Concurrent Goal Assignment and Collision-Free Trajectory Generation for Multiple Aerial Robots

Benjamin Gravell[†] **Tyler Summers**[†]

[†] *University of Texas at Dallas*

Abstract: We develop computationally tractable methods for concurrent goal assignment and planning of collision-free trajectories for multiple aerial robot systems. Our method first assigns robots to goals to minimize total time-in-motion, assuming straight-line maximum-speed trajectories. By coupling the assignment and trajectory generation, the initial motion plans tend to require only limited collision resolution. We then refine the plans by checking for potential collisions and resolving them using either start time delays or altitude assignment. Numerical experiments using both methods show significant reductions in the total time required for agents to arrive at goals with only modest additional computational effort in comparison to state-of-the-art prior work.

1. INTRODUCTION

Autonomous aerial robot teams are emerging as a compelling platform for a variety of application areas, including search-and-rescue, infrastructure inspection, environmental monitoring, and product delivery. A key challenge in safely and effectively deploying large fleets of robots is to minimize the time required for the team to simultaneously complete a set of tasks while ensuring robot-to-robot collisions do not occur. We consider a multi-robot planning problem which requires the assignment of robots in given start locations to given goal locations, along with the generation of minimum-cost trajectories which connect these locations, subject to collision avoidance constraints. As the number of robots increases, so too does the complexity of finding collision-free paths, warranting the development of computationally tractable methods to this end.

Traditional motion planning methods typically rely on discretizing the state space into a graph Švestka and Overmars (1998). Feasible paths are then found through a graph search Dijkstra (1959); Hart et al. (1968); Wang and Goh (2012); Koenig and Likhachev (2002); Koenig et al. (2004); Stentz (1993), or other combinatorial solving methods LaValle (2006). While these methods can in principle solve the multi-agent planning problem, they intrinsically introduce suboptimality and become computationally intractable quickly as the number of agents increases, leading to an exponential growth of the search space dimensionality Erdmann and Lozano-Perez (1986); LaValle (2006). Some methods have been explored which reduce the search space dimensionality LaValle (2006); Wagner et al. (2012); Wagner and Choset (2015), but are unable to sufficiently reduce the complexity for large numbers of agents Turpin et al. (2014). Other centralized planning approaches such as sequential mixed-integer linear optimization Schouwenaars et al. (2001); Richards and How (2002); Alonso-Ayuso et al. (2016), sequential convex programming Augugliaro et al. (2012); Chen et al. (2015), semidefinite programming Frazzoli et al. (2001), or formation space-based velocity planning Kloder and Hutchinson (2006) can work well for relatively small teams but do not scale well to large teams due to the high computational complexity.

An alternative approach to avoiding collisions is to utilize local decentralized feedback control laws to avoid neighboring agents and associated velocity obstacles Warren (1990); Fiorini and Shiller (1998); van den Berg et al. (2008, 2011); Guy et al. (2009, 2010); Cap et al. (2014, 2015). This requires real-time sense-and-avoid capabilities, increasing the system cost and complexity. In general, it is difficult to couple global motion planning with local collision avoidance while maintaining optimal or near-optimal performance and avoiding undesirable deadlock situations.

It was recently shown by Turpin et al. (2014) that when the robots are interchangeable (i.e., it does not matter which robot is assigned to a particular goal so long as all goals receive an assignment), combining the assignment and planning problems considerably facilitates finding collision-free trajectories. They proposed a concurrent assignment and trajectory planning algorithm which tractably gives collision-free trajectories for large robot teams for sufficiently spaced start and goal locations. However, their work optimized a total *squared* distance metric and assumed synchronized robot motion, which results in trajectories that can be significantly suboptimal in terms of total time in motion and may violate minimum velocity constraints associated with certain aerial robots. Here we consider a closely related problem setup and present methods to overcome these limitations.

Our main contributions are as follows. We first define a variation of the trajectory planning problem given by Turpin et al. (2014), then propose a semi-coupled strategy, in which goal assignment and initial trajectory planning

[⋆] The authors are with the Department of Mechanical Engineering at The University of Texas at Dallas, Richardson, TX, 75080 USA. E-mail: `bjgravell@gmail.com, tyler.summers@utdallas.edu` are performed concurrently, followed by a refinement step to resolve potential collisions using either time delays or altitude assignment. By minimizing the total *non-squared* distance and allowing asynchronous trajectories, our algorithm tractably produces collision-free trajectories with near optimal total time in motion. Our method can significantly reduce to total time in motion relative to Turpin et al. (2014), with only modest additional computational expense which becomes negligible for large numbers of agents. Finally, we present numerical experiments that illustrate the effectiveness of our algorithms.

2. PRELIMINARIES

Where applicable, we follow the notation of citeturpin2014. We consider the scenario where N agents begin at N start locations and move towards N goal locations in an n-dimensional Euclidean space. The set of integers between 1 and positive integer Z is denoted by $\mathcal{I}_Z \equiv \{1, 2, \dots, Z\}$, and the $Z \times Z$ identity matrix is denoted by I_Z.

