AIAA SciTech
5-9 January 2015, Kissimmee, Florida
AIAA Guidance, Navigation, and Control Conference

AIAA 2015-0599

Swarm Assignment and Trajectory Optimization Using Variable-Swarm, Distributed Auction Assignment and Model Predictive Control

Daniel Morgan* and Soon-Jo Chung†
University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA

Fred Y. Hadaegh‡
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109, USA

This paper presents a distributed, guidance and control algorithm for reconfiguring swarms composed of hundreds to thousands of agents with limited communication and computation capabilities. This algorithm solves both the optimal assignment and collision-free trajectory generation for swarms, in an integrated manner, when given the desired shape of the swarm (without pre-assigned terminal positions). The optimal assignment problem is solved using a distributed auction assignment that can vary the number of target positions in the assignment, and the collision-free trajectories are generated using sequential convex programming. Finally, model predictive control is used to solve the assignment and trajectory generation in real time using a receding horizon. The model predictive control formulation uses current state measurements to resolve for the optimal assignment and trajectory. The implementation of the distributed auction algorithm and sequential convex programming using model predictive control produces the swarm assignment and trajectory optimization algorithm that transfers a swarm of spacecraft to a desired shape in a distributed fashion. Once the desired shape is uploaded to the swarm, the algorithm determines where each spacecraft goes and how it should get there in a fuel-efficient, collision-free manner.

I. Introduction

Guidance, navigation, and control of multi-agent systems has been a major area of research over the past decades. Initially, the majority of this work focused on swarm robotics.[1–9] However, in recent years the idea of multi-agent systems has been extended to spacecraft.[10–18] The most recent idea is to fly a swarm containing a large number (hundreds to thousands) of femtosatellites (100-gram-class spacecraft).[19]

For a swarm to have a cost benefit compared to a monolithic agent or a smaller formation, the individual agents need to be smaller and cheaper. Due to their small size and low cost, the agents in a swarm have limited actuation, communication and computation capabilities, which require the guidance and control algorithms of the swarm to be both fuel and computationally efficient. Swarms of agents create interesting challenges in guidance and control due to the large number of agents, the small size of each individual agent, and the complicated dynamics. Specifically, the large number of spacecraft makes collision avoidance a major challenge. Also, the limited computation and communication capabilities of each agent require the swarm reconfiguration algorithm to be very simple so that it can be run on board the femtosats in real time.

The swarm reconfiguration problem consists of two parts: assignment and trajectory generation. The assignment problem consists of finding the optimal mapping from a set of agents to a set of targets or tasks in order to minimize the total cost. This problem has been well researched and many methods exist for finding the optimal assignment, including the Hungarian algorithm,[20] iterative methods,[21,22] and auction algorithms.[23,24] The drawback to many of these algorithms is that they are *centralized* with respect to

*Graduate Research Assistant, Department of Aerospace Engineering, morgan29@illinois.edu, AIAA Student Member
†Assistant Professor, Department of Aerospace Engineering, sjchung@illinois.edu, AIAA Senior Member
‡Senior Research Scientist and Technical Fellow, fred.y.hadaegh@jpl.nasa.gov, AIAA Fellow

communication and computation. More recent research on the auction algorithm[25, 26] has shown that it can be implemented in a distributed manner.

The second part of the swarm reconfiguration is trajectory generation and collision avoidance. This part requires an algorithm that can solve the nonlinear optimization that minimizes the cost of the trajectory while satisfying collision avoidance and dynamic constraints. The trajectory optimization has been solved using a variety of methods, including mixed integer linear programming,[27] pseudospectral methods,[28] convex programming,[29] and sequential convex programming (SCP).[30]

In this paper, we simultaneously solve both the target assignment and trajectory optimization problems using a variable-swarm, distributed auction assignment (VSDAA), which allows the swarm size to change over time, to solve the assignment problem and our prior work on SCP[30] to solve the trajectory optimization. Additionally, we integrate these algorithms using model predictive control (MPC)[31, 32] in order to run the algorithms in real time and on swarms with distributed communication networks.

The contributions of this paper are in the development of VSDAA and its integration with SCP in the MPC formulation. The variable-swarm characteristic means that VSDAA can adapt the number of targets in the assignment to match the number of agents. This is incredibly useful when the number of agents in the swarm changes. In the case of a significant loss of agents due to an external object or fuel/battery depletion, VSDAA would adjust the number of targets to match the number of remaining agents and the agents would fill in the gap left by the external object. This allows the swarm to handle the loss of a significant number of agents and still maintain the desired shape. On the other hand, VSDAA will also increase the number of targets if there are more agents than targets. This is a situation that will break a typical auction algorithm[23–26] since the agents cannot all be assigned to a target so they bid indefinitely.

Additionally, the implementation of VSDAA with SCP using MPC allows an assignment to be achieved even in a disconnected communication network. Since the assignment is updated throughout the reconfiguration, the distance-based, swarm communication network will be different every time an assignment is computed. Therefore, the agents do not need to be fully connected to every other agent at all times. In fact, if two agents from separate, disconnected networks are assigned to the same target, they will eventually move close enough to become connected and will be assigned to different targets.

