Goal Assignment and Trajectory Planning for Large Teams of Aerial Robots

Matthew Turpin
GRASP Lab
University of Pennsylvania
mturpin@seas.upenn.edu

Kartik Mohta
GRASP Lab
University of Pennsylvania
kmohta@seas.upenn.edu

Nathan Michael
The Robotics Institute
Carnegie Mellon University
nmichael@cmu.edu

Vijay Kumar
GRASP Lab
University of Pennsylvania
kumar@seas.upenn.edu

***Abstract*—This paper presents a computationally tractable, resolution-complete algorithm for generating dynamically feasible trajectories for N interchangeable (identical) aerial robots navigating through cluttered known environments to M goal states. This is achieved by assigning the robots to goal states while concurrently planning the trajectories for all robots. The algorithm minimizes the maximum cost over all robot trajectories. The computational complexity of this algorithm is shown to be cubic in the number of robots, substantially better than the expected exponential complexity associated with planning in the joint state space and the assignment of goals to robots. This algorithm can be used to plan motions and goals for tens of aerial robots, each in a 12-dimensional state space. Finally, experimental trials are conducted with a team of six quadrotor robots navigating in a constrained three-dimensional environment.**

I. Introduction

This paper addresses the problem of simultaneously finding optimal paths and assignments of aerial robots to goals in a setting where robots are identical, and in which it is desirable to minimize the maximum cost over all robot trajectories. This problem is relevant to missions with many tasks that have to be performed as quickly as possible and the tasks can be performed in parallel. In such settings, the cost function may need to reflect the maximum effort over all robots or the the maximum distance traveled by a robot. This is particularly important in first-response and search-and-rescue applications in which a number of robots must visit, for example, all of the rooms in a building. A constraint on the fuel or energy that can be spent on the mission leads quite naturally to a setting where we want to minimize the maximum cost. In such conditions, we are also interested in completeness. We would like the algorithm to find a solution when one exists, even though the solution may not be the optimal solution. Finally, it is necessary to find trajectories that are safe. We want robots to avoid collisions with the environment and with other robots.

Optimal trajectory planning for multiple robots with collision avoidance can be performed by a conventional path planner simply by planning in the joint state space. Of course, the computational complexity typically grows exponentially with the number of robots [2]. One approach studied to counter this growth is to decouple the path planning from the speed at which each path is traversed [2, 5]. Similarly, it is possible to plan trajectories using probabilistic roadmaps and random prioritization [21]. As decoupled approaches are generally not complete, several approaches seek to balance coupled and decoupled formulations in pursuit of completeness with computational tractability [8]. Additionally, computational techniques such as subdimensional expansion [23, 22] can be employed to mitigate increased computational complexity by selectively expanding the higher-dimensional state space as required. Other approaches switch robot positions until robots can easily navigate to goal positions without collision [16, 12], however these can result in extremely lengthy paths.

An alternative strategy is to formulate multi-robot coordination and collision avoidance as reactive control or local coordination problems. Both navigation functions [11] and the decentralized collision avoidance method proposed in [20] enable a robot team to navigate to an assigned set of goals and scale well with the number of agents but lack safety and optimality guarantees for systems with higher-order dynamics.

We approach the problem as one of task allocation as developed in the operational research community in the context of finding an optimal assignment of workers to goals [7] . This has recently been applied to multi-robot task allocation [3] and trajectory planning [6, 10, 18, 9]. As shown in [18, 24], coupling goal assignment with trajectory planning paradoxically reduces the complexity.

This paper, following the authors' previous work [19], considers the problem of simultaneously planning trajectories and goal assignments for idealized interchangeable robots with initial and final deployment states at rest in an environment with static obstacles. In this work (and in departure from [19]), we propose a complete algorithm for Goal Assignment and Planning (GAP) that computes optimal trajectories and assignments of robots to goals with complex vehicle dynamics and cost functions. The performance of the algorithm is studied in simulation for large teams in complex and cluttered environments and experimentally with a team of six quadrotor vehicles. These experimental results demonstrate that the proposed algorithm allows the generation of feasible, safe trajectories for dynamic systems in tightly constrained three-dimensional environments.

II. Preliminaries

Let the set of integers between 1 and z be represented by:

$$\mathcal{I}_z \equiv \{1, 2, \ldots, z\}$$

Let the state of robot i be designated by $x_i \in \mathcal{X}$ where $\mathcal{X}$ is the state space of a single robot. The set of all points in Euclidean space occupied by robot i is represented by the open set $B(x_i) \subset \mathbb{R}^3$. With slight abuse of notation, $B(x(t)) \subset \mathbb{R}^3$ is the set of all points swept out by trajectory $x(t) \in \mathcal{X}$. The initial state of robot $i \in \mathcal{I}_N$ will be designated as $s_i = x_i(0)$. Similarly, g_j specifies the $j \in \mathcal{I}_M$ desired goal state.

