Safe and Complete Trajectory Generation for Robot Teams with Higher-Order Dynamics

Sarah Tang and Vijay Kumar[1]

***Abstract*— In this work, we consider the labeled multi-robot planning problem. In this paradigm, a team of robots at fixed start positions must navigate to pre-specified and non-interchangable goal positions. While many algorithms have been proposed for finding optimal solutions to this problem, most methods assume that the robots are kinematic agents, whereas in reality, robots often have high-order dynamics that must be respected by their trajectories. Here, we propose a centralized method for generating trajectories for teams of robots with general n^{th}-order dynamics navigating to labeled goals. Our algorithm is safe and complete and additionally allows for decoupled optimization of each robot's trajectory as a Quadratic Program with linear constraints. We present simulation results for teams of up to 20 robots.**

I. Introduction

Recent years have seen a marked increase of interest in using autonomous multi-robot teams to solve complex tasks. Examples of such initiatives include automated warehouses [1] and drone delivery [2]. In these settings, robots are deployed in an obstacle-free space from given start locations and must travel to assigned goal locations to perform some task. This has traditionally been referred to as the *labeled* multi-robot planning problem. This is in contrast with the *unlabeled* planning problem, where goal locations can be freely interchanged between robots, and the k-color planning problem, where goal locations interchangeable between robots within group designations.

Centralized approaches to the labeled planning problem attempt to find solutions by searching the Cartesian product of all robots' state spaces. While methods have been successful in optimizing single-robot path planning algorithms for this higher-dimensional space [3][4], the dimensionality of this joint state space nonetheless grows exponentially with the size of the team and these algorithms quickly become computationally impractical. As a result, other types of centralized approaches have emerged, including rule-based algorithms [5] and Mixed Integer Programming [6]. These approaches are generally optimal, safe, and complete.

However, these algorithms often still cannot be scaled to large teams. In fact, recent results prove that finding optimal solutions to the labeled multi-robot planning problem on undirected graphs is NP-hard [7] [8]. This suggests that an algorithm that is safe and complete, but suboptimal, might be sufficient. In fact, a number of algorithms have proposed splitting the planning problem into multiple centralized subproblems [9] [10] to improve computational efficiency.

[1]S. Tang and V. Kumar are with the Department of Mechanical Engineering and Applied Mechanics. University of Pennsylvania, Phildadelphia, PA 19104, USA {sytang, kumar}@seas.upenn.edu

In addition, planning algorithms often model robots as kinematic agents. In reality, robots are often governed by higher n^{th}-order dynamics. Commanding a dynamic robot to instantaneously change its velocity can result in large errors in trajectory tracking and collisions with neighbors. One way this issue has been addressed is with reactive approaches. For example, van den Berg *et al.* [11] use "Reciprocal Velocity Obstacles" to navigate agents around each other. This method has been extended to arbitrary, non-linear dynamics, however, has no safety or completeness guarantees. Pallottiono *et al.* [12] propose a control policy for decentralized navigation of holonomic robots. However, the control policy is specific to a single, albeit commonly used, dynamic model and not generalizable to higher-order dynamics. Algorithms to safely navigate aircrafts past each other have also used Mixed Integer Programming formulations [13] [14]. However, these methods must concatenate the unknown coefficients of all robots' trajectories into one decision vector and as a result, are often not scalable to larger teams.

In this work, we adopt the terminology "labeled multi-robot trajectory generation problem" to explicitly express that we are concerned with robots' dynamics. In previous work, we introduce OMP_CHOP, a safe and complete centralized labeled planning algorithm for teams of kinematic robots [15]. Here, we extend our previous work to propose a new method to generate smooth, safe, optimal trajectories for robots. In particular, we formulate the trajectory generation problem as a Quadratic Program (QP). Quadratic Programming approaches have been used extensively in the single-robot domain [16] [17] [18] and have been shown to be fast enough for real-time trajectory computation.

Our proposed method has three advantages. First, the resulting trajectories are optimal with respect to robots' dynamics. Next, our algorithm is safe and complete. Finally, unlike previous optimization-based approaches, we do not require joint optimization of all robots' trajectories. Instead, we formulate constraints such that each robot solves a QP that is decoupled from its neighbors.

The remainder of the paper will proceed as follows. Section II formally defines the problem. Section III introduces our approach, which involves a *motion planning* and *trajectory generation* step. Section IV describes the motion planning algorithm, while Section V presents the trajectory generation method. Section VI prove algorithmic guarantees and Section VII analyzes the method's computational complexity. Finally, Section VIII presents simulation results while Section IX discusses our conclusions.

II. Preliminaries

Consider a team of N robots, indexed $i \in [1, N]$, operating in an obstacle-free two-dimensional space. Let $\mathbf{x}^i \in \mathbb{R}^2$ denote the position of robot i. We model robots as disks:

$$\mathcal{B}(\mathbf{x}^i) = \{\mathbf{x} \in \mathbb{R}^2 | \|\mathbf{x} - \mathbf{x}^i\|_2 \leq R\} \tag{1}$$

We assume robots have n^{th}-order dynamics :

$$\frac{d^n}{dt^n}\mathbf{x}^i(t) = \mathbf{u}^i(t) \tag{2}$$

We use subscripted variables $\mathbf{x}^i_x$ and $\mathbf{x}^i_y$ to denote the x and y components of $\mathbf{x}^i$, respectively. Each robot has state:

$$\mathbf{X} \in \mathbb{R}^{2n} = \begin{bmatrix} \mathbf{x}^i & \frac{d}{dt}\mathbf{x}^i & \dots & \frac{d^{n-1}}{dt^{n-1}}\mathbf{x}^i \end{bmatrix}^T \tag{3}$$

Each robot also has a start position $\mathbf{s}^i \in \mathbb{R}^2$ and a fixed goal position $\mathbf{g}^i \in \mathbb{R}^2$. Assume the positions satisfy:

$$\begin{aligned} &\|\mathbf{s}^i - \mathbf{s}^j\|_2 \geq 2\sqrt{2}R \\ &\|\mathbf{g}^i - \mathbf{g}^j\|_2 \geq 2\sqrt{2}R \quad \forall i, j \in [1, N], i \neq j \end{aligned} \tag{4}$$

$\gamma^i(t) : \mathbb{R} \to \mathbb{R}^2$ denotes the *trajectory* of robot i. A set of trajectories $\gamma(t)$ for the team is *safe* if:

$$\mathcal{B}(\gamma^i(t)) \cup \mathcal{B}(\gamma^j(t)) = \emptyset, \forall i, j \in [1, N], i \neq j, t \in [t, t_{max}] \tag{5}$$

Let t^i_f denote the end time of robot i's trajectory and $t_{max} = \max_{i=0}^{N} t^i_f$. Note robots can arrive at their goals at different times. We assume robots disappear from the workspace after they reach their goal. This is analogous to a robot landing or moving into a building to complete its task.