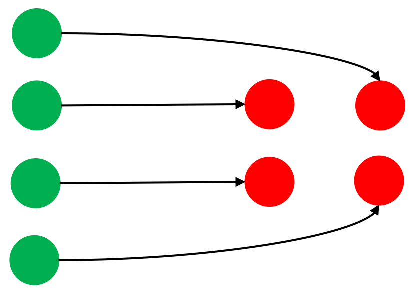
The result of the MPC implementation of VSDAA and SCP is the swarm assignment and trajectory optimization (SATO) algorithm. SATO, the main algorithm of this paper, is distributed in both communication and computation, and provides near-optimal, collision-free trajectories for swarm reconfiguration. In contrast to other methods that simultaneously solve the assignment and trajectory optimization,[33] SATO uses SCP as underlying trajectory optimizazition, which allows it to handle complex dynamic environments. Additionally, this algorithm works with disconnected communication networks and is robust to the loss or gain of a significant number of agents. Finally, the model predictive control formulation allows SATO to run on board each agent and provides robustness to unmodeled disturbances.

The paper is organized as follows. In Sec. II, the swarm reconfiguration problem is formulated as a constrained, nonlinear optimal control problem and converted to a nonlinear optimization. In Sec. III, the swarm reconfiguration problem is broken into an assignment problem and a trajectory optimization problem. Then, the assignment is solved using VSDAA. In Sec. IV, the trajectory optimization problem is converted to a convex optimization and the SCP algorithm is described. Additionally, the SCP algorithm is shown to converge to a trajectory that satisfies the Karush-Kuhn-Tucker (KKT) conditions[34, 35] of the nonconvex problem. In Sec. V, MPC is used to integrate VSDAA and SCP, and to implement a finite horizon so that the resulting algorithm, SATO, can be run on board each agent in real time with a disconnected communication network. In Sec. VI, SATO is run for both a 2-D, double integrator dynamics scenario and a 3-D, relative orbit dynamics scenario. The results of the two scenarios are analyzed and discussed.

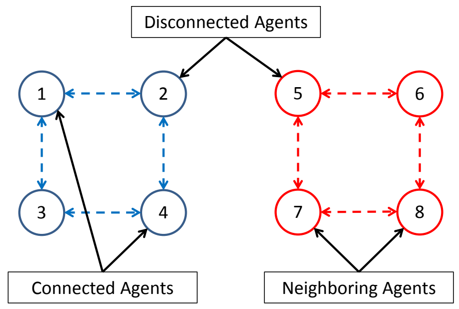
II. Problem Statement

In this section, the optimal swarm reconfiguration is presented as a continuous, finite horizon optimal control problem. The swarm reconfiguration involves the transfer of hundreds to thousands of agents from their current shape to a desired shape while satisfying various constraints, such as collision avoidance, and minimizing the total fuel used during the transfer. A visualization of a swarm reconfiguration and communication networks is shown in Fig. 1.

Figure 1a shows a visualization of the swarm reconfiguration problem. The agents begin at their current positions (green) and move towards the desired formation (red). The agents are interchangeable so any



(a) Visualization of a swarm reconfiguration



(b) Visualization of connected and disconnected networks. Dashed lines represent communication links.

Figure 1. Visualization of problem statement and communication networks

agent can go to any target. In Figure 1b, the connectedness of various agents is shown. The dashed arrows represent communication links between the agents and the different colors represent different communication networks. In other words, the blue agents are connected to the other blue agents (agents 1 and 4) but not to the red agents (agents 2 and 5). Two agents that have a communication link between them are called neighboring agents (agents 7 and 8).

A. Nonlinear Optimal Control Problem

The objective of the optimal swarm reconfiguration is to minimize the $\mathcal{L}_1$-norm of the control input. Therefore, we can define the swarm reconfiguration as follows

Problem 1 (Constrainted, Nonlinear Optimal Control)**.**

$$\min_{\mathbf{u}_j(t), j=1,\ldots,N} \sum_{j=1}^{N} \int_0^{t_f} \|\mathbf{u}_j(t)\|_q \, dt \quad \text{subject to} \tag{1}$$

$$\dot{\mathbf{x}}_j(t) = \mathbf{f}(\mathbf{x}_j(t)) + B\mathbf{u}_j(t)) \qquad \forall t \in [0, t_f], \qquad j = 1, \ldots, N \tag{2}$$

$$\|\mathbf{u}_j(t)\|_r \leq U_{\max} \qquad \forall t \in [0, t_f], \qquad j = 1, \ldots, N \tag{3}$$

$$\|G[\mathbf{x}_j(t) - \mathbf{x}_i(t)]\|_2 \geq R_{\text{col}} \qquad \forall t \in [0, t_f], \qquad i < j, \qquad j = 1, \ldots, N-1 \tag{4}$$