The assignment of robots is represented by the mapping $\phi : \mathcal{I}_N \rightarrow \mathcal{I}_M \cup 0$ where each goal can be assigned to a maximum of one robot:

$$\phi_i = \begin{cases} j & \text{if robot } i \text{ is assigned to goal } j \\ 0 & \text{otherwise} \end{cases}$$

A directed graph $G = (V, E)$ is constructed by sampling the state space using either a deterministic or probabilistic planner such as a Probabilistic Roadmap (PRM) [4]. Each vertex $v_i \in V$ represents a valid, collision-free state in $\mathcal{X}$ and V must contain all starting and goal states:

$$s_i \in V \ \forall \ i \in \mathcal{I}_N, \qquad g_j \in V \ \forall \ j \in \mathcal{I}_M \tag{1}$$

An edge $e_{ij} \in E$ corresponds to a collision free trajectory segment $\tau_{ij}(t)$ where $\tau_{ij}(t_0) = v_i$, $\tau_{ij}(t_f) = v_j$ and has cost $C(\tau_{ij}) > 0$. The underlying dynamics of the system will determine how to best compute these segments. Possible cost functions C include distance traveled, energy used, and trajectory duration.

The graph cannot include edges e_{kl} corresponding to trajectories τ_{kl} that can cause collisions with states s_i or g_j, unless one of the vertices of the edge (v_k or v_l) is s_i or g_j, respectively:

$$\begin{aligned} v_k \neq s_i \text{ AND } v_l \neq s_i &\implies B(\tau_{kl}) \cap B(s_i) = \emptyset \\ v_k \neq g_i \text{ AND } v_l \neq g_i &\implies B(\tau_{kl}) \cap B(g_i) = \emptyset \end{aligned} \tag{2}$$

For example, a regular orthogonally-connected grid for spherical robots satisfies Eq. (1) and Eq. (2).

A path in the graph G is represented by an ordered list of the edges in the graph, $P(v_i, v_j) = \{e_{ik}, e_{kl} \ldots, e_{mn}, e_{nj}\}$. The cost of a path in the graph is represented using the norm:

$$||P(v_i, v_j)|| = \sum_{e_{kl} \in P(v_i, v_j)} C(\tau_{kl})$$

The optimal path is that which minimizes the sum of costs of edges between vertices:

$$P^\star(v_i, v_j) = \underset{P(v_i, v_j)}{\arg\min} \quad ||P(v_i, v_j)||$$

The corresponding trajectory for a path from the graph can be constructed by the concatenation of trajectory segments. For example, $\gamma_{ij}(t)$ is the optimal trajectory from s_i to g_j and can be written as the sequence: $\gamma_{ij} = \{\tau_{ik}, \tau_{kl}, \ldots, \tau_{mn}, \tau_{nj}\}$.

Sufficient Conditions for Resolution Completeness: As a result of this algorithm decoupling trajectory generation and time parameterization, there are conditions which must be met to ensure resolution completeness.

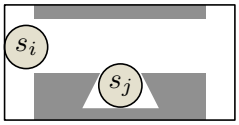


Fig. 1. Example of poor initial conditions for circular robot j which do not satisfy (3). The starting state of robot j has boundaries in free configuration space for which there exists no trajectory from s_j. In this example, robot i is now unable to pass through the passage.

The set of all valid states which share a boundary with vertex v_k can be defined as:

$$\mathcal{V}_k = \{v \big| \text{cl}(B(v)) \cap \text{cl}(B(v_k)) \neq \emptyset, B(v) \cap B(v_k) = \emptyset\}$$

where $\text{cl}(\cdot)$ is the closure of the set.

The following are conservative sufficient conditions for resolution completeness:

$$\exists \tau_{ij} \big| v_i = s_i, \forall v_j \in \mathcal{V}_i, B(\tau_{ij}) \cap B(s_k) = \emptyset \ \forall \ k \in \mathcal{I}_N \setminus i \tag{3}$$

$$\exists \tau_{ij} \big| v_i = g_i, \forall v_j \in \mathcal{V}_i, B(\tau_{ij}) \cap B(g_k) = \emptyset \ \forall \ k \in \mathcal{I}_N \setminus i \tag{4}$$

In words, these state that each robot at its start or goal is able to design a trajectory to any state in the configuration space which shares a boundary with that start or goal, respectively without collision with another start or goal condition. These conditions prevent a robot modifying shape of the configuration space for other robots. See Fig. 1 for an example of initial conditions which violate the assumption in Eq. (3).

III. Algorithm

The Goal Assignment and Planning (GAP) algorithm presented in Algorithm 1 relies heavily upon a number of well-known methods to generate collision-free trajectories for a large number of potentially complex dynamic robots.

