while** $\mathscr{C} \neq \emptyset$ **do**
5: ($\tau_s, \mathscr{R}_{add}, X_s, X_g, \mathscr{M}_{add}$) = Compute CHOP Parameters($\gamma, \mathscr{M}, \mathscr{C}, G$)
6: ($m_{new}, \mathbf{x}_c, r_c$) = Create CHOP($\tau_s, \mathscr{R}_{add}, X_s, X_g, R, v_{max}$)
7: $\mathscr{M} \leftarrow m_{new}$
8: $\mathscr{M}$ = Remove CHOPs($\mathscr{M}_{add}$)
9: γ = Compute Motion Plan ($S, G, \mathscr{M}$)
10: $\mathscr{C}$ = Find Imminent Collision(γ, R)
11: **end while**

their OMPs from their start positions. When $\mathscr{M}$ contains CHOPs, as in Line 9, each robot follows its OMP until its earliest CHOP's start time. It then follows optimal trajectories to each waypoint designated by the CHOP to its exit waypoint, when it again pursues an OMP until it encounters another CHOP or its goal. This process is pictured in Fig. 2. The choice of CHOP parameters, described in Section 4.3.3, will guarantee that CHOPs in $\mathscr{M}$ will always start along its robots' OMPs.

We will use a subscripted variable, such as m_k, to denote a particular CHOP in $\mathscr{M}$ and $\mathscr{R}_{m_k}, \tau_{s,m_k}, \tau_{f,i,m_k}$ to denote the indices of its robots, its start time, and the CHOP exit time of robot $i \in \mathscr{R}_{m_k}$, respectively.

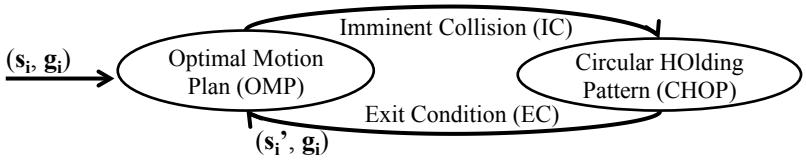


Fig. 2: Overview of our algorithm's motion plan: robots follow their OMP whenever possible, entering CHOPs to resolve collisions.

4.3.2 Find Imminent Collisions (ICs)

Line 3 finds the first Imminent Collision (IC) amongst robots following trajectories γ. We characterize a collision with its time, t_c, the number of robots in collision, N_c, and the set of indices of the colliding robots, $\mathscr{C}$. For all robots $i \in \mathscr{C}$, there must be at least one $j \neq i \in \mathscr{C}$ for which $\mathscr{B}(\mathbf{x}_i(t_c)) \cap \mathscr{B}(\mathbf{x}_j(t_c)) \neq \emptyset$. We will use $\mathscr{C}$ to denote both the collision and the set of robots in the collision.

4.3.3 Create Local CHOP

Line 5 of Algorithm 2 finds the parameters of a new CHOP, denoted m_{new}, that will resolve the detected IC. This function is presented in Algorithm 3.

m_{new} is characterized by the set of indices of the robots it contains, R_{add}. As shown in Line 1 of Algorithm 3, $\mathscr{R}_{add}$ initially contains only robots in the IC. Additionally, existing CHOPs in $\mathscr{M}$ might need to be *merged* with m_{new}. These CHOPs are contained in $\mathscr{M}_{add}$, which is initially empty. $\mathscr{M}_{curr}$, also initially empty, contains only the CHOPs to be merged that were identified in the most recent iteration.

Algorithm 3 $(\tau_s, \mathscr{R}_{add}, X_s, X_g, \mathscr{M}_{add})$ = Compute CHOP Parameters$(\gamma, \mathscr{M}, \mathscr{C}, G)$