$$\mathbf{x}_j(0) = \mathbf{x}_{j,0}, \qquad j = 1, \ldots, N \tag{5}$$

$$\mathbf{x}_j(t_f) \in \mathcal{X}_f, \qquad j = 1, \ldots, N \tag{6}$$

where $B = [\mathbf{0}_{3\times3} \quad \mathbf{I}_{3\times3}]^T$, $G = [\mathbf{I}_{3\times3} \quad \mathbf{0}_{3\times3}]$, $\mathbf{x}_j = (\boldsymbol{\ell}_j^T, \dot{\boldsymbol{\ell}}_j^T)^T$, $\boldsymbol{\ell}_j \in \mathbb{R}^n$ is the position vector of agent j, $\mathbf{x}_{j,0}$ is the initial state of agent j, $\mathbf{u}_j$ is the control vector of agent j, and N is the number of agents in the swarm. Equations (2)-(5) represent the dynamics constraint, maximum control constraint, collision avoidance constraint, and initial state constraint, respectively, with $U_{\max}$ being the maximum control magnitude and R_{col} being the minimum allowable distance between two agents. Eq. (6) represents the terminal state constraint with $\mathcal{X}_f$ being a set of M discrete points in $\mathbb{R}^n$. This constraint is what introduces the need for solving for the optimal assignment and differentiates this paper from our prior work[30] where individual terminal state assignments were given.

Remark 1 (Norms)**.** The norms used in Eqs. (1) and (3), $\|\cdot\|_q$ and $\|\cdot\|_r$, respectively, are dependent on the hardware used on board the agents. In this paper, we use $q = 1$ and $r = \infty$. However, the convex optimizations are valid for $q, r \in \{1, 2, \infty\}$.

Remark 2 (Fixed Terminal Time)**.** The optimizations used in this paper all have a fixed terminal time. This is due to the fact that the swarm is trying to reconfigure to a specific shape so the agents need to arrive at their terminal positions at the same time. If some agents arrive earlier than others, they will either drift off of their target position or require extra cost to maintain their position in the presence of dynamics. Additionally, the trajectories are generally cheaper for longer reconfiguration times so having some agents arrive earlier than others usually increases the cost for those agents arriving early.

B. Convexification of Differential Equations

In this section, the dynamics constraints in Eq. (2) are converted to affine equality constraints. This is done by linearizing Eq. (2) and discretizing Problem 1. This results in a finite number of linear equality constraints, which are acceptable in a convex programming problem.

In order to rewrite the dynamics in Eq. (2) as a constraint that can be used in a convex programming problem, these equations must first be linearized. Linearizing Eq. (2) yields

$$\dot{\mathbf{x}}_j = A(\mathbf{x}_j^0)\mathbf{x}_j + B\mathbf{u}_j + z(\mathbf{x}_j^0) \tag{7}$$

where $\mathbf{x}_j^0$ is the trajectory about which the equations are linearized. Additionally, $A(\mathbf{x}_j^0)$ and $z(\mathbf{x}_j^0)$ are

$$A(\mathbf{x}_j^0) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}_j}\right|_{\mathbf{x}_j^0}, \qquad z(\mathbf{x}_j^0) = \mathbf{f}(\mathbf{x}_j^0) - \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}_j}\right|_{\mathbf{x}_j^0} \mathbf{x}_j^0 \tag{8}$$

The next step in the process of converting Eq. (2) into a constraint that can be used in convex programming is to convert the ordinary differential equation in Eq. (7) to a finite number of algebraic constraints. In order to do this, the problem is discretized using a zero-order-hold approach such that

$$\mathbf{u}_j(t) = \mathbf{u}_j[k], \qquad t \in [t_k, t_{k+1}), \qquad k = k_0, \ldots, T-1 \tag{9}$$

where $t_f = T\Delta t$, T is the number of discrete time steps, $t_0 = 0$, $t_T = t_f$, and $\Delta t = t_{k+1} - t_k$ for $k = k_0, \ldots, T-1$. This method of discretization reduces Eq. (7) to

$$\mathbf{x}_j[k+1] = A_j[k]\mathbf{x}_j[k] + B_j[k]\mathbf{u}_j[k] + z_j[k], \qquad k = k_0, \ldots, T-1, \qquad j = 1, \ldots, N \tag{10}$$

where $\mathbf{x}_j[k] = \mathbf{x}_j(t_k)$, $\mathbf{u}_j[k] = \mathbf{u}_j(t_k)$, and

$$\begin{aligned} A_j[k] =& e^{A(\mathbf{x}_j^0(t_k))\Delta t}, \qquad B_j[k] = \int_0^{\Delta t} e^{A(\mathbf{x}_j^0(t_k))\tau} B \, d\tau, \\ z_j[k] =& \int_0^{\Delta t} e^{A(\mathbf{x}_j^0(t_k))\tau} z(\mathbf{x}_j^0(t_k)) d\tau \end{aligned} \tag{11}$$

Now that the nonlinear, continuous-time equations of motion from Eq. (2) have been rewritten as linear, finite dimensional constraints in Eq. (10), they can be used in a convex programming problem. The constraints from Eqs. (3)-(6) can be written in discretized form as

$$\|\mathbf{u}_j[k]\|_\infty \le U_{\max} \qquad k = k_0, \ldots, T-1, \qquad j = 1, \ldots, N \tag{12}$$

$$\|G(\mathbf{x}_j[k] - \mathbf{x}_i[k])\|_2 \ge R_{\text{col}} \qquad k = k_0, \ldots, T, \qquad i < j, \qquad j = 1, \ldots, N-1 \tag{13}$$

$$\mathbf{x}_j[0] = \mathbf{x}_{j,0}, \qquad j = 1, \ldots, N \tag{14}$$

$$\mathbf{x}_j[T] \in \mathcal{X}_f, \qquad j = 1, \ldots, N \tag{15}$$

Note that the only constraints that do not satisfy the requirements of convex programming are Eq. (13) and Eq. (15). These constraints will be modified in the following sections so that the problem can be efficiently solved using convex programming.