Algorithm 1 Goal Assignment and Planning (GAP) Overview

for $i \in \mathcal{I}_N$ **do**
  for $j \in \mathcal{I}_M$ **do**
    Compute $\gamma_{ij}(t)$, optimal trajectory from s_i to g_j
Compute optimal assignment of robots to goals, ϕ^*
for $i \in \mathcal{I}_N$ **do**
  for $j \in \mathcal{I}_N \setminus i$ **do**
    if $s_i \in P^\star(s_j, g_{\phi_j^\star})$ **then**
      Assign partial order $\mathcal{P}$: $j \succ i$
    if $g_i \in P^\star(s_j, g_{\phi_j^\star})$ **then**
      Assign partial order $\mathcal{P}$: $i \succ j$
Construct suitable total ordering ψ from partial ordering $\mathcal{P}$
Optional refinement of $\gamma_{i\phi_i^\star}(t)$. See Sect. V
for $i = 1 \rightarrow N$ **do**
  Compute $\hat{t}_{\psi_i}$, time offset of robot with priority i
return trajectories $\gamma_{i\phi_i^\star}(t - \hat{t}_{\psi_i})$

The generation of a graph G, which has the dimensionality of that of a single robot, is specified in Sect. II and must be performed before the algorithm begins. The method next relaxes inter-agent collision conditions to decouple the system into N multi-goal trajectory planning problems, a process which is detailed in Sect. III-A. Sect. III-B describes utilizing the Hungarian algorithm to assign robots to goal states. Next, Sect. III-C introduces the notion of prioritization of robots, which is always possible due to the minimum maximum cost property of the optimal assignment. Finally in Sect. III-D, trajectories satisfying robot dynamics and avoiding collisions with the environment and other robots are computed. Proofs of collision avoidance and the completeness of Algorithm 1 are deferred to Sect. IV.

A. Computing Individual Trajectories

After construction of graph G, the graph is searched for the minimum cost path for all NM combinations of robots and goal states. Dijkstra's algorithm [1] is a natural choice for this graph search as it returns optimal paths in the graph from the start vertex of a robot to all other vertices.

After the optimal path through the graph is found, full robot trajectories can be constructed from the trajectory segments τ_{kl} corresponding to $e_{kl} \in E$ to form $\gamma_{ij}(t)$ for robot i such that $\gamma_{ij}(t_{i,0}) = s_i$ and $\gamma_{ij}(t_{i,f}) = g_j$. Each robot will remain at its starting vertex when $t < t_{i,0}$ and its goal vertex if $t > t_{i,f}$.

If the graph search finds that goal g_j is unreachable from start location s_i, the cost is infinite $C(\gamma_{ij}) = \infty$.

B. Assignment

The optimal assignment is defined as the one that minimizes the p-norm of the costs incurred by the team:

$$\phi^* \equiv \underset{\phi}{\text{argmin}} \quad \left(\sum_{i \in \mathcal{I}_N} ||P(s_i, g_{\phi_i})||^p \right)^{\frac{1}{p}} \tag{5}$$

where p is sufficiently large such that the following minimum-maximum cost principle holds:

$$\begin{aligned} &\max \left(||P^\star(s_i, g_{\phi_i^\star})||, ||P^\star(s_j, g_{\phi_j^\star})|| \right) \\ &\leq \max \left(||P^\star(s_i, g_{\phi_j^\star})||, ||P^\star(s_j, g_{\phi_i^\star})|| \right) \end{aligned} \tag{6}$$

This means any pairwise exchange of assignment between robots i and j will not result in the maximum cost of the trajectory decreasing. This is a variant of the task assignment problem and can be solved using the well-known Hungarian algorithm [7] with bounded computational complexity of $\mathcal{O}(\max(N, M)^3)$.

The cost of a path to g_0 is defined as a very large constant, Λ, as g_0 corresponds to a robot not being assigned to a goal:

$$||P(s_i, g_0)|| = \Lambda, \quad \Lambda > \sum_{e_{ij} \in E} C(\tau_{ij})$$

This choice of cost will force as many robots as possible to navigate to a goal state.

C. Prioritization

The prioritization stage induces an ordering among all of the robots:

$$s_i \in P^\star(s_j, g_{\phi_j^\star}) \quad \Longrightarrow \quad i \prec j \qquad \forall i \neq j \tag{7}$$

$$g_i \in P^\star(s_j, g_{\phi_j^\star}) \quad \Longrightarrow \quad i \succ j \qquad \forall i \neq j \tag{8}$$

These relations induce a partial ordering $\mathcal{P}$. The proof that this is in fact a partial ordering is left to the reader as it is similar to that of the partial order in [19].

The next step is to construct a total ordering which respects all relations of the partial ordering $\mathcal{P}$. In the case of an ambiguity in the partial ordering, the cost of the trajectories can be used as a tie breaker when constructing this total ordering. This total ordering can then be represented by the mapping $\psi : \mathcal{I}_N \to \mathcal{I}_N$ where ψ_i identifies the robot with priority i.

D. Parameterization

To summarize up to this point: feasible trajectories for each robot have been computed for every robot-goal pair (γ_{ij}) and robots have either been assigned a goal destination $\phi_i^\star = j \in \mathcal{I}_M$ or will remain stationary $\phi_i^\star = 0$, and the robots have been given a prioritization order (ψ_i). The trajectories $\gamma_{i\phi_i^\star}(t)$ will navigate the robot from the initial state s_i to the final assigned state $g_{\phi_i^\star}$, however at this point, there may be collisions between robots. We propose a simple scheme that essentially entails a reparametrization of the planned trajectories. This section outlines a basic method which relies on the resting boundary conditions assumption.