The i^{th} agent center location is given by $\boldsymbol{x}_i \in \mathbb{R}^n, i \in \mathcal{I}_N$, and a ball $\mathcal{B}_R$ centered at $\boldsymbol{x}_i$ of radius R represents the collision zone. The i^{th} start location is given by $\boldsymbol{s}_i \in \mathbb{R}^n, i \in \mathcal{I}_N$. The j^{th} goal location is given by $\boldsymbol{g}_j \in \mathbb{R}^n, j \in \mathcal{I}_N$. The agents operate in an obstacle-free region $\mathcal{K}$, the convex hull of start and goal locations with the Minkowski sum of a ball of radius R: $\mathcal{K} \equiv \text{conv}\left(\{\boldsymbol{s}_i | i \in \mathcal{I}_N\} \cup \{\boldsymbol{g}_j | i \in \mathcal{I}_N\}\right) \oplus \mathcal{B}_R$. We define the $N \times n$ goal matrix as $\boldsymbol{G} = [\boldsymbol{g}_1{}^T, \boldsymbol{g}_2{}^T, \dots, \boldsymbol{g}_N{}^T]^T$. We define the $N \times N$ assignment matrix ϕ, which assigns agents to goals, as

$$\phi_{i,j} = \begin{cases} 1 & \text{if agent } i \text{ is assigned to goal } j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Therefore the ith row of $\phi\boldsymbol{G}$, denoted as $(\phi\boldsymbol{G})_i$, gives the goal location assigned to the ith agent. All agents are assigned to goals in a one-to-one mapping, so

$$\phi^T\phi = I_N. \tag{2}$$

3. TRAJECTORY PLANNING PROBLEM

The trajectory planning problem requires finding N n-dimensional trajectories, which are given agent-wise by

$$\gamma_i(t) : [t_{0,i}, t_{f,i}] \rightarrow \boldsymbol{x}_i, \quad i \in \mathcal{I}_N$$

and must satisfy the initial and terminal conditions

$$\gamma_i(t_{0,i}) = \boldsymbol{s}_i, \quad i \in \mathcal{I}_N, \tag{3}$$

$$\gamma_i(t_{f,i}) = (\phi\boldsymbol{G})_i, \quad i \in \mathcal{I}_N. \tag{4}$$

Each agent is assumed to have simple first order dynamics:

$$\dot{\boldsymbol{x}}_i = \boldsymbol{v}_i \tag{5}$$

and may move with a maximum speed

$$\|\boldsymbol{v}_i\|_2 \leq c_i. \tag{6}$$

The distance between the centers of agents i and j at any given instant is $d_{i,j}(t) = \|\boldsymbol{x}_j(t) - \boldsymbol{x}_i(t)\|_2$. The clearance between the physical extents of agents i and j at any given instant is then

$$e_{i,j}(t) = d_{i,j}(t) - 2R = \|\boldsymbol{x}_j(t) - \boldsymbol{x}_i(t)\|_2 - 2R.$$

We define the global start and end times for which motion may occur over all agents:

$$t_{0,\text{all}} = \min(t_{0,1}, t_{0,2}, \dots, t_{0,N}),$$
$$t_{f,\text{all}} = \max(t_{f,1}, t_{f,2}, \dots, t_{f,N}).$$

We ensure collision avoidance by requiring positive clearance between all agent pairs during the period of possible motion:

$$e_{i,j}(t) > 0 \text{ for } t: \ [t_{0,all}, t_{f,all}], \quad i \neq j \in \mathcal{I}_N. \tag{7}$$

We seek trajectories $\gamma(t) = [\gamma_1(t), ..., \gamma_N(t)]$ that minimize a cost functional:

$$\gamma^*(t) = \underset{\gamma(t)}{\text{argmin}} \sum_{i=1}^{N} \int_{t_{0,i}}^{t_{f,i}} L(\gamma_i(t))dt \quad \text{subject to } (2), (3), (4), (5), (6), (7)$$

Our strategy for finding a solution to this problem proceeds by temporarily ignoring clearance requirements, choosing a suitable cost function to reduce the problem to goal assignment, generating trajectories, then implementing refinement techniques to detect and resolve collisions.

3.1 Goal Assignment

In order to solve the trajectory planning problem, we first choose a meaningful cost function. The C-CAPT algorithm of Turpin et al. Turpin et al. (2014) uses a cost function of the distance traveled squared, but also requires agents to start and arrive at goals at the same time, which limits the speed at which agents may move. This approach also does not easily accommodate agents with heterogeneous speed capabilities. In addition, the choice of a cost function of the distance traveled squared results in sub-optimal assignments with respect to minimizing the total time-in-motion, which for many applications is the true cost of interest. Furthermore, in order to ensure collision avoidance, a sufficient separation spacing between starts and between goals of $2\sqrt{2}R$ must be enforced, which greatly limits the permissible locations.