1: $\mathscr{M}_{add} = \emptyset, \mathscr{R}_{add} = \mathscr{C}, t_s = t_c, merge = 1$ // Initialize variables
2: $\mathscr{M}_{curr} = \{m_k \in \mathscr{M} \mid \exists\, r \in \mathscr{C} \cap \mathscr{R}_{m_k}$ for which $t_c \in [\tau_{s,m_k}, \tau_{f,r,m_k}]\}$
3: **while** *true* **do**
4: // Find valid starting conditions
5: $\tau_s = \max_{t \le t_s} t$ such that $\|\mathbf{x}_i(t) - \mathbf{x}_j(t)\|_2 \ge 2\sqrt{2}R \;\; \forall j \ne i \in \mathscr{R}_{add}$
6: $X_s = \{\mathbf{x}_i(\tau_s) \mid i \in \mathscr{R}_{add}\}$, $X_g = \{\mathbf{g}_i \mid i \in \mathscr{R}_{add}\}$
7: **if** $merge == 0$, *break* **end if**
8: $(m_{curr}, \mathbf{x}_c, r_c)$ = Create CHOP$(\tau_s, \mathscr{R}_{add}, X_s, X_g, R, v_{max})$
9: // Merge robots and CHOPs whose paths intersect m_{curr}'s circle
10: $t_a = t_{1,m_{curr}}, t_b = \max_{i \in \mathscr{R}_{add}} \tau_{f,i,m_{curr}}$
11: l = Set of paths that all robots $r \in \mathscr{I}_N \setminus \mathscr{R}_{add}$ traverse between $[t_a, t_b]$
12: $\mathscr{R}_{OMP}$ = Robots whose OMP's paths are in l and intersect a circle at $\mathbf{x}_c$, radius $r_c + 2R$
13: $\mathscr{M}_{curr} \leftarrow$ CHOPs whose paths are in l and intersect a circle at $\mathbf{x}_c$, radius $r_c + 2R$
14: // Merge CHOPs that will cause conflicting motion plans for robots in $\mathscr{R}_{add}$
15: $\mathscr{R}_{add} \leftarrow \mathscr{R}_{OMP} \cup \{\mathscr{R}_{m_j} \mid m_j \in \mathscr{M}_{curr}\}$
16: **for** $r \in \mathscr{R}_{add}$ **do**
17: $\tau_{min,r} = \min(\tau_s \cup \{\tau_{s,m_j} \mid m_j \in \mathscr{M}_{curr}$ and $r \in \mathscr{R}_{m_j}\})$
18: **end for**
19: $\mathscr{M}_{curr} \leftarrow \{m_k \in \mathscr{M} \mid \exists\, r \in \mathscr{R}_{m_k} \cap \mathscr{R}_{add}$ and $\tau_{f,r,m_k} \ge \tau_{min,r}\}$
20: // Merge CHOPs that contain two or more common robots with R_{add}
21: $\mathscr{R}_{add} \leftarrow \{\mathscr{R}_{m_j} \mid m_j \in \mathscr{M}_{curr}\}$
22: $\mathscr{M}_{curr} \leftarrow \{m_k \in \mathscr{M} \mid |\mathscr{R}_{add} \cap \mathscr{R}_{m_k}| \ge 2\}$
23: // If any additional robots or CHOPs were identified to be merged, iterate again
24: **if** $\mathscr{R}_{OMP} \ne \emptyset$ or $\mathscr{M}_{curr} \setminus \mathscr{M}_{add} \ne \emptyset$ **then**
25: $\mathscr{M}_{add} \leftarrow \mathscr{M}_{curr}, \mathscr{R}_{add} \leftarrow \{\mathscr{R}_{m_j} \mid m_j \in \mathscr{M}_{curr}\}, t_s = \min(\tau_s \cup \{\tau_{s,m_j} \mid m_j \in \mathscr{M}_{curr}\})$
26: $merge = 1, \mathscr{M}_{curr} = \emptyset$
27: **else**
28: $merge = 0$
29: **end if**
30: **end while**

Algorithm 3 then grows $\mathscr{R}_{add}$ and $\mathscr{M}_{add}$ until a valid CHOP can be constructed. Line 2 indicates that if any robots in $\mathscr{C}$ are executing a CHOP when the IC occurs, their CHOPs must be merged with m_{new}. Lines 5-6 defines the CHOP start time and start positions for the current $\mathscr{R}_{add}$, ensuring that the start positions are always on robots' OMPs. Line 8 creates m_{curr}, the CHOP defined by the current $\mathscr{R}_{add}$. Additional robots and CHOPs are added based on three *merging conditions*:

1. Add robots and CHOPs whose paths intersect m_{curr}'s circle (Lines 10-13), so when moving between intermediate waypoints, robots in m_{curr} will be collision-free, even with robots not in the CHOP. Note we only consider robots' paths, which simplifies this condition to fast line segment-circle intersection tests.
2. Merge CHOPs that will cause conflicting motion plans for robots in $\mathscr{R}_{add}$ (Lines 15-19), so $\mathscr{M}$ will always translate to a valid motion plan.
3. Merge CHOPs that contain two or more common robots with R_{add} (Lines 21-22). This ensures that no two robots will be in the same CHOP more than once, which will help provide algorithm completeness in Section 4.5.