Robots are systematically assigned time offsets, $\hat{t} \geq 0$, to avoid collision with robots of higher priority. The robot with highest priority (ψ_1) begins with an offset of $\hat{t}_{\psi_1} = 0$. Then, the robot with the second highest priority (ψ_2) computes an offset such that its trajectory never collides with the robot with highest priority. It is usually best to minimize the offset times:

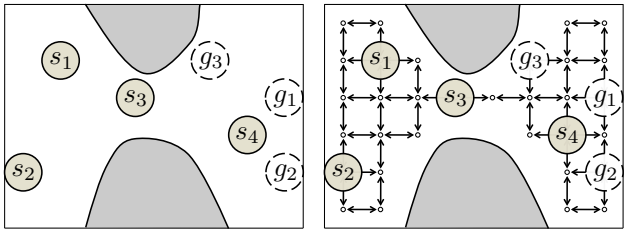
$$\hat{t}_{\psi_2} = \underset{\hat{t}_{\psi_2}}{\arg\min} \; B(\gamma_{\psi_2,\phi_{\psi_2}}(t - \hat{t}_{\psi_2}))) \cap B(\gamma_{\psi_1,\phi_{\psi_1}}(t - \hat{t}_{\psi_1})) = \emptyset$$

Similarly, the robot with the third highest priority then finds the minimum time offset which results in a collision free trajectory between it and all other robots with higher priority. This continues with each robot requiring knowledge of the full trajectory of all robots with higher priority than itself until all offset times are computed, $\hat{t}_i \; \forall \; i \in \mathcal{I}_N$.

E. Algorithm Applied to Simple Robotic Example

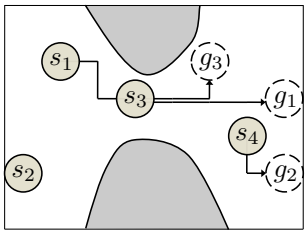
This section considers a team of interchangeable circular kinematic robots with diameter $D = 1$ that can move spatially in 2 dimensions. The state is represented as a 2-dimensional position vector, $x_i \in \mathbb{R}^2$. The graph structure chosen is a regular grid with vertex spacing of D. The trajectories corresponding to edges of the graph are chosen to be straight lines between vertices with constant velocity:

$$\tau_{ij}(t) = \left(\frac{v_i - v_j}{||v_i - v_j||_2} \right) \left(\frac{t - t_0}{t_f - t_0} \right)$$

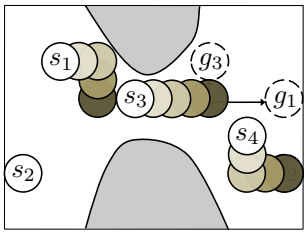


(a) Initial states (s_i), goal states (g_i), and obstacles (gray)

(b) Computation of $G = (V, E)$, trajectories τ shown.



(c) Optimal assignment trajectories ($\phi^\star$)



(d) Snapshots of trajectory tracking at $t = 0.5, 1.0, 1.5, 2.0$

Fig. 2. Key algorithmic steps for a simple circular robotic team of $N = 4$ robots navigating to $M = 3$ goal states.

The robot extent is equivalent to a ball of radius R, or $B(x_i) = x_i + \mathcal{B}_R$. A 4-connected grid using straight line trajectories satisfies the requirements in (3), (4) and (2). The edge costs can be defined as the distance traveled:

$$C(\tau_{ij}) = ||v_i - v_j||_2 = D = 1$$

Now, consider the $N = 4$ robot system with $M = 3$ goals depicted in Fig. 2(a). The graph and corresponding trajectories are constructed making use of the regular grid as shown in Fig. 2(b). Then the optimal path through the graph is planned for each robot to every goal, resulting in 12 trajectories. The trajectory costs returned for each of the paths through the graph in Fig. 2(b) are:

$$\left[\ ||P^\star(s_i, g_j)||\ \right] = \begin{bmatrix} 7 & 9 & 6 \\ 9 & 11 & 8 \\ 4 & 6 & 3 \\ 2 & 2 & 3 \end{bmatrix}$$

The optimal assignment using the Hungarian algorithm is to assign robot 1 to goal 3, robot 3 to goal 1, and robot 4 to goal 2, or $\phi_1^\star = 3, \phi_2^\star = 0, \phi_3^\star = 1, \phi_4^\star = 2$. The trajectories correlating to these assignments are plotted in Fig. 2(c).

Notice how in Fig. 2(c), the starting location of robot 3 is in the path of robot 1, or $s_3 \in P(s_1, g_{\phi_1^\star})$. Therefore, according to in (7), robot 3 has priority over robot 1, or a partial order is induced of $1 \succ 3$. A valid total ordering might be $\{3, 2, 1, 4\}$. In this case, $\psi_1 = 3$, $\psi_2 = 2$, $\psi_3 = 1$, and $\psi_4 = 4$. Now, robot ψ_1, or robot 3 assigns its time offset of $\hat{t}_3 = 0$. Then robot $\psi_2 = 2$ assignes its time offset, which can also be 0. Next, robot $\psi_3 = 1$ assigns its offset time, which can be $\hat{t}_1 = 0$ as well since the trajectory will not lead to a collision even with zero offset. Finally robot $\psi_4 = 4$ plans $\hat{t}_4 = 0$.