From a practical standpoint, it is desirable to minimize the task completion time, fuel consumption, and wear, which requires minimizing time-in-motion and allowing each agent to move with the maximum speed available to it. Therefore we choose to use a cost function of the time-in-motion:

$$\underset{\phi,\gamma(t)}{\text{minimize}} \quad \sum_{i=1}^{N} \int_{t_{0,i}}^{t_{f,i}} dt \quad \text{subject to} \quad (2), (3), (4), (5), (6)$$

An argument from the calculus of variations shows that the trajectories which minimize the integral of dt, which is the time-in-motion, are straight lines with constant maximum velocity and which satisfy the boundary conditions. Therefore the the problem reduces to simply finding the optimal assignment and connecting the starts to goals with straight line paths, effectively minimizing the total *non-squared* distance. The optimal assignment is given by

$$\phi^* = \underset{\phi}{\text{argmin}} \sum_{i=1}^{N} \sum_{j=1}^{N} \phi_{i,j} C_{i,j}$$

where entries of the cost matrix C contain the values of the time-in-motion taken by agent i to travel to goal j along a straight line with maximum speed:

$$C_{i,j} = \frac{\|\boldsymbol{g}_j - \boldsymbol{s}_i\|_2}{c_i}, \quad i \in \mathcal{I}_N, j \in \mathcal{I}_N.$$

This is a linear assignment problem which may be solved to optimality using the well-known Hungarian Algorithm Kuhn (1955); Munkres (1957), which runs in $\mathcal{O}(N^3)$ time.

3.2 Trajectory Generation

We define the heading vector and the unit heading vector:

$$\boldsymbol{h}_i = (\phi \boldsymbol{G})_i - \boldsymbol{s}_i, \quad \hat{\boldsymbol{h}}_i = \frac{\boldsymbol{h}_i}{\|\boldsymbol{h}_i\|_2}.$$

Since the optimal trajectories are straight lines between starts and goals with constant maximum velocity, we have

$$\gamma_i^*(t) = (t - t_{0,i}) c_i \hat{\boldsymbol{h}}_i + \boldsymbol{s}_i, \quad i \in \mathcal{I}_N$$

with corresponding velocity vectors $\boldsymbol{v}_i = c_i \hat{\boldsymbol{h}}_i, \quad i \in \mathcal{I}_N$.

A theorem proven via the triangle inequality by Turpin et al. Turpin et al. (2014) reveals that paths almost never intersect using this assignment. For this reason, collisions will be rare for sufficiently spaced agents. This is demonstrated empirically in Section 7 in Fig. 3 by the limited number of altitudes and Fig. 2 by the low number of large time delays required to resolve collisions.

The advantage of allowing asynchronous goal arrival is dependent on the distribution of the start and goal locations; when some trajectory lengths are much larger than others, the ability to move at maximum speed significantly improves speed resource utilization. For many practical applications, such as commercial package delivery, the service area may include goal locations which are both near and far from the start locations.

4. CONSTRAINED COLLISION DETECTION ALGORITHM (CCDA)

The optimal assignment yields a set of minimum time-in-motion trajectories, but it may be required to resolve collisions since collision constraints were ignored. We now develop an algorithm for detecting whether a collision will occur between agents i and j, and will subsequently present two methods to resolve collisions. We use the well-known Closest Point of Approach (CPA) method Muñoz and Narkawicz (2010); Bestaoui-Sebbane (2014) and incorporate terminal conditions at the trajectory bounds, which constrains the time domain. We call this algorithm the Constrained Collision Detection Algorithm (CCDA).

For CCDA and all other subsequent algorithms, we assume that each agent instantly appears before commencing motion from its start location and vanishes upon reaching its goal location, such that no collisions are possible whilst vanished. Physically, this corresponds to a 3D aerial robot taking off instantly to enter a 2D flight altitude and landing instantly upon arriving at the goal location. [1] Thus, collisions will not occur if positive clearance is maintained during periods of mutually occurring motion between all agent pairs:

$$e_{i,j}(t) > 0 \text{ for } t: \ [t_{0,i,j}, t_{f,i,j}], \quad i \neq j \in \mathcal{I}_N$$

where the shared start and end times are defined as

$$t_{0,i,j} = \max(t_{0,i}, t_{0,j}), \quad \text{and} \quad t_{f,i,j} = \min(t_{f,i}, t_{f,j}).$$

[1] It is possible to explicitly incorporate takeoff and landing maneuvers, but we do not in order to simplify the exposition.

For CCDA, we assume that all agents begin motion at the same time so that $t_{0,i} = t_{0,i,j} = t_0, \quad i \neq j \in \mathcal{I}_N$. Thus, collision detection is accomplished by determining whether $e_{i,j}(t) > 0$ for $t: \ [t_0, t_{f,i,j}], \quad i \neq j \in \mathcal{I}_N$. The minimum clearance which occurs during the shared motion is $e_{i,j,\min} \leq e_{i,j}(t)$ for $t: \ [t_0, t_{f,i,j}]$. The time at which $e_{i,j,\min}$ occurs is notated as $t_{\text{crit},i,j}$ so that $e_{i,j}(t_{\text{crit},i,j}) = e_{i,j,\min}$. We define the relative start position and velocity vectors:

$$\boldsymbol{s}_{i,j} = \boldsymbol{s}_j - \boldsymbol{s}_i,$$
$$\boldsymbol{v}_{i,j} = \boldsymbol{v}_j - \boldsymbol{v}_i.$$

We have the following exact collision detection condition.