III. Distributed Optimal Target Assignment

In this section, the nonlinear optimal control problem (Problem 1) is broken into two parts: an optimal assignment problem and an optimal trajectory-planning problem. This separation allows us to rewrite the terminal constraints in Eq. (15), which are nonconvex and require the problem to include integer variables. By solving an assignment problem to determine the terminal states of each agent, the remaining trajectory-planning problem can be approximated by a convex program and efficiently solved.

Claim 1 (Assignment)**.** If the terminal set ($\mathcal{X}_f$) is a set of points with every pair of points separated by a safe distance (R_{col}), then the constraints $\mathbf{x}_j[T] \in \mathcal{X}_f$ (Eq. (15)) and $\|G(\mathbf{x}_j[T] - \mathbf{x}_i[T])\|_2 \geq R_{\text{col}}$ (Eq. (13) at $k = T$) can be equivalently written as

$$\mathbf{x}_j[T] \in \mathcal{X}_f, \qquad \mathbf{x}_j[T] \neq \mathbf{x}_i[T], \qquad \forall j \neq i \tag{16}$$

Now, the assignment problem can be written as shown below.

Problem 2 (Assignment Problem)**.**

$$\min_{\mathbf{x}_{j,f},\ j=1\ldots N} \sum_{j=1}^{N} C(\mathbf{x}_{j,0}, \mathbf{x}_{j,f}) \quad \text{subject to} \tag{17}$$

$$\mathbf{x}_{j,f} \in \mathcal{X}_f, \qquad \mathbf{x}_{j,f} \neq \mathbf{x}_{i,f}, \qquad \forall j = 1 \ldots N, \qquad \forall i \neq j$$

where $C(\mathbf{x}_0, \mathbf{x}_f)$ is the cost required for an agent to go from $\mathbf{x}_0$ to $\mathbf{x}_f$.

The solution to the Assignment Problem (Problem 2) will yield the desired terminal points for each agent ($\mathbf{x}_{j,f}$), which are then used to formulate the following terminal constraint for the trajectory optimization problem.

$$\mathbf{x}_j[T] = \mathbf{x}_{j,f}, \qquad j = 1, \ldots, N \tag{18}$$

The resulting trajectory optimization can be written as follows:

Problem 3 (Trajectory Optimization)**.**

$$\min_{\mathbf{u}_j, j=1,\ldots,N} \sum_{j=1}^{N} \sum_{k=k_0}^{T-1} \|\mathbf{u}_j[k]\|_1 \Delta t \quad \text{subject to} \quad \{(10), (12), (13), (14), (18)\} \tag{19}$$

Remark 3 (Assignment Cost Function)**.** The assignment cost function ($C(\mathbf{x}_0, \mathbf{x}_f)$) should approximate the optimal cost of solving Problem 3 so that the assignment and trajectory optimization are optimizing the same quantity. However, an exact solution to Problem 3 is difficult to obtain when calculating the optimal assignment due to the coupling between agents in the collision avoidance constraint (Eq. (13)). Therefore, the cost function used in Problem 2 should be the solution to Problem 3 without the collision avoidance constraint (Eq. (13)). The resulting problem is a decoupled convex program, which can be efficiently solved. In some cases, such as double integrator dynamics, an algebraic function, such as distance between the initial and terminal points can be used as the cost function in Problem 2 to further simplify the calculations.

A. Distributed Auction Algorithm

To solve the Assignment Problem (Problem 2), an auction algorithm is used. This algorithm is typically used to solve a centralized assignment problem,[23, 24] but its structure allows it to be implemented in a distributed manner[25] even in some less desirable situations, including limited communication and changes in the number of agents.

In a typical auction algorithm, the computations can be centralized or decentralized, but the current bid prices, current highest bidders, and current bid for each agent must all be stored in shared memory that can be accessed by all of the agents. This requires all-to-all or all-to-one (star) communication, which is an impractical requirement of a large swarm. Additionally, the agents take turns bidding, which requires that each agent knows how many agents are in the swarm.

A distributed auction algorithm, VSDAA, is developed to solve the assignment problem for a swarm of agents that do not have all-to-all or all-to-one communication. VSDAA (Method 1) follows the same steps as a typical auction algorithm with a few exceptions. First, all of the computations are run in parallel and all of variables are stored locally (values for agent i are denoted by superscript i in Method 1). The bid prices of each agent ($\mathbf{p}^i$) are communicated to the neighboring agents and each agent stores the largest price received for each target (line 13). This introduces the possibility that agents are bidding based on outdated prices, but over time all of the agents that are connected on the communication graph will receive updated prices and bid appropriately. Finally, the number of target locations that agent i is bidding on (m^i) can be changed if the number of bids being received by agent i is different than the number of target locations.