IV. Algorithm Properties

Sect. IV-A proves collision avoidance for all robots and Sect. IV-B demonstrates completeness. Finally, Sect. IV-C shows polynomial growth of the complexity bound of the algorithm in the number of boundary conditions.

A. Collision Avoidance

The minor restrictions on allowable vertices in V and on trajectories τ will now be useful in demonstrating collision avoidance in Theorem 1.

Theorem 1. *Algorithm 1 returns trajectories for every robot in the system $\gamma_{i,\phi_i^\star}(t - \hat{t}_i)$, which will task as many robots to goal states as possible without collisions.*

Proof: The conditions in (2) require that in order for a robot to collide with another robot which is stationary at its start or assigned goal state, the moving robot must pass directly through the start or goal vertex of the stationary robot.

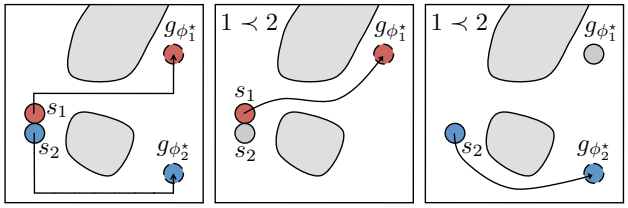
However, a robot passing directly through the start or goal vertex of another robot induces the ordering dictated by (7) or (8). These conditions have been proven in Sect. III-C to generate a partial ordering. Therefore the total ordering ψ which respects all partial orders also ensures conditions on which robots transverse the starting and goal states of other robots. For instance, $s_i \in P^\star(s_j, g_{\phi_j^\star})$ results in $g_i \notin P^\star(s_j, g_{\phi_j^\star})$. Since $\hat{t}$ is computed in the order of priority, robots will always depart the starting location before any other robot needs to enter it. Similarly, robots will never arrive at a goal location before any robots using that vertex have already left it. By induction, each robot will always be able to find a time $\hat{t}$ such that it does not intersect any robots of higher priority. ■

B. Completeness

The assumptions in (3) and (4) specify that a robot at either a start or goal state cannot increase the number of connected components of the configuration space of any other robot. The only vertices in the graph which yield a collision with initial and goal states are the initial and goal states themselves. Any robot which needs to pass through a boundary state of another will only do so if the robot with that start state is already moving as demonstrated in Theorem 1. Therefore the algorithm presented in this paper will preserve the completeness guarantees of Dijkstra's algorithm.

C. Complexity Analysis

This section will analyze the computational complexity of the presented algorithm in terms of number of the robots, N, and number of goals being used, M. Planning individual robot trajectories in Sect. III-A assumes that G is already constructed. To search for the optimal solution using Dijkstra's algorithm using a min priority queue for each start location to all vertices has complexity bound of $\mathcal{O}(|E| + |V|log|V|)$. For typical sampling, $|V|, |E| \gg \max(N, M)$ and therefore constant in N, M. Therefore, finding paths from all starting locations to all goal locations has bounded



(a) (b) (c)

Fig. 3. Optional refinement of trajectories. Figure 3(a) shows optimal initial trajectories and Fig. 3(b) shows trajectory refinement of a higher priority robot. The robot must avoid collisions with the starting states of all robots with lower priority. Figure 3(c) shows trajectory refinement of a lower priority robot. This robot must avoid goal states of higher priority robots.

complexity of $\mathcal{O}(N(|E| + |V|log|V|))$. The Hungarian algorithm can be used to solve the assignment problem in Sect. III-B which has a well-known polynomial complexity bound of $\mathcal{O}(\max(N, M)^3)$. The prioritization scheme designed in Sect. III-C grows with a complexity bound of $\mathcal{O}(N^2)$. Computing time offsets in Sect. III-D grows with complexity bound $\mathcal{O}(N^2)$. Therefore, the worst case complexity of this algorithm is $\mathcal{O}(\max(N, M)^3 + N(|E| + |V|log|V|))$.

V. Optional Trajectory Refinement

While the algorithm presented in Sect. III is very general for dynamic systems, systems with complex dynamics may require trajectory planning in the complete state space which may be very difficult, time consuming, or memory intensive to perform. Therefore, this section presents an optional method to refine trajectories, allowing a lower resolution graph to be constructed or even the use of a lower dimensional robot state.

An assigned trajectory $\gamma_{i\phi_i^\star}$ can be optionally refined to $\tilde{\gamma}_{i\phi_i^\star}$ immediately preceding calculation of time parameterization in Sect. III-D.