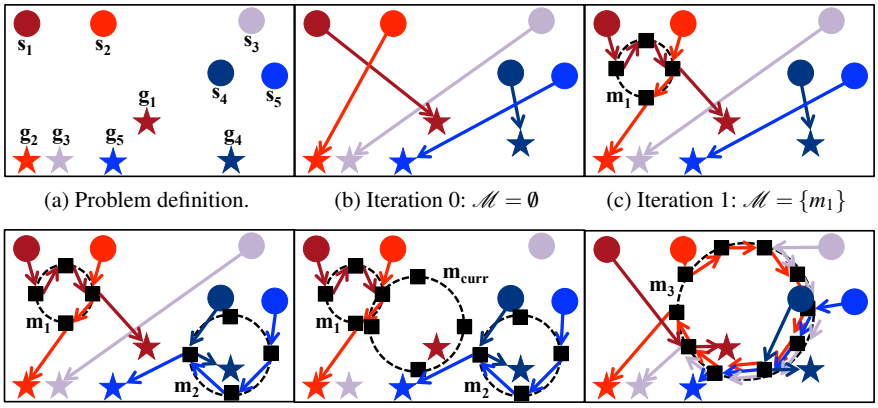
Line 21 adds any new robots to $\mathscr{R}_{add}$ and Line 25 merges any new CHOPs. To merge the CHOPs in $\mathscr{M}_{add}$, their constituent robots are added to $\mathscr{R}_{add}$. If any merged

CHOPs occur before m_{curr}, m_{curr}'s start time is shifted to the earliest start time. We then reconstruct m_{curr} with the updated $\mathscr{R}_{add}$ and iterate again as necessary.

With the returned parameters, we use Algorithm 1 to create the new CHOP, m_{new}, which is added to $\mathscr{M}$. The merged CHOPs in $\mathscr{M}_{add}$ are removed. A new motion plan is computed and the next IC is resolved until the motion plan is collision-free.

4.3.4 Example Problem

Fig. 3 illustrates Algorithms 2 and 3 on the example problem in Fig. 3a.



(a) Problem definition. (b) Iteration 0: $\mathscr{M} = \emptyset$ (c) Iteration 1: $\mathscr{M} = \{m_1\}$

(d) Iteration 2: $\mathscr{M} = \{m_1, m_2\}$ (e) Iteration 3: m_{curr} created, and additional merging is needed. (f) Iteration 3: $\mathscr{M} = \{m_3\}$

Fig. 3: Illustration of Algorithm 2 on the example problem in Fig. 3a. Robots start at circles and must navigate to stars of the same color.

Fig. 3b shows the initial motion plan, where $\mathscr{M} = \emptyset$ and all robots follow their OMPs from their start positions. Figs. 3c-3d shows the motion plans after the first two ICs are resolved. Next, an IC between robots 1 and 3 is detected.

m_{curr} in Fig. 3e represents the initial CHOP created in Line 1 of Algorithm 3, where $\mathscr{R}_{add} = \mathscr{C} = \{1,3\}$. In Lines 12-13, robot 5's OMP's path and robot 2's path in m_1 are found to intersect m_{curr}'s circle. Thus, $\mathscr{R}_{OMP} = \{5\}, \mathscr{M}_{curr} = \{m_1\}$. At Line 15, $\mathscr{R}_{add} = \{1,2,3,5\}$. Evaluating Line 19, m_2 contains robot 5, which is in $\mathscr{R}_{add}$, and $\tau_{f,m_2,5} > \tau_{min,5} = \tau_{s,m_{curr}}$. Thus, m_2 is added to $\mathscr{M}_{curr}$. Lines 21-22 will not change $\mathscr{R}_{add}$ or $\mathscr{M}_{curr}$. Finally, from Line 25, $t_s = \tau_{m_1}, \mathscr{R}_{add} = \{1,2,3,4,5\}$, and $\mathscr{M}_{add} = \{m_1, m_2\}$. No further additions to $\mathscr{R}_{add}$ or $\mathscr{M}_{add}$ are needed.