We define the *labeled multi-robot trajectory generation* problem as follows: given a set of start positions $\mathbf{s}$ and fixed goal assignments $\mathbf{g}$, find a set of collision-free trajectories $\gamma(t)$ that navigate each robot from its start to goal position.

We will drop variables' subscripts to refer to the set of variables for the team. For example, $\mathbf{s}$ denotes the set of all start positions.

III. Algorithm Overview

Throughout this paper, we will refer to the problem illustrated in Fig. 1 as a running example. Robots' start positions are represented as circles and their corresponding goal positions are illustrated as stars of the same color.

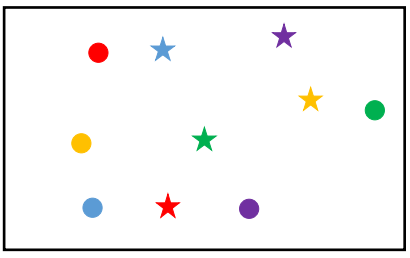
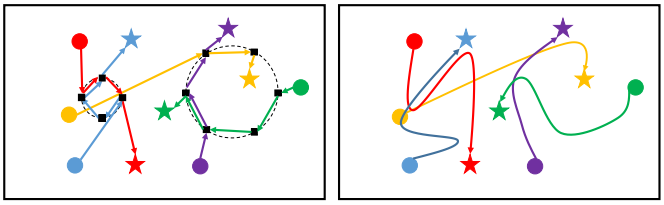


Fig. 1: Example problem. Robots must navigate from start positions, circles, to designated, non-interchangeable goals, stars of the same color.



(a) Motion plans, $\mathcal{M}$. (b) Trajectories, $\gamma(t)$

Fig. 2: Example motion plan and trajectory solutions for the labeled multi-robot trajectory generation problem. Robots must navigate from start positions, circles, to designated, non-interchangeable goals, stars of the same color.

Our proposed algorithm contains two steps. In the *motion planning* step, we find a *motion plan* $\mathcal{M}^i = \{\mathcal{T}^i, \mathbf{x}^i_{des}\}$ for each robot. This motion plan will serve as a set of waypoint constraints from which a safe trajectory can be derived.

We define $\mathcal{T}^i = \{t^i_0, t^i_1, ..., t^i_{m_i}\}$ as a set of times we will refer to as *breakpoint times*. $\mathbf{x}^i_{des} = \{\mathbf{x}^i_{des,0}, \mathbf{x}^i_{des,1}, ..., \mathbf{x}^i_{des,m_i}\}$ is the set of positions, or *waypoints*, at which we want robot i to arrive at times t^j. Let m^i denote the number of elements in robot i's breakpoint times set. Robots do not have to have equally sized breakpoint time sets. Fig. 2a illustrates a motion plan for our example problem. We use straight-line paths to represent the order of waypoints in each robot's motion plan. The construction of $\mathcal{M}$ will be described in Section IV.

In the *trajectory generation* step, we use the information in $\mathcal{M}$ to formulate a Quadratic Program (QP) optimization problem for each robot. The solution to each QP will be the coefficients of a trajectory, $\gamma^i(t)$, for each robot. The set $\gamma(t)$ will be safe, as well as smooth and optimal with respect to the robots' dynamics. Fig. 2b illustrates solution trajectories derived from the motion plan in Fig. 2a. This process will be described in Section V.

IV. Motion Planning

In this section, we present the motion planning step of our algorithm. We first present a solution for kinematic robots, then extend that solution to robots with n^{th}-order dynamics.

A. Kinematic Robots

Our proposed algorithm builds off the OMP_CHOP [15] algorithm. OMP_CHOP, outlined in Algorithm 2, is a cen-

Algorithm 1 $\mathcal{M}$ = OMP_CHOP($\mathbf{s}, \mathbf{g}, R$)

1: **for** $i \in [1, N]$ **do**
2: $\quad \mathcal{T}^i = \{t_0, t_0 + \phi(\mathbf{s}, \mathbf{g})\}$
3: $\quad \mathbf{x}^i_{des} = \{\mathbf{s}, \mathbf{g}\}$
4: $\quad \mathcal{M}^i = \{\mathcal{T}^i, \mathbf{x}^i_{des}\}$
5: **end for**
6: $\psi(t)$ = Get Trajectory($\mathcal{M}$)
7: **while** $\psi(t)$ is not safe **do**
8: $\quad \mathcal{M}$ = Resolve First Collision($\mathbf{s}, \mathbf{g}, \mathcal{M}, \psi(t)$)
9: $\quad \psi(t)$ = Get Trajectory($\mathcal{M}$)
10: **end while**

tralized, safe, and complete algorithm for solving the labeled multi-robot planning problem for kinematic robots.