Theorem 1. Trajectories are collision-free if and only if

$$e_{i,j,\min} = \|(t_{\text{crit},i,j} - t_0)\boldsymbol{v}_{i,j} + \boldsymbol{s}_{i,j}\|_2 - 2R > 0, \qquad i \neq j \in \mathcal{I}_N$$

where

$$t_{\text{crit},i,j} = \begin{cases} t_0 & \text{if } t_0 \geq t_{\text{cpa},i,j} \\ t_{\text{cpa},i,j} & \text{if } t_0 < t_{\text{cpa},i,j} < t_{f,i,j} \\ t_{f,i,j} & \text{if } t_{\text{cpa},i,j} \geq t_{f,i,j} \end{cases}$$

and

$$t_{\text{cpa},i,j} = -\frac{\boldsymbol{s}_{i,j} \cdot \boldsymbol{v}_{i,j}}{\boldsymbol{v}_{i,j} \cdot \boldsymbol{v}_{i,j}} + t_0.$$

Proof: The time of the closest point of approach without time constraints $t_{\text{cpa},i,j}$ is found by minimizing the square of the separation distance $d_{i,j}(t)$. Taking the derivative with respect to time and setting equal to zero yields

$$2\left(t_{\text{cpa},i,j} - t_0\right)\left(\boldsymbol{v}_{i,j} \cdot \boldsymbol{v}_{i,j}\right) + 2(\boldsymbol{s}_{i,j} \cdot \boldsymbol{v}_{i,j}) = 0.$$

By comparing endpoint times to $t_{\text{cpa},i,j}$, the critical time at which $e_{i,j,\min}$ occurs is found, leading to the desired expression. ■

We associate as CCDA output a binary flag matrix $F \in \{0,1\}^{N \times N}$ whose off-diagonal entries are 1 to indicate a collision between two agents and 0 otherwise:

$$F_{i,j} = \begin{cases} 1 & \text{if } e_{i,j,\min} < 0, i \neq j \in \mathcal{I}_N \\ 0 & \text{if } e_{i,j,\min} \geq 0, i \neq j \in \mathcal{I}_N \\ 0 & \text{if } i = j \in \mathcal{I}_N \end{cases}$$

Clearly, trajectories are collision free if and only if $F_{i,j} = 0, \ \forall i,j \in \mathcal{I}_N$. Note also that F is symmetric, since the computation of $e_{i,j,\min}$ is unchanged if the indices i and j are swapped.

The condition of Theorem 1 is *exact*, meaning it only flags agents as colliding when they actually do and does not flag them if they do not. In contrast, the approach of Turpin et al. (2014) stipulates a simple, sufficient (but not necessary) condition for ensuring collision avoidance, namely that starts and goals be separated by a distance of $2\sqrt{2}R$. By comparison, CCDA gives a slightly more complicated, but exact condition for collision avoidance.

Due to symmetry and trivial values on the main diagonal of F, one must perform $\frac{N^2-N}{2}$ computations of $e_{i,j,\min}$ to detect all potential collisions. In our subsequent numerical experiments, we observe solve times which increase as N^2 or N^3 in Section 7.4 in Fig. 5, indicating that our approaches for resolving collisions do not require significantly more computation than assignment and collision detection. Another beneficial property of CCDA is that it is trivially parallelizable, as the result for each agent-pair is independent and may be solved concurrently.

Now that we know which agents will collide, when they will collide, and how much (possibly negative) clearance will exist, we seek to resolve these collisions.

5. COLLISION RESOLUTION VIA ALTITUDE ASSIGNMENT

One way to resolve collisions between agents in the $\mathbb{R}^n$ space is to introduce an additional dimension such that the new global space is $\mathbb{R}^{n+1}$. We can then move agents into discrete parallel $\mathbb{R}^n$ hyperplanes within the $\mathbb{R}^{n+1}$ global space. Thus any agent within one hyperplane cannot collide with any agent in another hyperplane. We will hereafter assume a $\mathbb{R}^3$ global space split into many parallel $\mathbb{R}^2$ planes in which the agents operate. This scenario may be physically realized by confining 3D aerial robots to various horizontal 2D altitudes with sufficient vertical spacing.

Altitude Assignment Algorithm

Once the collision flag matrix F is known, the agents may be assigned to altitudes in such a way that all collisions are eliminated. We develop a simple randomly prioritized sequential algorithm (RPSA) which generates a set of altitude assignments which eliminate all collisions, though it does not guarantee a globally minimum number of altitudes. This algorithm, described in Algorithm 1, yields a $N \times a$ altitude assignment matrix $\mathcal{A}$ which assigns N agents to a altitudes:

$$\mathcal{A}_{i,j} = \begin{cases} 1 & \text{if agent } i \text{ is assigned to altitude } j \\ 0 & \text{otherwise.} \end{cases}$$

The algorithm assigns the first agent to the first altitude, then iterates through the remaining agents and adds a new altitude whenever it is not possible to assign an agent to an existing altitude without a collision. For space considerations, we indicate the end of a conditional statement by a decrease in indentation for all algorithms.