We create $m_{new} = m_3$ and add it to $\mathscr{M}$, and m_1 and m_2 are removed from $\mathscr{M}$. No other ICs exist. Fig. 3f illustrates the resulting motion plan.

Note Algorithm 2 can be modified to accommodate vehicles with a lower velocity bound, v_{min}, instead of v_{max}. With an additional constraint that a CHOP's intermediate waypoints must be at least $2\sqrt{2}R$ away from its start positions, the minimum length of any synchronized trajectory is $d_{min} = 2\sqrt{2}R$. The maximum length is $d_{max} = \sqrt{2}r_{c,max}$, where $r_{c,max}$ is the radius of a CHOP involving all N robots and contains all goals in G. Thus, running Algorithm 2 with $v_{max} = v_{min}\frac{d_{max}}{d_{min}}$ will ensure that robots will not travel slower than v_{min}.

4.4 Safety

Theorem 1. *Robots are collision-free when executing a CHOP from Algorithm 1.*

Proof. Consider a CHOP $m = \{\{X_i \mid i \in \mathscr{R}_m\}, T, T_{goal}\}$ with final goals X_g. Let $X_s^k = \{X_{i,k-1} \mid i \in \mathscr{R}_{m_k}\}$ denote the positions of robots in $\mathscr{R}_{m_k}$ at t_{k-1} and X_g^k denote the set $\{\mathbf{x}_{next,i} \mid i \in \mathscr{R}_{m_k}\}$. Here, $\mathscr{R}_{m_k}$ and $\mathbf{x}_{next,i}$ are defined as in Eq. 8. We show that robots' trajectories are collision-free for all $k = 2, \ldots, \max_{i \in \mathscr{R}_m} |X_i| + 1$.

We use the CAPT algorithm to assign entry waypoints, so for $k = 2$, when robots move from their CHOP start positions to their entry waypoints, the assignment of goals X_g^k to robots at X_s^k minimizes the total distance squared.

In subsequent intervals, X_s^k contains only intermediate waypoints while X_g^k can contain both intermediate waypoints and goals. Suppose robot $i \in \mathscr{R}_{m_k}$ is moving between intermediate waypoints. Robots enter at every other intermediate waypoint and subsequent rotations are synchronized, so $X_{i,j} \neq X_{j,k-1} \forall j \neq i \in \mathscr{R}_{m_k}$. Thus:

$$\|X_{i,k} - X_{i,k-1}\|_2^2 \leq \|X_{i,k} - X_{j,k-1}\|_2^2 \quad \forall j \neq i \in \mathscr{R}_{m_k} \tag{11}$$

Now, suppose robot i is moving from its exit waypoint to its goal. By design, the exit waypoint is the closest intermediate waypoint to the goal. Thus:

$$\|X_{g,i} - X_{i,k-1}\|_2^2 \leq \|X_{g,i} - X_{j,k-1}\|_2^2 \quad \forall j \neq i \in \mathscr{R}_{m_k} \tag{12}$$

As a result, no alternate assignment of points in X_s^k to those in X_g^k will result in paths with a lower total distance squared than the CHOP's specified assignment. Thus, in each time interval, robots move from their positions in X_s^k to the the one in X_k^g that coincides with the minimum total distance squared assignment.

Line 5 of Algorithm 3 and safety conditions 1 and 2 of Algorithm 1 guarantee positions in X_s^k and X_g^k for all k meet the separation conditions in Eq. 2. The CAPT algorithm guarantees all synchronized trajectories between waypoints are collision-free [14]. Finally, safety condition 3 and the priority sets in Algorithm 1 ensure robots stationary at their goals will not collide with moving robots.

By assigning inter-waypoint motions that match the optimal unlabeled allocation, we inherit the collision avoidance guarantees of the CAPT algorithm. In essence, we use a series of solutions to the unlabeled problem to move towards labeled goals.

4.5 Completeness

Theorem 2. *Algorithm 2 is complete.*

Proof. To be complete, an algorithm must always find a collision-free motion plan in finite time if one exists and indicate that there is no solution when one does not.

From Thm. 1, a CHOP containing all N robots will always be a valid solution. We must additionally show that Algorithm 2 returns a solution in finite iterations.