The algorithm begins by allowing each robot to take a constant-velocity straight-line trajectory to its goal. This is reflected in Lines 1 - 5. Here, ϕ is a heuristic function that, given start and end positions, $\mathbf{x}_0$ and $\mathbf{x}_f$, respectively, will return an estimated time for the robot to travel between the two points. For example, for robots with a maximum velocity, a possible heuristic function is:

$$\phi = \frac{\|\mathbf{x}_0 - \mathbf{x}_f\|_2}{v_{max}}$$

Line 6 translates the each motion plan $\mathcal{M}^i$ into a trajectory ψ^i. A kinematic robot can simply travel at constant velocity between waypoints to arrive at the proper breakpoint time. Dropping superscripts i for simplicity, this trajectory is:

$$\psi(t) = \begin{cases} \mathbf{x}_{des,0} + \frac{t-t_0}{t_1-t_0}(\mathbf{x}_{des,0} - \mathbf{x}_{des,1}) & t_0 \leq t \leq t_1 \\ \mathbf{x}_{des,1} + \frac{t-t_1}{t_2-t_1}(\mathbf{x}_{des,1} - \mathbf{x}_{des,2}) & t_1 \leq t \leq t_2 \\ \dots \\ \mathbf{x}_{des,m-1} + \frac{t-t_{m-1}}{t_m-t_{m-1}}(\mathbf{x}_{des,m} - \mathbf{x}_{des,m-1}) & t_{m-1} \leq t \leq t_m \end{cases} \tag{6}$$

Lines 7 - 10 iteratively resolves collisions between robots' trajectories by constructing *Circular HOlding Patterns* (CHOPs), roundabout-like maneuvers that direct colliding robots safely past each other, and adding them to robots' motion plans. This process continues until the trajectories given by Eq. 6 for the current motion plan are safe.

Fig. 3 illustrates this algorithm for our example problem. Panel 1 shows the initial straight-line trajectories. A collision between the red and blue robots is detected. A CHOP, defined by a set of circularly-distributed intermediate waypoints, is constructed. The CHOP's intermediate waypoints are pictured as black squares in Panel 2. The red and blue robots visit these waypoints synchronously to move past each other.

Panels 3 and 4 illustrate the resolution of the next collision between the purple and green robots. However, as shown in Panel 5, the purple and green robots' CHOP results in a collision between the yellow and purple robots. Thus, in Panel 6, the next iteration refines the solution such that the yellow, purple, and green robots execute a single CHOP.

Algorithm details, as well as safety and completeness proofs, can be found in [15]. The algorithm output is a motion plan $\mathcal{M}$. This motion plan's corresponding trajectory set is safe by construction. We will refer to it as our *nominal solution* and denote it $\gamma_{nom}(t)$.

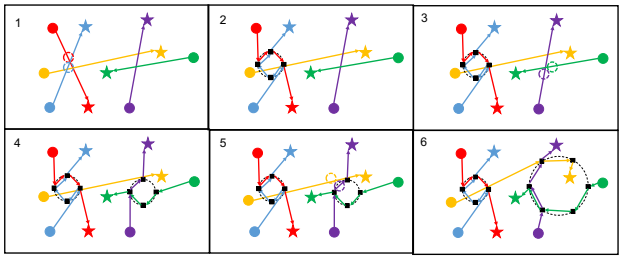


Fig. 3: OMP_CHOP algorithm applied to example problem.

B. General n^{th}-Order Robots

We wish to first extend the OMP_CHOP algorithm to plan for n^{th}-order robots. Consider the problem:

$$\psi(t) = \underset{\psi(t)}{\operatorname{argmin}} \int_0^{t_f} \|\dot{\psi}(t)\|_2^2 dt$$
$$\text{subject to: } \psi(t_0) = \mathbf{x}_0,\ \psi(t_f) = \mathbf{x}_f \tag{7}$$

The solution is the straight-line trajectory:

$$\psi(t) = \mathbf{x}_0 + \left(\frac{t-t_0}{t_f-t_0}\right)(\mathbf{x}_f - \mathbf{x}_0) \tag{8}$$

Eq. 8 gives the trajectories used by kinematic robots in $\gamma_{nom}(t)$ to travel between waypoints in their motion plans.

Now, consider the analogous optimization problem:

$$\psi(t) = \underset{\psi(t)}{\operatorname{argmin}} \int_0^{t_f} \|\psi^{(n)}(t)\|_2^2 dt$$
$$\text{subject to: } \psi(t_0) = \mathbf{x}_0,\ \psi(t_f) = \mathbf{x}_f$$
$$\psi^{(j)}(t_0) = 0,\ \psi^{(j)}(t_f) = 0 \quad \forall j = 1, ..., n-1 \tag{9}$$

Turpin *et al.* [19] show the solution is:

$$\psi(t) = \mathbf{x}_0 + \beta(t)(\mathbf{x}_f - \mathbf{x}_0) \tag{10}$$

$\beta(t)$ is a polynomial $\beta(t) = \sum_{i=0}^{2n-1} \alpha_i t^i$ such that $\beta(0) = 0$ and $\beta(t_f) = 0$. The minimization problem in Eq. 9 contains exactly $2n$ boundary conditions. We can use these to solve for the coefficients of $\beta(t)$. The solution is a straight-line trajectory with a smooth, non-constant velocity profile. We will refer to boundary conditions were the derivatives 1 through $n-1$ are 0 at both endpoints as *homogeneous boundary conditions*.

We can easily apply the OMP_CHOP algorithm to n^{th}-order robots by replacing Eq. 6 with:

$$\psi(t) = \begin{cases} \mathbf{x}_{des,0} + \beta(t)(\mathbf{x}_{des,0} - \mathbf{x}_{des,1}) & t_0 \leq t \leq t_1 \\ \mathbf{x}_{des,1} + \beta(t)(\mathbf{x}_{des,1} - \mathbf{x}_{des,2}) & t_1 \leq t \leq t_2 \\ \dots \\ \mathbf{x}_{des,m-1} + \beta(t)(\mathbf{x}_{des,m} - \mathbf{x}_{des,m-1}) & t_{m-1} \leq t \leq t_m \end{cases} \tag{11}$$

We can replace the heuristic function ϕ to consider limits in higher derivatives. The OMP_CHOP will run as in the kinematic case and return a motion plan, $\mathcal{M}$. Again, Eq. 11 corresponding to $\mathcal{M}$ is a safe nominal solution, $\gamma_{nom}(t)$.

V. Trajectory Generation

While the nominal trajectory set is safe, it is inefficient. The homogenous boundary conditions dictate that each robot must accelerate from and stop at rest when moving between each pair of waypoints. In this section, we present a method to generate smoother safe trajectories from $\mathcal{M}$.