In VSDAA (Method 1), each agent runs the algorithm in parallel (line 6) and all computations are done locally. First, we assume that each agent knows the cost ($\mathbf{c}^i(j) = C(\mathbf{x}_{i,0}, \mathcal{X}_f(j))$) of moving from its current location ($\mathbf{x}_{i,0}$) to every target position ($\mathcal{X}_f(j)$). Then, the agents begin the bidding process. Each agent first checks to see if it has been outbid by another agent. If it has been outbid, it checks to see if the number of bids is equal to the number of agents it believes are in the swarm (m^i). If this is the case, the agent increases the number of targets used in the assignment by one (line 11) and increases the magnitude of the bid on each target (line 12) so that each agent has the opportunity to bid on the new target. Additionally, every bid is made negative so that every target becomes available. Then, the agent bids on one of the first m^i target positions by choosing the target with the lowest total cost (line 14), which is composed of the fixed, predetermined cost ($\mathbf{c}^i$) and the variable, bidding cost ($\mathbf{p}^i$). Once the desired target (j^i) for agent i is chosen, the bid amount (γ^i) is set as the difference between the cost of the two cheapest targets (v^i and w^i) with the addition of a small increment (ϵ), which ensures that the bid price on any target is strictly increasing. Finally, the bid price of the chosen target is increased by the bid amount (line 18). The agent then resets its counter to zero. If the agent was not outbid, but some other bid has changed, the agent sets its counter to zero but does not go through the bidding calculations. If none of the bids have changed, the agent increases its counter by one (line 24). Finally, the agent stores its current bid estimates ($\mathbf{p}^i_{\text{old}}$), communicates its bid estimates to its neighbors, and updates its current bids based on the bids received from its neighbors ($\mathcal{N}_i$). This process continues until the agent's counter reaches twice the communication network diameter (D_{comm}), which guarantees that no agents are bidding and an assignment has been achieved. In Method 1, $|\{\cdot\}|$ is used to denote the cardinality of a set.

B. Properties of Variable-Swarm Distributed Auction Assignment

In addition to distributing the computation and communication requirements of the auction algorithm, Method 1 also allows for the number of target locations to adjust based on the number of agents in the swarm. This is done by only using the bidding information so no additional consensus algorithm is needed to determine the number of agents. This is a very useful property when the number of agents in the swarm is large and the loss of some agents needs to be overcome to achieve the goal.

In Method 1, each agent keeps track of the number of targets it thinks the swarm needs (m^i), which is the same as the number of agents it thinks are in the swarm. The number of targets can be increased during the algorithm (line 11) if an agent is about to place a bid and all of the m^i targets already have been bid upon. Since an agent will only change its bid if it is outbid, every target that has been bid upon will have a bidder at all future times. Therefore, every target having a bid implies that there are m^i agents in the swarm plus the agent that is bidding, and therefore is not assigned to a target. For this reason, m^i is increased to allow enough targets for all of the agents. This can happen as many times as needed provided that m^i does not exceed M, which is the number of targets defined in $\mathcal{X}_f$.

In addition to increasing m^i when there are more agents than expected, the algorithm will also adjust to having fewer agents than expected if it notices that some of the targets have not been bid upon. This case is more complicated because it may not be desirable in certain situations and cannot be detected until the bidding is finished. An assignment with more targets than agents can be solved by any auction algorithm, but in the case of swarm reconfiguration, it may not result in the desired shape if the number of agents is significantly less than the number of targets. Therefore, it may be desirable to either rerun the algorithm with the correct number of targets or update the number of targets needed for the next reconfiguration. This idea is shown in the optional step on line 31 where the agents adjust the number of targets in the assignment to be equal to the number of bids it received. This step is labeled optional because it might not be desirable to decrease the number of targets in certain situations. Specifically, if the number of agents is not expected to change and the communication network may become disconnected. In this case, the algorithm will only see the bids of the agents that it is connected to and it will decrease the number of desired targets accordingly. In other words, the algorithm cannot distinguish between agents it cannot communicate with because they are disconnected and agents that have been lost.

Remark 4 (Desired Targets ($\mathcal{X}_f$))**.** VSDAA (Method 1) will access the first m^i targets in $\mathcal{X}_f$ based on how many agents it thinks are in the swarm. To achieve a desirable shape, $\mathcal{X}_f$ should have two characteristics. First, the number of targets (M) defined in $\mathcal{X}_f$ should be large enough that m^i never exceeds the number