First note that Algorithm 3 always returns in finite iterations, as there are finite numbers of robots and CHOPs that can be added to $\mathscr{R}_{add}$ and $\mathscr{M}_{add}$, and elements

are never removed. Define $\mathscr{A}$ as the set of *interactions* in $\mathscr{M}$. An interaction is a pair of indices of robots, $\{i, j\}$, such that $i, j \in \mathscr{R}_m$ for some $m \in \mathscr{M}$. For example, in Fig. 3d, $\mathscr{A} = \{\{1,2\},\{4,5\}\}$. When all robots are in a single CHOP, $\mathscr{A} = [\mathscr{I}_N]^2$.

In each iteration of Algorithm 2, either the algorithm terminates, or a new CHOP is added to $\mathscr{M}$. In the latter case, the set of interactions in $\mathscr{A}$ is strictly growing.

To see this, first note that at each iteration, all removed CHOPs have been merged into m_{new}, so interactions are never removed. Alternatively, $\mathscr{A}$ can remain unchanged. This can only occur if $\mathscr{M}_{add}$ contains a single CHOP, m_1, identical to m_{new}. Suppose m_{new} resolves the IC, $\mathscr{C}$. Then, $\mathscr{C} \subseteq \mathscr{R}_{m_{new}} = \mathscr{R}_{m_1}$. m_1 resolves the first IC between robots in $\mathscr{C}$ and guarantees they reach their goals collision-free. Thus, robots in $\mathscr{C}$ can only collide if they abandon their OMPs to enter other CHOPs. Let $\mathscr{M}_{after}$ be the set of CHOPs that robots in $\mathscr{C}$ enter after exiting m_1. CHOPs in $\mathscr{M}_{after}$ fulfill merging condition 2, so $\mathscr{M}_{after} \subset \mathscr{M}_{add}$, and $\mathscr{M}_{add} \neq \{m_1\}$. We have a contradiction, so $\mathscr{A}$ must contain at least one new interaction.

Merging condition 3 guarantees that robots will interact at most once. In finite iterations, $\mathscr{A}$ will contain all unique interactions. This represents the case where all robots are in a single CHOP, which is a collision-free motion plan.

5 Simulation Results

Finally, we examine the algorithm's performance in simulations. Experiments were done on a 2.5 GHz Macbook Pro in MATLAB and C++ Mex, with a maximum algorithm runtime of 10 minutes.

We define a solution's sub-optimality ratio using the total distance of its paths:

$$r_d = \frac{\sum_{i=0}^{N} \int_0^{t_{f,i}} \dot{\gamma}_i(t) dt}{\sum_{i=0}^{N} \|\mathbf{s}_i - \mathbf{g}_i\|_2} \tag{13}$$

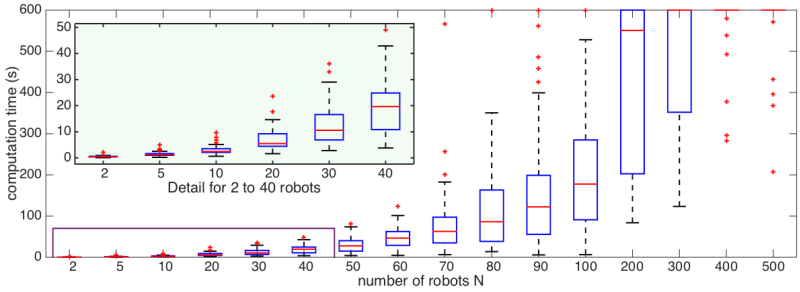
The denominator is an underestimate of the optimal total distance, as for problems like Fig. 1a, the straight-line paths to goals have no collision-free velocity profile.

To detect ICs, we sample trajectories at $dt = \frac{R}{v_{max}}$, where $R = 1, v_{max} = 5$, to ensure no collisions occur between samples. We check for collisions using a spatial hashing algorithm [8] and further eliminate checks for robots moving between intermediate waypoints and between pairs of robots executing the same CHOP.

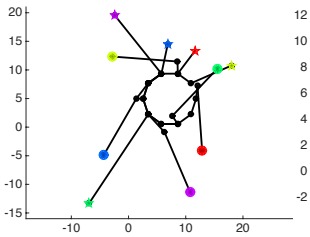
5.1 Variations in Team Size

To examine the effect of the team's size on computation time, we randomly generate case students for 500 robots. We then subsample 400 start to goal assignments from the original set, 300 assignments from the remaining set of 400, and so on.