A. Quadratic Program Formulation

From each $\mathcal{M}^i(t)$ and its associated breakpoint set $\mathcal{T}^i$, we can define a *global breakpoint set*, $\mathcal{T}^g = [t_0, t_1, ..., t_m]$. $\mathcal{T}^g$ the sorted union of all times from all $\mathcal{T}^i, i \in [1, N]$. For example, given breakpoint sets $\mathcal{T}^1 = [0, 1, 2]$ and $\mathcal{T}^2 = [0, 1.5, 2, 2.5]$, $\mathcal{T}^g = [0, 1.5, 2, 2.5]$.

For each robot, we wish to find a piecewise polynomial $\gamma^i(t) = [x^i(t) \;\; y^i(t)]^T$, where:

$$x^i(t) = \begin{cases} \sum_{j=0}^{2n-1} c^i_{j,0,x} t^s, & t_0 \le t < t_1 \\ \sum_{j=0}^{2n-1} c^i_{j,1,x} t^s, & t_1 \le t < t_2 \\ \dots \\ \sum_{j=0}^{2n-1} c^i_{j,m-1,x} t^s, & t_{m-1} \le t \le t_m \end{cases} \tag{12}$$

$y^i(t)$ is defined analogously. $\gamma^i(t)$ is optimal with respect to the robots' dynamics if it solves the optimization problem:

$$\gamma^i(t) = \underset{\gamma^i(t)}{\operatorname{argmin}} \int_0^{t_m} \| \frac{d^n}{dt^n} \gamma^i(t) \|_2^2 dt \tag{13}$$

subject to:

1) Waypoint constraints: $\gamma^i(0) = \mathbf{s}^i$, $\gamma^i(t_m) = \mathbf{g}^i$, $\frac{d^k}{dt^k}\gamma^i(0) = 0$, $\frac{d^k}{dt^k}\gamma^i(t_m) = 0 \;\; \forall k = [1, n-1]$.
2) Continuity constraints: $\gamma^i(t)$ is at least $\mathcal{C}^{n-1}$.
3) Collision avoidance constraints: The set $\gamma(t)$ is safe.

Let the decision vector, $\mathbf{c}^i$, contain the unknown trajectory coefficients:

$$\mathbf{c}^i = \big[c^i_{0,0,x} \;\; c^i_{1,0,x} \;\; \dots \;\; c^i_{2n-1,0,x} \;\; c^i_{0,1,x} \;\; c^i_{1,1,x} \;\; \dots \;\; c^i_{2n-1,m-1,x} \;\; c^i_{0,0,y} \;\; c^i_{1,0,y} \;\; \dots \;\; c^i_{2n-1,m-1,y}\big]^T \tag{14}$$

For any n, the cost in Eq. 13 is quadratic:

$$J^i = \int_0^{t_m} \| \frac{d^n}{dt^n} \gamma^i(t) \|_2^2 dt = (\mathbf{c}^i)^T \mathbf{Q} \mathbf{c}^i \tag{15}$$

The waypoint constraints can easily be written in linear matrix form $A_w \mathbf{c}^i = b_w$. To achieve the desired continuity, we impose constraints of the form:

$$\frac{d^k}{dt^k} \sum_{j=0}^{2n-1} c^i_{j,s,x} t^j_{m-1} = \frac{d^k}{dt^k} \sum_{j=0}^{2n-1} c^i_{j,s+1,x} t^j_{m-1} \tag{16}$$
$$\forall s \in [0, m-2], k \in [0, n-1]$$

These are also linear constraints of the form $A_c \mathbf{c}^i = b_c$. Thus, we can describe all waypoint and continuity constraints using the linear matrix constraint $A_{eq} \mathbf{c}^i = b_{eq}$.

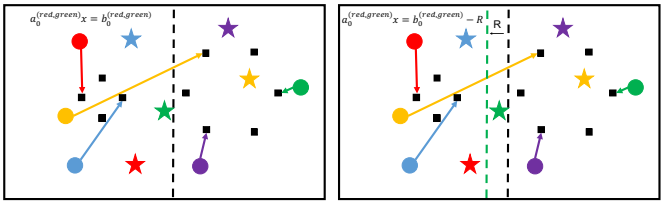
B. Collision-Avoidance Constraints

The set $\gamma(t)$ is safe if:

$$||\gamma^i(t) - \gamma^j(t)||_2 \ge 2R \;\; \forall i, j \in [1, N], i \ne j \tag{17}$$

Eq. 17 is a coupled, non-convex, quadratic equation. We propose an alternate approach to formulate the collision avoidance constraints as decoupled, linear constraints. For each time interval in the global breakpoint set, we:

1) Partition the available free space into N disjoint convex spaces.



(a) Separating hyperplane. (b) Shift of separating hyperplane towards red robot.

Fig. 4: Illustration of construction of constraint hyperplane.

2) Constrain the trajectory of each robot to a designated convex space.

To begin, recall the *Separating Hyperplane Theorem*: for two nonempty, disjoint convex sets, $\exists \mathbf{a}, \mathbf{b}$ such that $\mathbf{a}\mathbf{x} \le \mathbf{b}$ for all points in one set and $\mathbf{a}^T\mathbf{x} \ge \mathbf{b}$ for all points in the other. $\mathbf{a}\mathbf{x} = \mathbf{b}$ is the *separating hyperplane* [20].

Given any time interval $[t_s, t_{s+1}]$, each robot's nominal path is a line segment from $\mathbf{x}^i_{des,t_s}$ to $\mathbf{x}^i_{des,t_{s+1}}$, a convex set in $\mathbb{R}^2$. Assume robots' paths are disjoint. Then, there exists $N-1$ separating hyperplanes, $\mathbf{a}^{(i,j)}_s \mathbf{x} = \mathbf{b}^{(i,j)}_s$, between robot i and its neighbors j. Fig. 4a shows this for the red and green robots during interval 0 of our running example.