Algorithm 1 RPSA for altitude assignment

```
$a = 1$, $\mathcal{A} = zeros(N,1)$, $\mathcal{A}_{1,1} = 1$
for $i = 2 \to N$ do
    for $j = 1 \to a$ do
        currentAltAgents=$\{k \mid \mathcal{A}(k,j) = 1\}$
        if $F$(currentAltAgents, $i$) $= 0$ then
            $\mathcal{A}_{i,j} = 1$ break
        else
            if $j = a$ then
                $a = a + 1$
                $\mathcal{A} = [\mathcal{A}, zeros(N,1)]$
                $\mathcal{A}_{i,a} = 1$
```

In the worst case, $a = N$, which occurs if every agent collides with every other agent. In the best case $a = 1$, which occurs if no agent collides with any other agent. Our numerical experiments indicate that often only a small number of altitudes are needed even in quite dense scenarios.

6. COLLISION RESOLUTION VIA TIME DELAYS

Another way to resolve collisions between agents is to introduce a delay time $t_{d,i}$ to each agent before beginning motion. For this approach, we restrict motion to a single 2D flight altitude. We also assume that each agent will start on the local ground, remain motionless during the delay time, instantly enter the flight altitude and begin motion, then instantly move to the local ground upon reaching the goal location. In this approach, collisions may be eliminated regardless of start and goal location. As a motivating example, consider a pair of agents which must exchange positions. By introducing sufficient delay time to one agent, the other is allowed to complete its motion before the first enters the flight altitude, avoiding a collision. This illustrates that there always exists a feasible set of time delays, which is easily generated by allowing only a single agent to be in motion at any given time, essentially sending agents one-by-one to their goal locations. The problem then is to find the set of time delays which minimize some objective function, such as the sum of the delay times of each agent, while avoiding collisions as calculated by the collision detection algorithm. This also necessitates modifications to the collision detection algorithm which accommodate the presence of time delays.

6.1 Constrained Collision Detection Algorithm with Delay Times (CCDA-DT)

We develop an algorithm which considers only the time that a pair of agents are both in motion, e.g. if one agent is on the ground then there cannot be a collision. This is equivalent to moving the agent with lower time delay forward along its path by the delay time difference $t_{\text{diff}} = t_{d,j} - t_{d,i}$ to a "virtual" start location, then running CCDA. We call this algorithm the Constrained Collision Detection Algorithm with Delay Times (CCDA-DT), detailed in Algorithm 2.

Algorithm 2 CCDA-DT

```
if $t_{d,i} > t_{d,j}$ then
    if $|t_{\text{diff}}| > (t_{f,j} - t_{0,j})$ then
        $e_{i,j,\min} = +\infty$, $F_{i,j} = 0$ return
    $s'_j = s_j + v_j \, |t_{diff}|$
    $t'_{f,j} = t_{f,j} - |t_{diff}|$
else if $t_{d,i} < t_{d,j}$ then
    if $|t_{\text{diff}}| > (t_{f,i} - t_{0,i})$ then
        $e_{i,j,\min} = +\infty$, $F_{i,j} = 0$ return
    $s'_i = s_i + v_i \, |t_{\text{diff}}|$
    $t'_{f,i} = t_{f,i} - |t_{\text{diff}}|$
CCDA with $s'_i$, $t'_{f,i}$, $i \in \mathcal{I}_N$
```

For cases where the time delays are such that both agents are never moving at the same time (one is on the ground while the other is moving), the effective clearance is infinite. CCDA-DT thus consists of running CCDA using the new virtual start location in place of the actual start location to find the minimum clearance and the presence of a collision.

Optimization Using CCDA-DT as a constraint equation where $e_{i,j,\min}(t_{d,i}, t_{d,j}) \geq 0$ is enforced for each pair of agents, strong nonlinearities are present due to the time domain considerations. Furthermore, the constraint equation will be nonconvex in general, necessitating nontrivial global optimization schemes which are beyond the scope of this work. For the purposes of this paper, we devise a simple heuristic algorithm for finding feasible, suboptimal time delays.

6.2 Randomly Prioritized Sequential Algorithm for Finding Time Delays

We now develop an algorithm, described in Algorithm 3, which works by sequentially searching for small time delays necessary to avoid collisions. This is achieved by iterating through the agents, increasing at fixed increments the time delay on the current agent, and running CCDA-DT against previously set agents after each delay time increase. This process is repeated sequentially, advancing through each agent. This guarantees that eventually, a feasible set of time delays will be found. Though there is no guarantee of global optimality, in practice the time delays found are good, as demonstrated in Fig. 4. We chose to use a time increment $t_{\text{inc}} = A\left(\frac{R}{c_i}\right)$ where $A = 0.1$, which was found to offer a good compromise between optimality and computation time. Running CCDA-DT for agents i and j is denoted as CCDA-DT(i, j).