Method 1 Variable-Swarm, Distributed Auction Algorithm (VSDAA)

```
 1: $\mathcal{X}_f$ = terminal positions in desired shape
 2: $\mathbf{c}^i(j)$ = cost of agent $i$ choosing target $j$
 3: $m^i$ = # of targets available for agent $i$ to bid on
 4: $\mathbf{p}^i = \mathbf{0}_{1\times m^i}$
 5: $\mathbf{p}^i_{\text{old}} = -\mathbf{1}_{1\times m^i}$
 6: for $i = 1 \ldots N$ (run in parallel) do
 7:   while $count^i < 2D_{\text{comm}}$ do
 8:     if $|\mathbf{p}^i(j^i)| > \mathbf{p}^i_{\text{old}}(j^i)$ then
 9:       $m^i = \max\left(m^i, |\{j|\mathbf{p}^i(j) \neq 0\}|\right)$
10:       if $|\{j|\mathbf{p}^i(j) > 0\}| \geq m^i$ then
11:         $m^i = |\{j|\mathbf{p}^i(j) > 0\}| + 1$
12:         $\mathbf{p}^i(1 : m^i) = -\left(|\mathbf{p}^i(1 : m^i)| + \epsilon\right)$
13:       end if
14:       $v^i = \min_{s=1\ldots m^i} \left(\mathbf{c}^i(s) + |\mathbf{p}^i(s)|\right)$
15:       $j^i = \arg\min_{s=1\ldots m^i} \left(\mathbf{c}^i(s) + |\mathbf{p}^i(s)|\right)$
16:       $w^i = \min_{s=1\ldots m^i, s\neq j^i} \left(\mathbf{c}^i(s) + |\mathbf{p}^i(s)|\right)$
17:       $\gamma^i = w^i - v^i + \epsilon$
18:       $\mathbf{p}^i(j^i) = |\mathbf{p}^i(j^i)| + \gamma^i$
19:       $count^i = 0$
20:     else if $\mathbf{p}^i \neq \mathbf{p}^i_{\text{old}}$ then
21:       $m^i = \max\left(m^i, |\{j|\mathbf{p}^i(j) \neq 0\}|\right)$
22:       $count^i = 0$
23:     else
24:       $count^i = count^i + 1$
25:     end if
26:     $\mathbf{p}^i_{\text{old}} = \mathbf{p}^i$
27:     for $j = 1 \ldots m^i$ do
28:       $\mathbf{p}^i(j) = \min_{s\in\arg\max_{s\in\mathcal{N}_i}(|\mathbf{p}^s(j)|)} \left(\mathbf{p}^s(j)\right)$
29:     end for
30:   end while
31:   Optional: $m^i = |\{j|\mathbf{p}^i(j) \neq 0\}|$
32:   Optional: Go back to line 4 and rerun with new $m^i$
33:   $\mathbf{x}_{i,f} = \mathcal{X}_f(j^i)$
34: end for
```

of targets defined. Second, Method 1 will access the first m^i targets of $\mathcal{X}_f$. Therefore, $\mathcal{X}_f$ should be created so that the desired shape is maintained when only the first m^i points are used. For example, if a unit circle defined by $\{x, y\} = \{\cos\theta, \sin\theta\},\ \theta \in [0, 2\pi]$ is the desired shape, $\theta = \{0 : \frac{\pi}{M} : 2\pi\}$ is not desirable since it would result in a semicircle if only half of the targets are used. Instead, $\theta = \{0 : \frac{2\pi}{M} : 2\pi, \frac{\pi}{M} : \frac{\pi}{M} : \frac{(2M-1)\pi}{M}\}$ is desirable since using half the targets would still result in a circle.

The ability of Method 1 to adjust the number of targets in the assignment to match the number of agents in the swarm allows the algorithm to handle a variety of situations that a typical auction algorithm cannot handle. The main benefit of having the algorithm vary the number of targets is in the case where the number of agents is changing. This is a likely scenario when the swarm has a very large number of agents (over 100) because in this situation, the loss of a few agents is acceptable provided that the swarm can maintain its shape. However, the desired targets will need to be reduced so that the desired shape does not have holes in it. Additionally, varying the number of targets also allows the algorithm to run when the communication network is disconnected. In this case, the agents can only communicate with other agents connected to them in the network, which means that as the network changes and more agents can communicate, there might need to be more targets added to the assignment.

In addition to the aforementioned properties, Method 1 maintains the termination and optimality properties of a typical auction algorithm. The proofs in this section are based on those for a typical auction algorithms,[25] but are modified to account for the changing number of agents and the use of negative price values. In the following proposition, we show that Method 1 terminates in a finite number of bidding iterations and determine an upper bound on the number of iterations.

Proposition 1 (Maximum bids)**.** *The maximum number of bidding iterations that can occur in Method 1 is upper bounded by*

$$D_{\text{comm}}(N-1) \max_{i=1,\dots,N} \left(\lceil \frac{\max_{j=1,\dots,N} \left(\mathbf{c}^i(j)\right) - \min_{j=1,\dots,N} \left(\mathbf{c}^i(j)\right)}{\epsilon} \rceil \right) \tag{20}$$

where $\lceil \cdot \rceil$ represents the ceiling operator (round up to next integer).

Proof. To determine the maximum number of bids, the worst case scenario is considered. In this scenario, only minimum bids of increment ϵ are placed. Without loss of generality, we order the targets so that the cost vector for each agent i satisfies $\mathbf{c}^i(j) >= \mathbf{c}^i(j-1)$ for all j.