We can shift this hyperplane by R towards robot i's path to create a *constraint hyperplane* for robot i. Fig. 4b illustrates this shifted plane, which in our example, is defined by:

$$\mathbf{a}^{(i,j)}_s \mathbf{x} = \mathbf{b}^{(i,j)}_s - R$$

Assume the minimum distance between robot i and j's paths is $d_{min} \ge 2R$. Robot i's path is contained in the half-plane:

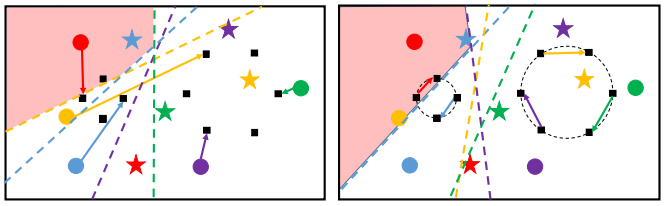
$$\mathbf{a}^{(i,j)}_s \mathbf{x} \le \mathbf{b}^{(i,j)}_s - R \tag{18}$$

and robot j's path is contained in the half-plane:

$$\mathbf{a}^{(i,j)}_s \mathbf{x} \le -(\mathbf{b}^{(i,j)}_s + R) \tag{19}$$

We can find an equation of the form Eq. 18 or Eq. 19 for robot i with respect to each neighbor. The feasible region defined by the intersection of these half-planes will be convex, disjoint from the feasible region of all neighbors, and contain robot i's nominal path. Fig. 5 illustrates two of the red robot's feasible regions for our running example.

To guarantee collision-avoidance without explicit knowledge of neighbors' trajectories, we require each robot's



(a) First time interval. (b) Second time interval.

Fig. 5: Construction of feasible convex regions for red robot.

smoothed trajectory segment to also be within this feasible region. Previous approaches enforced this type of containment constraint for polynomials by enforcing the set of half-plane inequality constraints at sample times along the trajectory. However, this allows the trajectory to move outside the convex region in between samples [16]. Instead, we adopt the method in Flores [18].

Consider the *Bernstein basis* form of a polynomial:

$$\mathbf{p}(t) = \sum_{j=0}^{P} \mathbf{p}_j B_j(t) \tag{20}$$

The basis functions are $B_j(t) = \binom{P}{j} t^j (1-t)^{P-j}$. The coefficients $\mathbf{p}_j = [p_{j,x} \;\; p_{j,y}]^T$ are *control points*. Let $\mathbf{p}^i$ be a vector containing control points for all segments of $\gamma^i(t)$:

$$\mathbf{p}^i = [p_{0,0,x} \;\; p_{0,0,y} \;\; \cdots \;\; p_{2n-1,m-1,x} \;\; p_{2n-1,m-1,y}] \tag{21}$$

The power and Bernstein basis coefficients are related by the constant linear equation $\mathbf{p}^i = T\mathbf{c}^i$.

Bernstein basis curves exhibit a well-known *convex hull properly*: $\mathbf{p}(t) \in \mathbb{R}^2$ will be contained entirely within the convex hull defined by its control points [21]. A trajectory segment can be constrained to a desired convex region by constraining only the control points of its Bernstein basis representation. Eqs. 18 - 19 become linear constraints:

$$\mathbf{a}_s^{(i,j)} T\mathbf{c}^i \leq b_s^{(i,j)} - R \tag{22}$$

$$\mathbf{a}_s^{(i,j)} T\mathbf{c}^i \leq -(b_s^{(i,j)} + R) \tag{23}$$

A trajectory of degree $2n-1$ has $2n$ control points. Thus, at interval s with respect to neighbor j, robot i will have $2n$ inequality constraints. We concatenate the constraints with respect to all neighbor over all time intervals to get the linear matrix inequality $A_{ineq}\mathbf{c}^i \leq b_{ineq}$.

In essence, we partition the free space such that a convex portion is designated to each robot. We then constrain each robot to generate a trajectory that smoothly moves through its dedicated series of convex regions. Each robot's path from $[t_s, t_{s+1}]$ will share an endpoint with its path from $[t_{s+1}, t_{s+2}]$. This ensures the constraint regions for consecutive time intervals will overlap, making a continuous trajectory feasible. In shifting each hyperplane towards each robot, we account for the robots' finite extent. Further, note we do not require trajectories to pass exactly through intermediate waypoints, allowing robots to smooth their trajectories within its feasible region. Finally, note the nominal trajectory set, $\gamma_{nom}(t)$, satisfies all inequality constraints.

C. Full Planning Algorithm

Algorithm 2 presents our complete multi-robot trajectory generation algorithm. It addresses a number of details.

Recall we assumed sufficient separation between robots' paths. However, $\mathcal{M}$ does not guarantee this separation. $\gamma_{nom}(t)$ can contain intervals in which two robots' paths intersect, but do not collide because of their time parameterizations. To address this, Line 4 defines the *state constraint set*, which begins with waypoint and derivative constraints only at the trajectory's endpoints. Line 5 defines the inequality constraint set, which begins empty. For time intervals in which a robot cannot not successfully construct half-plane constraints with all neighbors, we add waypoint and homogeneous boundary conditions (BC) constraints at that path's endpoints to the state constraint set. This dictates that the robot must follow the safe, straight-line nominal trajectory for that time interval.

Further, the QP could fail to have a solution. In this situation, robots travel along their nominal solution, $\gamma^i_{nom}(t)$. Since the nominal solution also satisfies the expected inequality constraints, one robot can default to its nominal solution while its neighbors generate smooth trajectories.