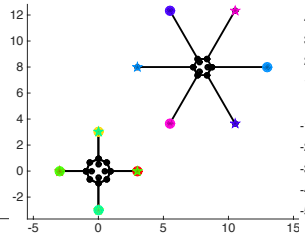
Figs. 4a plots the algorithm computation time for various team sizes. All motion plans for $N \leq 100$ were found in less than 4 minutes. Fig. 7 plots the suboptimality ratios of the solutions, r_d, which is below 7 for all solved problems. Figs. 4b-4d shows the paths of the final motion plan for three example problems.



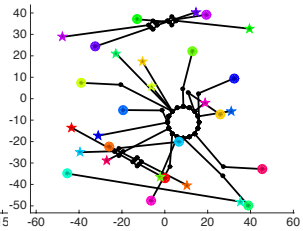
(a) Computation time in average-case settings over 50 trials for each N.



(b) 5 robots with 1 CHOP



(c) 7 robots with 2 CHOPs



(d) 15 robots with 5 CHOPs

Fig. 4: Fig. 4a displays our algorithm's performance over 50 randomly generated case studies. Figs. 4b-4d illustrates the final motion plans for example problems.

5.2 Variations in Problem Density

Next, for a given team size N, we deterministically generate a set of start positions from Halton sequences. These positions are sorted by y-coordinate and stored in S_{init}. For each experiment, we choose a constant D_k and construct the sets $S = D_k S_{init}$ and $G = S + [2R \;\; 0]^T$. Robot $i \in \mathscr{I}_N$ is assigned start position $\mathbf{s}_i = S_i$ and goal $\mathbf{g}_i = G_{\phi_i^{D_k}}$. $\phi_i^{D_k} = i$ for $i \leq \left\lceil \frac{D_k}{D_{k,max}} N \right\rceil$, and $\phi_i^{D_k}$ for other robots are a random permutation of each other's indices. We designate $D_{k,max} = 50$. When $D_k = D_{k,max}$, $\phi_i^{D_k} = i$ for all robots, representing the best-case scenario: robots are sparsely located and their OMPs, a simple translation rightwards, are safe. As D_k decreases, the available free space decreases and the number of collisions increases.

Fig. 5 shows the computation time and Fig. 7 shows the corresponding r_d values over 25 trials for each combination of N and D_k. For small D_k, robots are tightly packed and a single large CHOP will likely be created in a few iterations. Solutions are found quickly, but r_d values are high. As D_k increases, the available free space allows for formation of more local CHOPs, causing smaller deviations from robots' OMPs. This decreases r_d, but increases the computation time. This increase in computation time is more dramatic for larger values of N.

For large D_k, collisions become sparse and fewer CHOPs need to be constructed, decreasing both the computation time and r_d. When $D_k = D_{max}$, no CHOPs need to be created, so the computation time required is small. In short, our algorithm finds solutions quickly for both extremely sparse and dense settings, but requires more computation time when planning many local CHOPs for large teams.

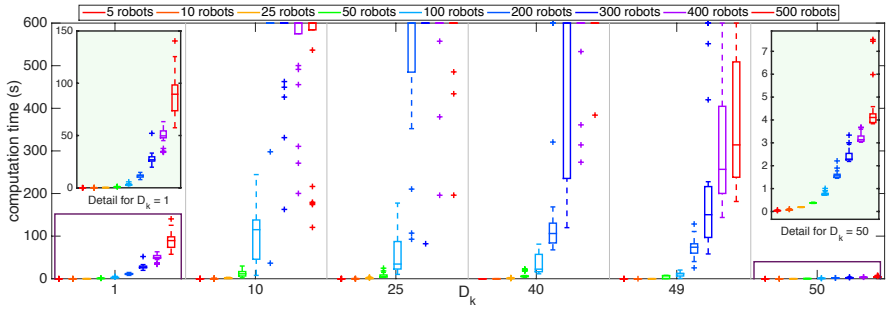


Fig. 5: Computation time over 25 trials for each combination of N and D_k.

5.3 Worst-Case Distributions

We also evaluate the algorithm's performance in the worst-case scenario. For a given N, we find the densest packing of N equally-sized circles in a square that satisfies the separation conditions [12]. We use these circles' centers as both the start and goal positions and generate 50 random assignments for each N. These problems pose the additional challenge that each robot's goal is the start position of another robot.