There are four requirements for a valid trajectory refinement. First, the refined trajectories must be dynamically feasible and not cause a collision with any obstacle. Next, refined trajectories must satisfy the boundary conditions s_i and g_i. Third, refined trajectories must respect the constraints specified by the computed total ordering ψ:

$$i \prec j \implies B(\tilde{\gamma}_{i\phi_i^\star}) \cap B(g_j) = \emptyset$$
$$i \succ j \implies B(\tilde{\gamma}_{i\phi_i^\star}) \cap B(s_j) = \emptyset$$

Finally, the refinement must decrease the cost of the trajectory for an individual robot:

$$C(\tilde{\gamma}_{i\phi_i^\star}) < C(\gamma_{i\phi_i^\star}) = \sum_{e_{ij} \in P^\star(s_i, g_{\phi_i^\star})} C(\tau_{ij})$$

If these conditions are met, the results from Theorem 1 continue to apply and collision avoidance is guaranteed. The basic steps of trajectory refinement are shown in Fig. 3.

VI. Application to Quadrotor Micro Aerial Vehicle and Experimental Results

The algorithm presented in this paper was implemented on a team of homogeneous quadrotors to navigate a complex maze-like environment. Figure 4 shows the test vehicle with a mass of 75 g used in these experiments. To guide the experiment design, consider a scenario where a team of quadrotors enters a structure, explores a series of chambers connected by narrow passageways, and departs the structure. The structure used for experimentation, shown in Fig. 5, has been designed to be reconfigurable with small passageways. For this experiment, the passageways are configured such that there exist very long internal paths within the structure.

Due to their dynamics, quadrotors require planning in a 12-dimensional state space to fully model the robots' capabilities. Fortunately, the quadrotor has been shown to be differentially flat with flat outputs of position and yaw: $x \in R^3 \times SO(2)$. For the purpose of these experiments, orientation of the quadrotor is not relevant due to symmetry, so the state of the robots is the position vector $x_i \in \mathbb{R}^3$. Minimum snap trajectories have been shown to be very well suited to quadrotors [13, 15] and these trajectories can be generated through a number of waypoints quickly and reliably.

In a closed environment as in these experiments, aerodynamic effects such as ground and inter-vehicle effects are substantial for the quadrotors, particularly between multiple robots in a confined space. Therefore, $B(x)$ is represented by an ellipsoid with elongated vertical dimension to minimize downwash effects on other robots (as proposed in [14]).

The graph, G, is systematically generated by selecting a large number of vertices $v_i \in \mathbb{R}^3$ in and out of the structure which have zero velocity, acceleration, and jerk and respect the conditions (1), (2). The trajectory τ_{ij} is generated by minimizing snap to navigate from one vertex to the next where the cost function used is the integral of snap over the trajectory:

$$C(\tau_{ij}) = \int_{t_o}^{t_f} \dddot{x}_i(t)^2 \, dt$$

Edge e_{ij} is added if $||v_i - v_j||^2$ is below a threshold distance and trajectory τ_{ij} is obstacle collision-free.

Then, optimal individual trajectories are computed by using Dijkstra's algorithm. Optimal assignment $\phi^\star$ and the total ordering ψ are then computed based on the optimal paths, boundary conditions, and path costs. Finally, time offsets are computed in order of priority ψ to ensure collision avoidance of trajectories. Additional time is added to each time offset robots to minimize the effects of lingering aerodynamic turbulence and downwash.

As the quadrotor has homogeneous boundary conditions at every vertex, the original trajectories tend to take a substantial amount of time. Therefore, trajectory refinement is used to complement the complex dynamics of the quadrotor. The zero velocity, acceleration, and jerk boundary conditions are relaxed where possible to allow the quadrotor to make faster progress and expend less energy. In many instances, this results

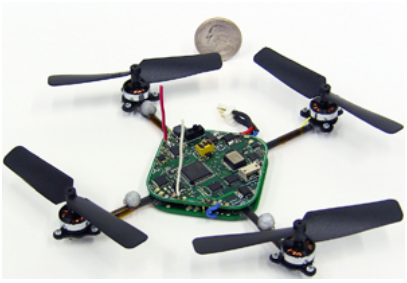


Fig. 4. KMel Robotics NanoQuad quadrotor used in experimentation.

Fig. 5. The structure used for experimentation with 12 chambers.

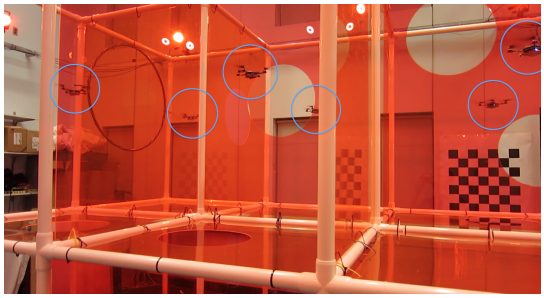


Fig. 6. Six robots flying in the structure used for experimentation.