Algorithm 3 RPSA for time delays

```
$t_{d,i} = 0\ \forall i,\ \ t_{\text{inc}} = A\left(\frac{R}{c_i}\right),\ \ A = 0.1$
**for** $i = 2 \to N$ **do**
    **for** $j = 1 \to N$ **do**
        $F_{i,j} = \text{CCDA-DT}(i, j)$
    **while** $\exists k < i$ such that $F_{i,k} = 1$ **do**
        $t_{d,i} = t_{d,i} + t_{\text{inc}}$
        **for** $j = 1 \to (i-1)$ **do**
            $F_{i,j} = \text{CCDA-DT}(i, j)$
```

7. NUMERICAL EXPERIMENTS

In order to analyze the performance of our algorithms as well as existing algorithms, we used Monte Carlo estimation with start and goal locations generated uniformly at random over an operation area. The operation area is a square of side length $S+2R$ with rounded corners of radius R, so that the agent centers are drawn from a square of side length S. We varied the number of agents N and the area density, defined as

$$\eta = \frac{A_{\text{agents}}}{A_{\text{space}}} = \frac{N\pi R^2}{S^2 + 4RS + \pi R^2}.$$

Fig. 1 visualizes 100 agents with $\eta = 0.4$ and uniform speeds at a time when many agents are in an intermediate state between start and goal locations.

Requiring a minimum separation distance between starts and goals leads to an upper bound on the density, which occurs when the start locations are hexagonally close packed Chang and Wang (2010). For a separation of $2R$ the uppper limit of density is $\frac{\pi}{2\sqrt{3}} \approx 0.9068$ and for a separation of $2\sqrt{2}R$ as in Turpin et al. (2014) the limit is $\frac{\pi}{4\sqrt{3}} \approx 0.4534$. For reference, a typical area density for commercial aircraft traffic management is 10^{-5} air (2016), at which collisions are quite rare. For applications involving many unmanned robots the traffic may be considerably more dense, so we perform simulations for a wide range of densities.

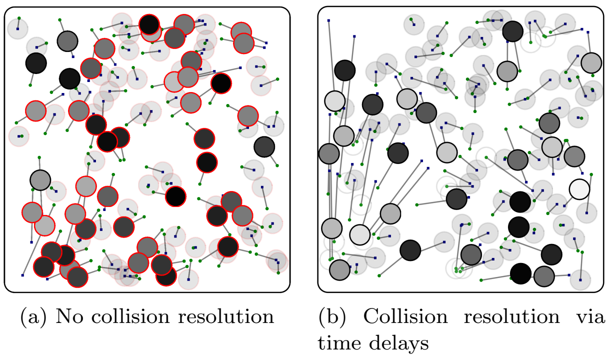
(a) No collision resolution (b) Collision resolution via time delays

Fig. 1. Visualization of 100 agents without (a) and with (b) collision resolution via time delays.

7.1 Time Delay Distribution

Fig. 2 shows a typical histrogram of time delays using our time delay collision resolution method for $N = 1000$ agents and a density of $\eta = 0.1$.This shows that even for start and goal locations which are allowed to be less than $2R$ apart, most agents have zero or low-valued time delays and only a few have large time delays. The small number of large delays and the low average delay is a direct result of coupling the assignment and trajectory generation.

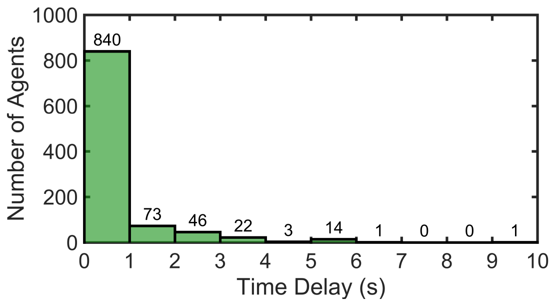

Fig. 2. Histogram of time delays for 1000 agents with $R = 1m$, uniform speed of $c = 1\frac{m}{s}$, and $\eta = 0.1$.

7.2 Altitudes Required

Using our altitude assignment approach, we studied the number of altitudes required to resolve collisions. Fig. 3 shows the empirical average number of altitudes as a function of area density. As expected, the number of altitudes required grew as the density increased, as a result of more potential collisions. Again, we observe that by coupling the assignment and trajectory generation, only a few altitudes are typically required even for highly dense scenarios.