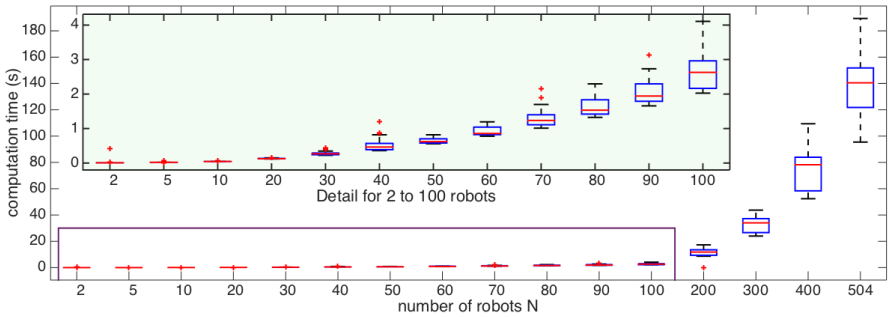


Fig. 6: Computation time in worst-case settings over 50 trials for each N.

Fig. 6 shows we can efficiently solve these problems for $N \leq 504$ in less than 3.5 minutes. Again, once the first collision is found, it is probable that a CHOP containing all N robots will be formed in a only a few iterations. As shown in Fig. 7, r_d becomes rather high for large teams. Nonetheless, we are able to find safe motion plans for teams of hundreds of robots in a challenging environment.

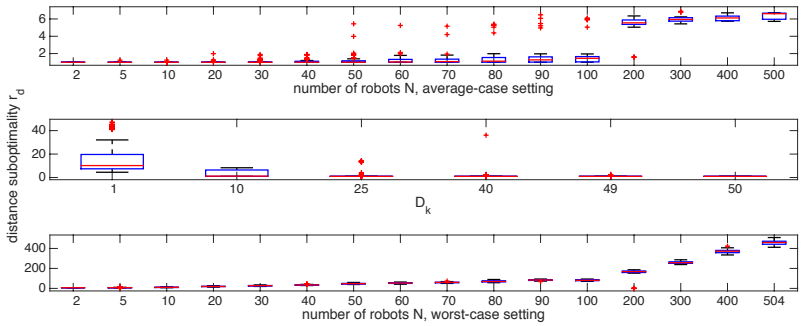


Fig. 7: Suboptimality over all experiments.

5.4 Comparison with Other Multi-Robot Planning Algorithms

Finally, we discuss the performance of our algorithm in comparison with M* with heuristic function inflated by ε [18] and Optimal Reciprocal Collision Avoidance (ORCA) [16]. Table 1 reports the algorithms' performances for a problem generated in Section 5.2 with $N = 10$, $D_k = 1, 5$, and 10.

The M* algorithm builds on A* as an underlying algorithm, searching for optimal paths to goals for each robot in its individual configuration space when robots are collision-free and in a higher-dimensional joint configuration space when they collide. M* retains the completeness and optimality guarantees of A*. In the best-case scenario, M* is extremely fast, as its search space remains low-dimensional. However, its computation time scales up quickly as robots become more densely packed, as the size of the search space grows exponentially with each additional robot in collision. The computation time of our OMP+CHOP algorithm does not scale up as quickly. We note that variants of M* can improve performance, but no results for M*-based algorithms have been reported for problems where $N > 200$ [18].

ORCA is a decentralized, real-time algorithm that, at each time step, assigns each robot a safe velocity based on the observed velocities of its neighbors. The assigned velocity is guaranteed to be collision-free for a known time horizon. We report the total time of the motion plan as the algorithm's planning time, but note that these are different measures. Compared to OMP+CHOP, ORCA's solutions are more optimal. However, in highly dense scenarios, it is possible that a guaranteed safe velocity cannot be found and robots are forced to choose a "best possible" velocity instead. While ORCA has been shown to perform well for large teams in dense settings in practice [16], there are no safety or completeness guarantees.

Table 1: Comparison of performances of multi-robot planning algorithms.