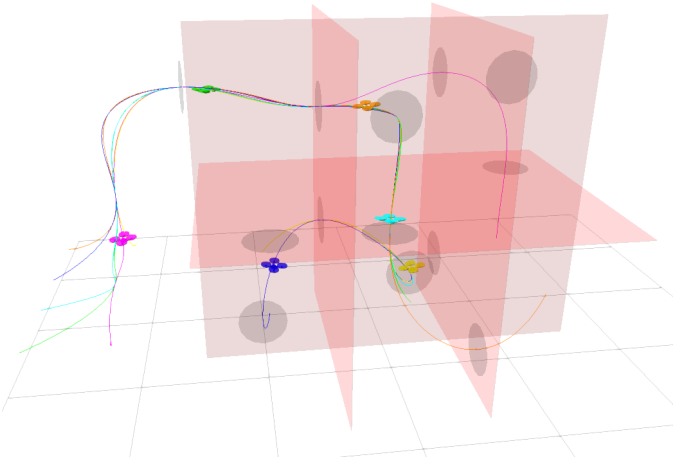


Fig. 7. Visualization of actual experiment with refined minimum snap trajectories being tracked for 6 robots in confined structure. Video available online[1].

in longer distance traveled for each robot, but less energy expended and shorter mission completion times.

One experimental trial begins with 6 quadrotors hovering outside of the structure. Next, 6 goal states are specified, one goal state per chamber on the lower level of the structure. The trajectory for every robot is then computed using GAP (Algorithm 1). Figure 7 shows the trajectories computed, as well as the actual position of the robots in an experiment at one instant. Next, 6 additional goal states are given with one in each chamber of the upper level and those trajectories are computed and then tracked. Finally, 6 goal states are specified on the outside of the structure. See Fig. 6 for an image of the 6 robots arriving at specified waypoints.

Video results of single and multiple robot experiments are available online[1]. This video also includes a simulation of 25 robots navigating a cluttered 3D environment, which was computed in 0.5 seconds.

Figure 8 shows the spherical error probable (SEP) for a single robot tracking the original trajectory, the SEP for a single robot tracking a refined trajectory, and the SEP for 6 robots tracking refined trajectories. These plots clearly demonstrate the quadrotor's improved accuracy tracking these refined trajectories.

Robot states are tracked using a Vicon motion capture system. All experiments are computed on an external computer which is running control code to maintain the quadrotors in flight, as well as computing the trajectories as a separate process. Code is mostly written in MATLAB. Construction of the graph with over 1000 vertices generally takes less than 100 ms. Path planning for $N = 6$ robots reliably takes less than 1 ms. Assignment takes less than 10 ms. Prioritization generally takes 5 ms for 6 robots. Trajectory refinement using minimum snap trajectory generation requires between $5 - 10$ ms per robot, depending on the length of trajectory. Computing time offsets requires a large number of collision checks of ellipsoids, and therefore takes between $10 - 100$ ms for 6 robots depending on the length of trajectories. Thus, the

[1] http://youtu.be/DRJPgOyN2so

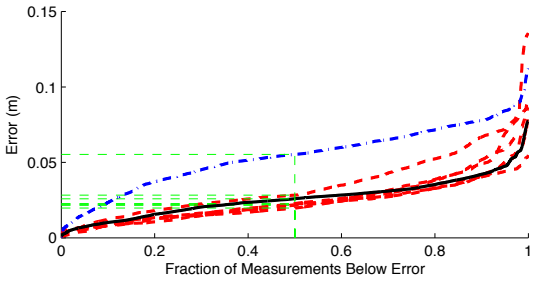


Fig. 8. Spherical Error Probable (SEP) for trials with 1 robot following an unrefined trajectory (blue, dash dot), 1 robot following a refined trajectory (black solid), and 6 robots tracking refined trajectories as a team (red dashed). Notice that the refined trajectories give substantially lower error. Additionally, the initial trajectory takes 6 times longer than a refined trajectory to complete as it does not respect the quadrotor dynamics.

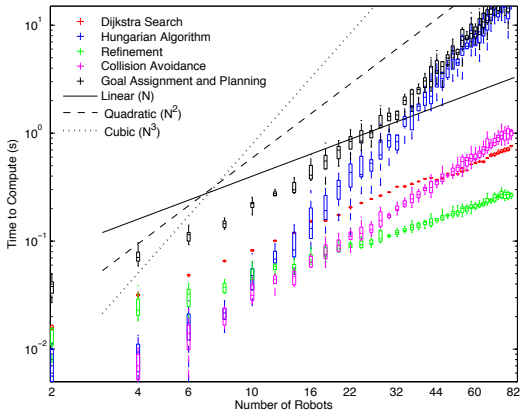


Fig. 9. 100 simulation trials for each value of N to demonstrate time duration spent computing each stage of the algorithm applied to quadrotors. Note that both the planning (red) and refinement (green) grow roughly linearly in the number of robots and collision avoidance (pink) grows roughly quadratically. As the number of robots grows larger, the N^3 complexity of the Hungarian algorithm begins to dominate.

total computation time is typically just under 0.2 s to generate dynamically feasible trajectories for 6 quadrotors through a tight space in MATLAB, however this could be optimized substantially to get better performance.

A much larger environment, similar to the maze-like structure in 5, was modeled in simulation to test the computational requirements as the number of robots grows much larger than the current experimental space allows. This simulated graph has $|V| > 10^4$, $|E| > 6 \times 10^5$, which for $N < 100$ are roughly constant in the number of simulated robots. The times of computation for each stage of the algorithm are displayed in Fig. 9.