VI. Safety and Completeness Guarantees

The trajectory set returned by Algorithm 2 will always be safe. Consider any pair of robots i and j in any time interval of the global breakpoint set. If both robots follow smooth trajectories generated by their QP, the trajectories must satisfy the collision-avoidance constraints and therefore be safe. If both robots follow their nominal straight-line trajectories, their trajectories are collision-free by virtue of the construction of their motion plans. Suppose robot i follows a

Algorithm 2 γ = Multi-Robot Trajectory Generation($\mathbf{s}, \mathbf{g}$)

1: $\mathcal{M} \leftarrow$ OMP_CHOP($\mathbf{s}, \mathbf{g}, R, n$)
2: $\gamma_{nom}(t) \leftarrow$ nominal solution from $\mathcal{M}$
3: **for all** robots $i \in [1, N]$ **do**
4: $\quad X^i_{des} = \{\gamma^i(0) = \mathbf{s}^i, \gamma^i(t_m) = \mathbf{g}^i$, homogenous BC at 0 and t_m $\}$
5: $\quad [A_{ineq}, b_{ineq}] = \emptyset$
6: $\quad$ **for all** time intervals $[t_s, t_{s+1}]$ in $\mathcal{T}^g$ **do**
7: $\quad\quad ineq\_success =$ TRUE
8: $\quad\quad$ **for all** neighbors $j \in [1, N], j \neq i$ **do**
9: $\quad\quad\quad d_{min} \leftarrow$ minimum distance between paths of robots i and j during $[t_s, t_{s+1}]$
10: $\quad\quad\quad$ **if** $d_{min} < 2R$ **then**
11: $\quad\quad\quad\quad ineq\_success =$ FALSE
12: $\quad\quad\quad$ **end if**
13: $\quad\quad$ **end for**
14: $\quad\quad$ **if** $ineq\_success =$ TRUE **then**
15: $\quad\quad\quad [A_{ineq}, b_{ineq}] \leftarrow$ half-plane constraints for all neighbors
16: $\quad\quad$ **else**
17: $\quad\quad\quad X^i_{des} \leftarrow \{\gamma^i(t_s) = \mathbf{x}^i_{des,s}, \gamma^i(t_{s+1}) = \mathbf{x}^i_{des,s+1}$, homogenous BC at t_s and t_{s+1} $\}$
18: $\quad\quad$ **end if**
19: $\quad$ **end for**
20: $\quad [A_{eq}, b_{eq}] \leftarrow$ waypoint constraints from X^i_{des}, continuity constraints
21: $\quad [success, \gamma^i_{qp}(t)]$ = Solve Quadratic Program
22: $\quad$ **if** success **then**
23: $\quad\quad$ **return** $\gamma^i_{qp}(t)$
24: $\quad$ **else**
25: $\quad\quad$ **return** $\gamma^i_{nom}(t)$
26: $\quad$ **end if**
27: **end for**

smooth trajectory and robot j follows its nominal trajectory. Since robot i was able to successfully smooth its trajectory, it must have been able to find constraint hyperplanes with respect to all its neighbors, including robot j. Thus, robot j must also be able to define a constraint hyperplane with respect to robot i. Robot i's smooth trajectory and robot j's nominal trajectory will both be contained in their respective corresponding half-planes and therefore be safe. This reasoning applies to all pairs of robots across all time intervals, thus, the trajectory set returned by Algorithm 2 is safe.

Further, Algorithm 2 is complete. The OMP_CHOP algorithm is complete and will always return a safe motion plan [15]. Thus, robots will always at least have a safe nominal solution $\gamma^i_{nom}(t)$.

VII. Computational Complexity

In this section, we will focus on the complexity of the trajectory generation QP. Suppose the problem has N robots with n^{th}-order dynamics, the global breakpoint set has m intervals, and half-plane constraints are found at p time intervals. The resulting QP has:

- $4mn$ variables in the decision vector
- $n(m-1)$ continuity constraints
- $2n + (m-p)(2n)$ waypoint constraints
- $(N-1)(2n)(p)$ collision-avoidance constraints

The problem scales with both m and N. The value of m is complicated to quantify. It is determined by the motion plan and increases as robots need to enter more CHOPs to reach their goals. This depends directly on the configuration of the start and goal locations, however, generally speaking, m will increase as N increases. Thus, QP complexity is dictated strongly by the number of robots in the team.

VIII. Experimental Results

We implemented a non-optimized version of our algorithm on a 2.5 GHz Intel Core i7 2015 Macbook Pro in Matlab 2015, using CPLEX as the QP solver. Fig. 6 illustrates two sample problems and their solutions. We use circles to represent start positions and stars of the same color to represent corresponding goals. Figs. 6a and 6c show the nominal solution. Here, black circles represent waypoints in the motion plan; from these, we can see the locations of the constructed CHOPs. Figs. 6b and 6d show the optimal trajectories. In Fig. 6b, the trajectories of the two outer robots are smoothed, however, the two inner robots continue to follow their nominal trajectories. Here, robots have 4^{th} order dynamics and radii of 1 m. In Fig. 6d, all CHOPs become smooth trajectories. Here, $n = 2, R = 0.6$ m.

We further explore the effects of a number of parameters. Table I summarizes the computational performance.

A. Effect of Number of Robots

We first explore the effect of the team's size on computational performance. We generated a set of 20 of start and goal position assignments. We then generated smaller test sets by randomly selecting a subset of assignment pairs from the previous problem. Here, $n = 2$ and $R = 0.6$ m.

Table I presents our results. As expected from our complexity analysis, larger teams demand more computation time. Further, the time required for trajectory generation increases faster than required planning time. This is reasonable, as a larger team will increase the number of collision avoidance constraints at each time interval by a constant, while in the motion planning step, each robot needs to plan CHOPs only with neighbors it collides with. It takes our algorithm about 8 s to solve a problem with 20 robots.

B. Effect of Order of Dynamics

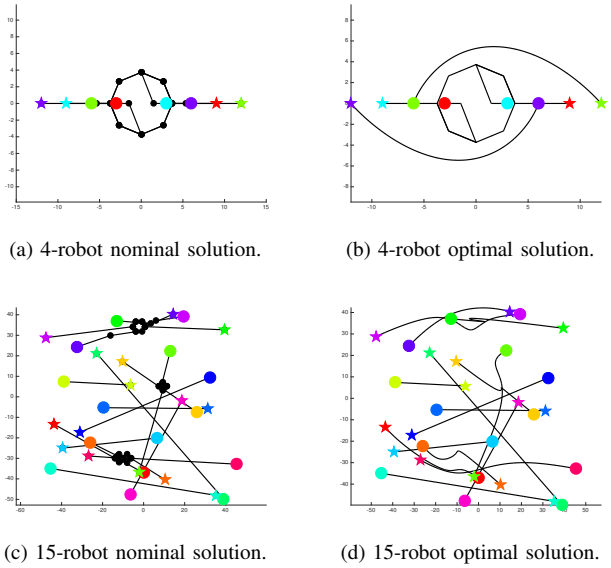
Next, we explore the effects of the order of the robots' dynamics. We solve the 7-robot problem shown in Fig. 7 for robots with 1^{st} through 4^{th} order dynamics, $R = 0.7$ m.