		OMP+CHOP	M* ($\varepsilon = 1.5$)	ORCA [15]
Best case	Planning Time (s)	2.78	0.0020	6.00
$D_k = 10$	Suboptimality Ratio	1.00	1.00	1.00
Average case	Planning Time (s)	2.59	0.027	70.25
$D_k = 5$	Suboptimality Ratio	1.07	1.001	1.001
Worst case	Planning Time (s)	2.65	16.09	23.00
$D_k = 1$	Suboptimality Ratio	5.35	1.11	1.07

6 Conclusions and Future Work

We present the OMP+CHOP algorithm to solve the labeled multi-robot planning problem. This algorithm is scalable while still maintaining safety and completeness guarantees. CHOPs are designed analytically, and no high-dimensional graph searches are required to resolve imminent collisions between robots. This becomes particularly beneficial in densely packed regions or when many robots converge at a single collision point, where other motion planning algorithms reach bottlenecks.

However, we trade off optimality for safety and scalability. In particular, in densely packed problems, the motion plan can be very suboptimal and some robots might circle the CHOP many times before exiting. Immediate directions for future research are applying the algorithm to robots with higher order dynamics and developing a decentralized algorithm requiring only local communication. Future work will also work towards analytically characterizing the algorithm's suboptimality.

Acknowledgements We gratefully acknowledge the support of ONR grants N00014-09-1-1051 and N00014-09-1-103, NSF grant IIS-1426840, and Exyn Technologies. Sarah Tang is supported by NSF Research Fellowship Grant No. DGE-1321851. The authors would also like to thank Levi Cai from the University of Pennsylvania for his implementation of the M* algorithm.

References

1. Buckley S., "Fast motion planning for multiple moving robots," in *Proceedings of the 1989 IEEE International Conference on Robotics and Automation (ICRA)*, 1989, pp. 322–326.
2. de Wilde B., ter Mors A. W., and Witteveen C., "Push and rotate: Cooperative multi-agent path planning," in *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, 2013, pp. 87–94.
3. Erdmann M. and Lozano-Perez T., "On multiple moving objects," *Algorithmica*, vol. 2, pp. 1419–1424, 1986.
4. FAA , "Overview of small uas notice of proposed rulemaking," February 2015.
5. Forbes , "Meet amazon prime air, a delivery-by-aerial-drone project," December 2013.
6. Goldenberg M., Felner A., Stern R., Sharon G., Sturtevant N., Holte R. C., and Schaeffer J., "Enhanced partial expansion A*," *Journal of Artificial Intelligence Research*, vol. 50, no. 1, pp. 141–187, 2014.
7. Hart P. E., Nilsson N. J., and Raphael B., "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, July 1968.
8. Hastings E. J., Mesit J., and Guha R. K., "Optimization of large-scale, real-time simulations by spatial hashing," in *Proceedings of the 2005 Summer Computer Simulation Conference*, 2005, pp. 9–17.
9. Kant K. and Zucker S. W., "Toward efficient trajectory planning: The path-velocity decomposition," *The International Journal of Robotics Research (IJRR)*, vol. 5, no. 3, pp. 72–89, 1986.
10. Kuhn H., "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.
11. Peng J. and Akella S., "Coordinating multiple robots with kinodynamic constraints along specified paths," *The International Journal of Robotics Research (IJRR)*, vol. 24, no. 4, pp. 295–310, 2005.
12. Specht E., "The best known packings of equal circles in a square," October 2013. [Online]. Available: http://hydra.nat.uni-magdeburg.de/packing/csq/csq.html
13. Tomlin C., Pappas G. J., and Sastry S., "Conflict resolution for air traffic management: a study in multi-agent hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, pp. 509–521, 1998.
14. Turpin M., Michael N., and Kumar V., "CAPT: Concurrent assignment and planning of trajectories for multiple robots," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 98–112, 2014.
15. van den Berg J., "RVO2 library documentation," 2008. [Online]. Available: http://gamma.cs.unc.edu/RVO2/documentation/2.0/index.html
16. van den Berg J., Guy S. J., Lin M. C., and Manocha D., "Reciprocal *n*-body collision avoidance," in *The 14th International Symposium on Robotics Research (ISRR)*, 2009, pp. 3–19.
17. van den Berg J., Snoeyink J., Lin M., and Manocha D., "Centralized path planning for multiple robots: Optimal decoupling into sequential plans," in *Proceedings of Robotics: Science and Systems (RSS)*, 2009.
18. Wagner G. and Choset H., "Subdimensional expansion for multirobot path planning," *Artificial Intelligence*, vol. 219, pp. 1–24, 2015.
19. Yu J. and LaValle S. M., "Planning optimal paths for multiple robots on graphs," in *Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 3612–3617.