This plot confirms the complexity analysis as a function of the number of robots in IV-C, where as the number of robots increases, solving the task assignment grows roughly cubically and begins to dominate the computation time at about 16 robots for the given parameters of the simulation. Of course, as $|V|$ or $|E|$ increase, the Dijkstra search will contribute more, moving the crossover of graph search and task assignment to higher N.

VII. Conclusions and Future Work

This paper presents a resolution-complete tractable trajectory generation method for a team of robots with arbitrarily complex dynamics. The algorithm is shown to provide collision-free trajectories for large numbers of robots. The concept of trajectory refinement allows us to incorporate the dynamics of the robot without substantially increasing the algorithm's complexity. This algorithm is then applied to a team of quadrotor aerial vehicles.

In current work, we are pursuing distributed solutions as well as relexing the assumption of completely interchangeable robots. It also appears this algorithm could be very useful in solving the dynamic vehicle routing problem[17], but would require additional consideration to handle time evolving goal states. Finally, the flexibility of the cost function may be useful in creating algorithms for designing controllers which could be used for exploration with teams of robots.

Acknowledgments

Research supported by: ONR Grant N00014-07-1-0829, ONR MURI Grant N00014-08-1-0696, NSF CCF-1138847, ARL Grant W911NF-08-2-0004, ONR Grant N00014-09-1-1051, Matthew Turpin was supported by NSF Fellowship Grant DGE-0822.

References

[1] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[2] M. Erdmann and T. Lozano-Perez. On multiple moving objects. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 1419–1424. IEEE, 1986.

[3] B.P. Gerkey and M.J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

[4] D. Hsu, L. Kavraki, J.C. Latombe, R. Motwani, S. Sorkin, et al. On finding narrow passages with probabilistic roadmap planners. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, volume 1998, 1998.

[5] K. Kant and S.W. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *The International Journal of Robotics Research*, 5(3):72–89, 1986.

[6] S. Kloder and S. Hutchinson. Path planning for permutation-invariant multirobot formations. *Robotics, IEEE Transactions on*, 22(4):650–665, 2006.

[7] H.W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2): 83–97, 1955.

[8] S.M. LaValle and S.A. Hutchinson. Optimal motion planning for multiple robots having independent goals. *Robotics and Automation, IEEE Transactions on*, 14(6): 912–925, 1998.

[9] L. Liu and D. Shell. A distributable and computation-flexible assignment algorithm: From local task swapping to global optimality. In *Proc. of Robot.: Sci. and Syst.*, Sydney, Australia, July 2012.

[10] L. Liu and D.A. Shell. Multi-level partitioning and distribution of the assignment problem for large-scale multi-robot task allocation. In *In Proc. of Robotics: Science and Systems*, Los Angeles, CA, June 2011.

[11] Savvas G Loizou and Kostas J Kyriakopoulos. Navigation of multiple kinematically constrained robots. *Robotics, IEEE Transactions on*, 24(1):221–231, 2008.

[12] Ryan Luna and Kostas E Bekris. Push and swap: Fast cooperative path-finding with completeness guarantees. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One*, pages 294–300. AAAI Press, 2011.

[13] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Shanghai, China, May 2011.

[14] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar. The GRASP multiple micro UAV testbed. *IEEE Robot. Autom. Mag.*, 17(3):56–65, September 2010.

[15] M. J. Van Nieuwstadt and R. M. Murray. Real-time trajectory generation for differentially flat systems. *Intl. J. Robust and Nonlinear Control*, 8(11):995–1020, December 1998.

[16] Mike Peasgood, Christopher Michael Clark, and John McPhee. A complete and scalable strategy for coordinating multiple robots within roadmaps. *Robotics, IEEE Transactions on*, 24(2):283–292, 2008.

[17] Harilaos N Psaraftis. Dynamic vehicle routing problems. *Vehicle routing: Methods and studies*, 16:223–248, 1988.

[18] M. Turpin, N. Michael, and V. Kumar. Trajectory planning and assignment in multirobot systems. In *Workshop on the Algorithmic Foundations of Robotics*, Boston, MA, June 2012.

[19] M. Turpin, N. Michael, and V. Kumar. Computationally efficient trajectory planning and task assignment for large teams of unlabeled robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, May 2013.

[20] J. Van Den Berg, S. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. *Robotics Research*, pages 3–19, 2011.

[21] J.P. van den Berg and M.H. Overmars. Prioritized motion planning for multiple robots. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 430–435. IEEE, 2005.

[22] G. Wagner, M. Kang, and H. Choset. Probabilistic path planning for multiple robots with subdimensional expansion. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2886–2892. IEEE, 2012.

[23] Glenn Wagner and Howie Choset. M*: A complete multirobot path planning algorithm with performance bounds. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3260–3267, San Francisco, CA, Sept. 2011.

[24] Jingjin Yu and M LaValle. Distance optimal formation control on graphs with a tight convergence time guarantee. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 4023–4028. IEEE, 2012.