7.3 Normalized Total Time

The time-in-motion and total time of agent i are defined respectively as $t_{m,i} = t_{f,i} - t_{0,i}$ and $t_{t,i} = t_{m,i} + t_{d,i}$. If

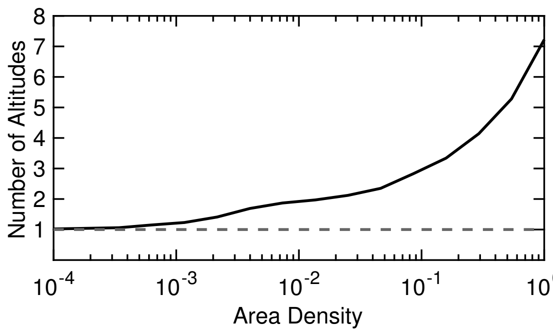

Fig. 3. Number of altitudes required to resolve collisions as a function of area density using 100 agents. Mean value from 100 trials at each of 32 densities between 10^{-4} and 10^{0}.

collisions are resolved via altitude assignment, then there is no need for time delay and consequently the time-in-motion and total time are equivalent. We also define a characteristic time t_c which is the time an agent traveling with average speed would take to traverse the longest straight-line path within the space, which for a square space is simply $\sqrt{2}S$. If all agents travel with uniform speed, then $c_{\text{avg}} = c$ and $t_c = \frac{\sqrt{2}S}{c}$. The normalized average total time is

$$t_{t,\text{avg,norm}} = \frac{t_{t,\text{avg}}}{t_c} = \frac{c\sum_{i=1}^{N}(t_{m,i}+t_{d,i})}{N\sqrt{2}S}.$$

With respect to this metric, plotted in Fig. 4, our altitudes approach gave the best results for all densities. At low densities, our time delay approach gave nearly the same performance as a consequence of small time delays which vanish as the density goes to zero. At higher agent densities, the time delay approach result began to increase as the physical extent of the agents became more influential. Both of our approaches gave better performance at all densities than the Turpin et al. approach. This increase in performance became more apparent when agents had heterogeneous maximum speeds, mainly due to the fact that the Turpin approach does not consider speed and is not suited to handling this situation. Nevertheless it was instructive to observe that the introduction of various maximum speeds did not have a deleterious effect on performance in our approach. We allowed collisions for simulations using the Turpin et al. approach to avoid imposing the $2\sqrt{2}R$ separation condition required for that approach to be collision-free for simplicity in generating the starts and goals.

7.4 Computational Speed

Computation time is critical for scalability to large robot teams in practice. Our simulations were implemented in MATLAB running on a desktop with an Intel i7 6700K quad-core processor running at 4.0GHz. The results are given in Fig. 5, where we observe reasonable computation times for relatively large teams.

At high agent numbers the assignment computations, which grew as N^3, dominated over our collision resolution steps, which grew only as N^2. Assignment required almost exactly the same computational time whether using a

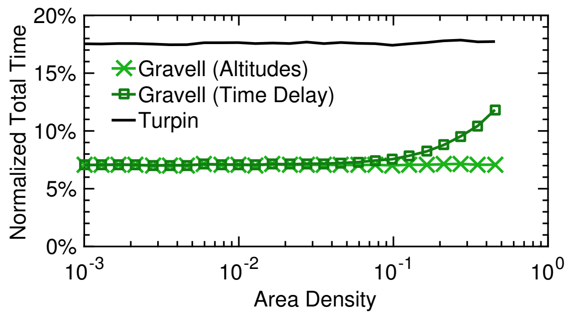

Fig. 4. Normalized average total time as a function of area density using various trajectories for 100 agents with uniform maximum speed. Mean value from 1000 trials at each of 25 area densities.

time-in-motion cost as in our approach (Gravell, no resolution) or a total distance squared approach, as in Turpin et al. (2014).

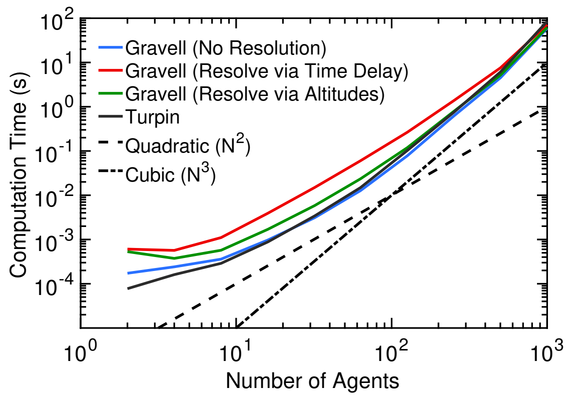

Fig. 5. Computational time as a function of number of agents using various trajectories using an area density of 0.1. Mean value from 10 trials at each of 10 numbers of agents between 2 and 1000.

8. CONCLUSION

We proposed semi-coupled solutions to assignment and collision-free trajectory planning problem for multiple robots. Our assignment of agents to goals minimizes the total time-in-motion, and the resulting straight-line, maximum-speed, asynchronous trajectories take full advantage of possibly heterogeneous speed capabilities. The trajectories were then refined using collision resolution methods involving either altitude assignment or start-time delays. The results of numerical simulations revealed promising decreases in the total time without unworkable increases in the computation time over existing approaches, allowing faster task completion. Our algorithm also allowed eliminated restrictions on start and goal locations as opposed to other methods which require enforcement of a minimum separation.