First, we consider the case when there are at least as many targets as agents ($m^i \geq N$). Now, we will consider how many minimum bids each agent can place before N targets become equally attractive. Initially, each agent will bid on its cheapest target ($\mathbf{c}^i(1)$) until it reaches $\mathbf{c}^i(2)$. This will require at most $\lceil \frac{\mathbf{c}^i(2) - \mathbf{c}^i(1)}{\epsilon} \rceil$ bids. Now, we define a new cost ($\bar{\mathbf{c}}^i(j)$), which represents the lowest bid which exceeds the cost of target j.

$$\begin{aligned} \bar{\mathbf{c}}^i(j) &= \epsilon \lceil \frac{\mathbf{c}^i(j) - \bar{\mathbf{c}}^i(j-1)}{\epsilon} \rceil + \bar{\mathbf{c}}^i(j-1) \\ \bar{\mathbf{c}}^i(1) &= \mathbf{c}^i(1) \end{aligned} \tag{21}$$

Next, it will bid on targets 1 and 2 until they reach $\mathbf{c}^i(3)$, which will require at most $2 \lceil \frac{\mathbf{c}^i(3) - \bar{\mathbf{c}}^i(2)}{\epsilon} \rceil$ more bids. By continuing with this process until all of the prices are $\mathbf{c}^i(N)$, we say that the maximum number of bids is $\sum_{j=1}^{N-1} j \lceil \frac{\mathbf{c}^i(j+1) - \bar{\mathbf{c}}^i(j)}{\epsilon} \rceil$. This quantity can be bounded as follows

$$\begin{aligned} \sum_{j=1}^{N-1} j \lceil \frac{\mathbf{c}^i(j+1) - \bar{\mathbf{c}}^i(j)}{\epsilon} \rceil &\leq (N-1) \sum_{j=1}^{N-1} \lceil \frac{\mathbf{c}^i(j+1) - \bar{\mathbf{c}}^i(j)}{\epsilon} \rceil \\ &= (N-1) \lceil \frac{\mathbf{c}^i(N) - \mathbf{c}^i(1)}{\epsilon} \rceil \\ &= (N-1) \lceil \frac{\max_{j=1,\dots,N} \left(\mathbf{c}^i(j)\right) - \min_{j=1,\dots,N} \left(\mathbf{c}^i(j)\right)}{\epsilon} \rceil \end{aligned} \tag{22}$$

Once an agent has N equally desirable targets it can always find one that it desires regardless of the choices of the other agents. Once a target is bid above $c^i(N)$, there will always be a more desirable option for agent i. Therefore, the bidding must be finished if all of the agents have N equally desirable choices. In the worst case, this requires $(N-1)\max_{i=1,\dots,N}\left(\lceil\frac{\max_{j=1,\dots,N}\left(c^i(j)\right)-\min_{j=1,\dots,N}\left(c^i(j)\right)}{\epsilon}\rceil\right)$. Additionally, each bid can take up to D_{comm} iterations to propagate through the communication network. Multiplying, the maximum number of bids by the number of iterations needed to communicate the bids results in the desired condition.

Now we consider $m^i < N$. In this case, there are more agents than targets and m^i will increase as described in line 11. When m^i is increased, the magnitude of the price of every target is increased by ϵ (line 12). However, since every price is increased, the net change in the cost of all of the targets is zero so this step does not affect the number of bidding iterations. Additionally, when an agent increases m^i, it still places a minimum bid on a single target. Therefore, each iteration still increases one of the target prices by the minimum and the bound on the maximum number of iterations still holds. $\square$

In addition to showing that the VSDAA (Method 1) terminates in finite time, we show in the following proposition that the assignment that results from this algorithm is near optimal.

Proposition 2 (Near-Optimal Assignment). *Assuming that the communication network is connected, the assignment achieved by Method 1 (j^i) satisfies the following equation:*

$$\sum_{i=1}^{N}\left(\mathbf{c}^i(j^i)\right) \le \min_{s^i}\left(\sum_{i=1}^{N}\left(\mathbf{c}^i(s^i)\right)\right) + N\epsilon \tag{23}$$

where s^i can be any assignment mapping of N agents to m targets.

Proof. Since Method 1 waits for all bids to be communicated throughout the swarm before terminating and all agents are connected, we now can define $\mathbf{p}(j) = \mathbf{p}^i(j)$ and $m = m^i$ for all i, j. These quantities represent the values for the bidding vector and number of targets, on which all agents agree. After any bid, the bidding process sets the new price to be $\mathbf{p}(j^i) = |\hat{\mathbf{p}}(j^i)| + \gamma^i$ where $\hat{\mathbf{p}}$ is the price vector before a bid is placed. This equation is equivalent to the following.