Fig. 7a pictures the paths taken by robot type. The nominal trajectories in each case are identical to the 1^{st}-order robots' trajectories, colored purple. However, after trajectory generation, each robot type takes trajectories that are optimal for their dynamics. Fig. 7b shows the velocity profiles of the red robot for each n. In the 1^{st} order case, velocity is allowed to change instantaneously. However, the velocity profiles become smooth as n increases.

Table I shows that computation time for trajectory generation increases slightly with the order of the robot's dynamics. This is because robots with higher order dynamics are subject to more continuity constraints, increasing the complexity of each robot's QP. However, the computation time for planning remains approximately the same, as the motion plans for the problems are identical.

C. Effect of Problem Density

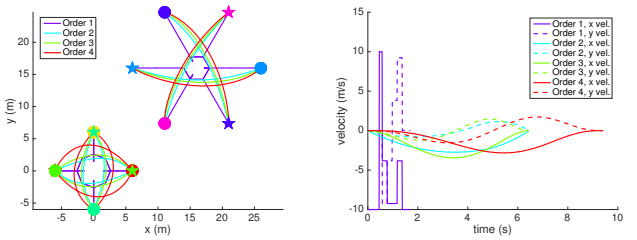
We further test the performance of our algorithm as the problem difficulty increases. We generate a set of start



(a) 4-robot nominal solution. (b) 4-robot optimal solution.

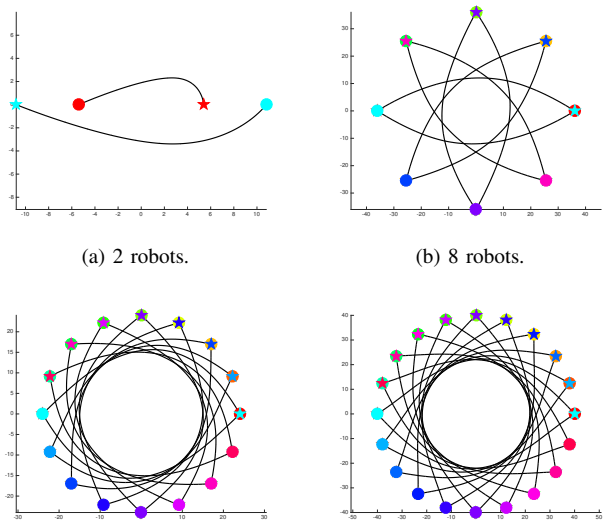
(c) 15-robot nominal solution. (d) 15-robot optimal solution.

Fig. 6: Sample trajectories generated by proposed algorithm. Circles represent start positions and stars of the same color represent the corresponding goals.



(a) Path in xy. (b) Velocity over time.

Fig. 7: Solutions to the same 7-robot problem for different types of robots. "Order" refers to n, the order of the robots' dynamics. Circles represent start positions and stars of the same color represent the corresponding goals.



(a) 2 robots. (b) 8 robots.

(c) 16 robots. (d) 20 robots.

Fig. 8: Solutions to problems where robots collide at a single point. Circles represent start positions and stars of the same color represent the corresponding goals.

positions for 20 robots, $\mathbf{s}_{init}$. We then consider the start positions $\mathbf{s} = D_k \mathbf{s}_{init}$ and nominal goal positions $\mathbf{g}_{nom} = \mathbf{s} + [2R; 0]$, where D_k is a chosen constant. We assign the first $\lceil \frac{D_k}{D_{k,max}} N \rceil$ robots to move directly to the goal $2R$ to their right. These robots will be able to move directly to their goals. The remaining robots are assigned to a random remaining goal. Thus, as D_k decreases, the number of collisions increases and the available free space decreases. Table I lists results for various values of D_k. As expected, "denser" problems were in general more computationally expensive to solve. However, there does not seem to be a clear trend after D_k increases to values greater than 50, as in these scenarios, the amount of free-space available decreases the chances of collision. For these experiments, $n = 3$, $R = 1\ m$, $D_{k,max} = 100$.

As a second experiment, we generated problems where all robots collide at a single choke point. Fig. 8 shows example problems. This scenario often proves computationally difficult for many planning algorithms. Here, $n = 2$, $R = 1$ m.

Fig. 8 shows a number of solutions. In each case, robots are able to successfully circle around each other to reach their goals. We see from Table I that computation time increases for large teams, however again, this increase is mainly in the trajectory generation step as the number of collision avoidance constraints increases.

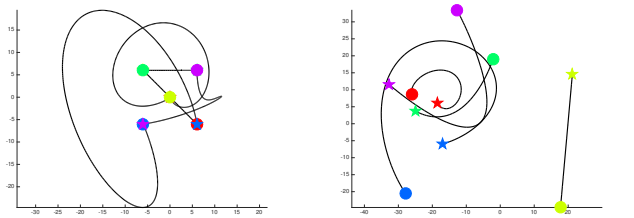
D. Future Work

There are a number of areas in which the algorithm can be improved. As noted by past works [14] [17], this type of QP formulation for trajectory generation can yield trajectories with large excursions if the time allocation between waypoints is poorly chosen. This problem is amplified in the multi-robot planning domain, as all robots must be synchronized in their motions between feasible regions for collision avoidance. As a result, a good time allocation for one robot might cause the trajectory of a neighbor to become suboptimal. Fig. 9a illustrates an example of this, where the blue robot has an extremely long trajectory because there is "too much" time allocated to a trajectory segment. In future work, we hope to develop better time allocation heuristics.

Further, as the number of robots in the team increases, the commercial solver begins to encounter numerical instability. This is caused by both the increase in the size of the decision vector as the number of waypoints in the motion plan

TABLE I: Computational time for various experiments. "Planning" refers to time needed to generate the motion plan, "Traj. Gen." refers to time needed to generate trajectories, and "Total" refers the time for the entire algorithm.