Future work includes extension to agents with more complex dynamics and/or motion constraints, combining time delays with altitudes, and a parallel implementation to decrease solve times.

REFERENCES

(2016). *Doc 4444 - Procedures for Air Navigation Services - Air Traffic Management*. International Civil Aviation Organization, 16 edition.

Alonso-Ayuso, A., Escudero, L.F., and Martn-Campo, F.J. (2016). Multiobjective optimization for aircraft conflict resolution. a metaheuristic approach. *European Journal of Operational Research*, 248(2), 691 – 702. doi: http://dx.doi.org/10.1016/j.ejor.2015.07.049.

Augugliaro, F., Schoellig, A.P., and D'Andrea, R. (2012). Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1917–1922. IEEE.

Bestaoui-Sebbane, Y. (2014). *Planning and Decision Making for Aerial Robots*, volume 71. Springer International Publishing, Gewerbestrasse 11, 6330 Cham, Switzerland, 1 edition.

Cap, M., Novak, P., Kleiner, A., and Seleck, M. (2015). Prioritized planning algorithms for trajectory coordination of multiple mobile robots. *IEEE Transactions on Automation Science and Engineering*, 12(3), 835–849.

Cap, M., Novak, P., and Kleiner, A. (2014). Finding near-optimal solutions in multi-robot path planning. *CoRR*, abs/1410.5200. URL `http://arxiv.org/abs/1410.5200`.

Chang, H.C. and Wang, L.C. (2010). A Simple Proof of Thue's Theorem on Circle Packing. *ArXiv e-prints*. URL `https://arxiv.org/abs/1009.4322`.

Chen, Y., Cutler, M., and How, J.P. (2015). Decoupled multiagent path planning via incremental sequential convex programming. In *2015 IEEE International Conference on Robotics and Automation*, 5954–5961.

Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. *Numer. Math.*, 1(1), 269–271.

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

Fiorini, P. and Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7), 760–772.

Frazzoli, E., Mao, Z.H., Oh, J.H., and Feron, E. (2001). Resolution of conflicts involving many aircraft via semidefinite programming. *Journal of Guidance, Control, and Dynamics*, 24(1), 79–86.

Guy, S.J., Chhugani, J., Curtis, S., Dubey, P., Lin, M., and Manocha, D. (2010). Pledestrians: A least-effort approach to crowd simulation. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, 119–128. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland.

Guy, S.J., Chhugani, J., Kim, C., Satish, N., Lin, M., Manocha, D., and Dubey, P. (2009). Clearpath: Highly parallel collision avoidance for multi-agent simulation. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, 177–187. ACM, New York, NY, USA.

Hart, P.E., Nilsson, N.J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100–107.

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

Koenig, S. and Likhachev, M. (2002). Incremental A*. In T.G. Dietterich, S. Becker, and Z. Ghahramani (eds.), *Advances in Neural Information Processing Systems 14*, 1539–1546. MIT Press.

Koenig, S., Likhachev, M., and Furcy, D. (2004). Lifelong planning A*. *Artificial Intelligence*, 155(1-2), 93–146.

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

LaValle, S.M. (2006). *Planning Algorithms*. Cambridge University Press, Cambridge, U.K.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1), 32–38.

Muñoz, C.A. and Narkawicz, A.J. (2010). Time of closest approach in three-dimensional airspace. Technical report, National Aeronautics and Space Administration, Langley Research Center, Hampton, Va.

Richards, A. and How, J.P. (2002). Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *American Control Conference*, volume 3, 1936–1941. IEEE.

Schouwenaars, T., De Moor, B., Feron, E., and How, J. (2001). Mixed integer programming for multi-vehicle path planning. In *European Control Conference*, 2603–2608. IEEE.

Stentz, A. (1993). Optimal and efficient path planning for unknown and dynamic environments. *International Journal of Robotics and Automation*, 10, 89–100.

Švestka, P. and Overmars, M.H. (1998). *Probabilistic path planning*, 255–304. Springer Berlin Heidelberg, Berlin, Heidelberg.

Turpin, M., Michael, N., and Kumar, V. (2014). Capt: Concurrent assignment and planning of trajectories for multiple robots. *The International Journal of Robotics Research*, 33(1), 98–112.

van den Berg, J., Lin, M., and Manocha, D. (2008). Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation*, 1928–1935.

van den Berg, J., Guy, S.J., Lin, M., and Manocha, D. (2011). *Reciprocal n-Body Collision Avoidance*, 3–19. Springer Berlin Heidelberg, Berlin, Heidelberg.

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

Wagner, G. and Choset, H. (2015). Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219, 1 – 24.

Wang, W. and Goh, W.B. (2012). Multi-robot path planning with the spatio-temporal A* algorithm and its variants. In *Proceedings of the 10th International Conference on Advanced Agent Technology*, AAMAS'11, 313–329. Springer-Verlag, Berlin, Heidelberg.

Warren, C.W. (1990). Multiple robot path coordination using artificial potential fields. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, 500–505. IEEE.