Exp.	Factor	Plan. (s)	Traj. Gen. (s)	Total (s)
Team Size	Num. Robots			
	2	0.26	1.3	1.6
	5	0.29	1.5	1.8
	8	0.32	1.7	2.0
	10	0.81	1.9	2.7
	15	1.2	2.9	4.1
	18	2.0	5.8	7.8
	20	2.3	7.3	9.6
Dynamics	Order			
	1	0.63	1.9	2.5
	2	0.63	1.9	2.5
	3	0.67	2.0	2.7
	4	0.71	2.1	2.7
Density	D_k			
	1	1.09	9.2	10.3
	10	2.8	11.1	13.9
	20	2.2	3.6	5.8
	40	0.91	2.7	3.6
	50	4.4	3.6	8.0
	60	1.2	2.2	3.4
	80	1.5	2.3	3.8
	100	0.49	2.0	2.5
Antipodal	Num. Robots			
	2	0.55	1.3	1.8
	4	0.56	1.7	2.2
	8	0.63	1.9	2.6
	10	0.64	2.3	2.9
	16	0.65	4.2	4.8
	20	0.79	6.8	7.6



(a) A bad time allocation causes long trajectories for blue robot.

(b) Excessively long trajectory for blue robot.

Fig. 9: Examples of suboptimal trajectories. Circles represent start positions and stars of the same color represent the corresponding goals.

increases and the number of collision avoidance constraints. Eliminating redundant inequality constraints could alleviate this problem.

Finally, the trajectory is bound to remain within the specific series of convex regions laid out by the motion plan. This can also result in suboptimal trajectories. For example, in the example problem in Fig. 9b, once the green, red, and purple robots have smoothed out their nominal trajectories into optimal trajectories, the blue robot can easily move directly to its goal. However, because the motion plan includes the blue robot in the CHOP, its trajectory must also circle around its neighbors before arriving at its goal. Situations like this are extremely difficult to detect and avoid, as allowing the blue robot to move directly to its goal would violate the convex region partition required by the collision-avoidance constraints. We hope to explore solutions to these issues in future work.

The most notable advantage of our algorithm is its formulation of the trajectory generation problem as a QP with only linear constraints, whereas previously proposed methods have required integer constraints. It has been shown in single-robot applications that QPs can be used for real-time planning. Further, the decoupling of the QP for each robot suggests this algorithm would be practical for a decentralized system as well. We hope to leverage these capabilities and apply this algorithm for real-time planning for a decentralized multi-robot team.

IX. Conclusion

In this work, we present a novel algorithm to solve the labeled multi-robot trajectory generation problem. Unlike many multi-robot planning algorithms, our method generates optimal trajectories for robots with general n^{th}-order dynamics. Further, our formulation allows decoupling of the trajectory generation problem into separate Quadratic Programs for each robot. Our algorithm is safe and complete. In future work, we hope to improve the quality of our solutions by finding better heuristics for time allocation and eliminating redundant constraints from the QP. We further hope to formulate a decentralized version of this algorithm.

Acknowledgment

We gratefully acknowledge the support of ONR grants N00014-09-1-1051 and N00014-09-1-103, ARL grantW911NF-08-2-0004, ARO grant W911NF-13-1-0350, and Exyn Technologies. Sarah Tang is supported by NSF Research Fellowship Grant No. DGE-1321851.

References

[1] J. J. Enright and P. R. Wurman, "Optimization and coordinated autonomy in mobile fulfillment systems," in *AAAI Conference on Artificial Intelligence*, 2011.

[2] Forbes, "Meet amazon prime air, a delivery-by-aerial-drone project," December 2013.

[3] M. Goldenberg, A. Felner, R. Stern, G. Sharon, N. Sturtevant, R. C. Holte, and J. Schaeffer, "Enhanced partial expansion A*," *Journal of Artificial Intelligence Research*, vol. 50, no. 1, pp. 141–187, 2014.

[4] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning," in *Algorithmic Foundations of Robotics XI - Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2014, pp. 591–607.

[5] R. Luna and K. E. Bekris, "Push and swap: Fast cooperative path-finding with completeness guarantees," in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, 2011, pp. 294–300.

[6] J. Yu and S. M. LaValle, "Planning optimal paths for multiple robots on graphs," in *Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 3612–3617.

[7] J. Yu, "Intractability of optimal multi-robot path planning on planar graphs," *IEEE Robotics and Automation Letters*, 2015.

[8] J. Yu and S. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *The Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013, pp. 1444–1449.

[9] J. van den Berg, J. Snoeyink, M. Lin, and D. Manocha, "Centralized path planning for multiple robots: Optimal decoupling into sequential plans," in *Proceedings of Robotics: Science and Systems (RSS)*, 2009.

[10] J. Yu and D. Rus, "An effective algorithmic framework for near optimal multi-robot path planning," in *The International Symposium on Robotics Research (ISRR)*, 2015.

[11] J. van den Berg, M. C. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

[12] L. Pallottino, V. G. Scordio, E. Frazzoli, and A. Bicchi, "Decentralized cooperative policy for conflict resolution in multi-vehicle systems," *IEEE Transactions on Robotics*, vol. 23, pp. 1170–1183, 2007.

[13] L. Pallottino, E. E. Feron, and A. Bicchi, "Conflict resolution problems for air traffic management systems solved with mixed integer programming," *IEEE Transactions Intelligent Transportation Systems*, vol. 3, pp. 3–11, 2002.

[14] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 477–483.

[15] S. Tang and V. Kumar, "A complete algorithm for generating safe trajectories for multi-robot teams," in *International Symposium on Robotics Research*, 2015.

[16] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 2520 – 2525.

[17] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for quadrotor flight," in *Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[18] M. E. Flores, "Real-time trajectory generation for constrained non-linear dynamical systems using non-uniform rational b-spline basis functions," Ph.D. dissertation, California Institute of Technology, 2008.

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

[20] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[21] L. Piegl and W. Tiller, *The NURBS Book*. Springer Verlag, 1997.