$$\begin{aligned}
\mathbf{p}(j^i) &= |\hat{\mathbf{p}}(j^i)| + w^i - v^i + \epsilon\\
\mathbf{p}(j^i) &= |\hat{\mathbf{p}}(j^i)| + \min_{s=1\dots m^i, s\neq j^i}\left(\mathbf{c}^i(s) + |\hat{\mathbf{p}}(s)|\right) - \min_{s=1\dots m^i}\left(\mathbf{c}^i(s) + |\hat{\mathbf{p}}(s)|\right) + \epsilon\\
\mathbf{p}(j^i) &= |\hat{\mathbf{p}}(j^i)| + \min_{s=1\dots m^i, s\neq j^i}\left(\mathbf{c}^i(s) + |\hat{\mathbf{p}}(s)|\right) - \left(\mathbf{c}^i(j^i) + |\hat{\mathbf{p}}(j^i)|\right) + \epsilon\\
\mathbf{p}(j^i) + \mathbf{c}^i(j^i) &= \min_{s=1\dots m^i, s\neq j^i}\left(\mathbf{c}^i(s) + |\hat{\mathbf{p}}(s)|\right) + \epsilon\\
\mathbf{p}(j^i) + \mathbf{c}^i(j^i) &\le \min_{s=1\dots m^i, s\neq j^i}\left(\mathbf{c}^i(s) + |\mathbf{p}(s)|\right) + \epsilon
\end{aligned} \tag{24}$$

Since $\mathbf{p}(j^i)$ can only increase and is positive at the termination of the algorithm, we can state that after the termination of the Method 1, $\mathbf{c}^i(j^i) + |\mathbf{p}(j^i)| \le \min_{s=1,\dots,m}\left(\mathbf{c}^i(s) + |\mathbf{p}(s)|\right) + \epsilon$. Now consider the total cost for any assignment s^i.

$$\begin{aligned}
\sum_{i=1}^{N}\left(\mathbf{c}^i(s^i)\right) &= \sum_{i=1}^{N}\left(\mathbf{c}^i(s^i) + |\mathbf{p}(s^i)|\right) - \sum_{s=1}^{m}\left(|\mathbf{p}(s)|\right)\\
&\ge \sum_{i=1}^{N}\left(\min_{s=1,\dots,m}\left(\mathbf{c}^i(s) + |\mathbf{p}(s)|\right)\right) - \sum_{s=1}^{m}\left(|\mathbf{p}(s)|\right)
\end{aligned} \tag{25}$$

since summing over $|\mathbf{p}(s)|$ is equivalent for all permutations of s. Eq. (25) holds for any assignment so we can write the total cost of the optimal assignment as

$$\begin{aligned}
\min_{s^i}\left(\sum_{i=1}^{N}\left(\mathbf{c}^i(s^i)\right)\right) &\ge \sum_{i=1}^{N}\left(\min_{s=1,\dots,m}\left(\mathbf{c}^i(s) + |\mathbf{p}(s)|\right)\right) - \sum_{s=1}^{m}\left(|\mathbf{p}(s)|\right)\\
&\ge \sum_{i=1}^{N}\left(\mathbf{c}^i(j^i) + |\mathbf{p}(j^i)| - \epsilon\right) - \sum_{s=1}^{m}\left(|\mathbf{p}(s)|\right)
\end{aligned} \tag{26}$$

$$= \sum_{i=1}^{N} \left(\mathbf{c}^i(j^i) \right) - N\epsilon$$

which is equivalent to Eq. (23) and completes the proof. $\square$

IV. Optimal Trajectory Generation

In this section, the optimization problem described in Problem 3 is solved to determine the optimal, collision-free trajectories that take each agent from its current position to its desired target, which was found using VSDAA (Method 1) in the previous section. To solve Problem 3 efficiently, the collision avoidance constraints (Eq. (13)) are convexified and decoupled so that each agent can use SCP to determine its optimal trajectories. The SCP method is described in more detail in Morgan *et al.*[30]

A. Decoupling and Convexification of Collision Avoidance Constraints

The collision avoidance constraints are dependent on the position of other agents. This makes the optimization coupled in the sense that every agent must know the optimal trajectory of every other agent, which requires the entire optimization to be solved at once. This is undesirable since it requires centralized computations. In order to decouple the optimizations, the nominal trajectories ($\bar{\mathbf{x}}_j$) are used for the positions of the other agents. These nominal trajectories will be defined in Method 2. Now, each agent can solve its own optimization since the objective and constraints no longer depend on other agents' optimal trajectories.

The decoupled version of Problem 3 is written as the following optimization for each agent $j = 1, \dots, N$.

Problem 4 (Decoupled Optimization)**.**

$$\min_{\mathbf{u}_j} \sum_{k=k_0}^{T-1} \|\mathbf{u}_j[k]\|_1 \Delta t \quad \text{subject to} \tag{27}$$

$$\mathbf{x}_j[k+1] = A_j[k]\mathbf{x}_j[k] + B_j[k]\mathbf{u}_j[k] + z_j[k], \qquad k = k_0, \dots, T-1 \tag{28}$$

$$\|\mathbf{u}_j[k]\|_\infty \le U_{\max} \qquad k = k_0, \dots, T-1 \tag{29}$$

$$\mathbf{x}_j[0] = \mathbf{x}_{j,0} \tag{30}$$

$$\mathbf{x}_j[T] = \mathbf{x}_{j,f} \tag{31}$$

$$\|G(\mathbf{x}_j[k] - \bar{\mathbf{x}}_i[k])\|_2 \ge R_{\text{col}} \qquad k = k_0, \dots, T, \qquad i \in \{\mathcal{N}_j \cap \mathcal{P}_j\} \tag{32}$$

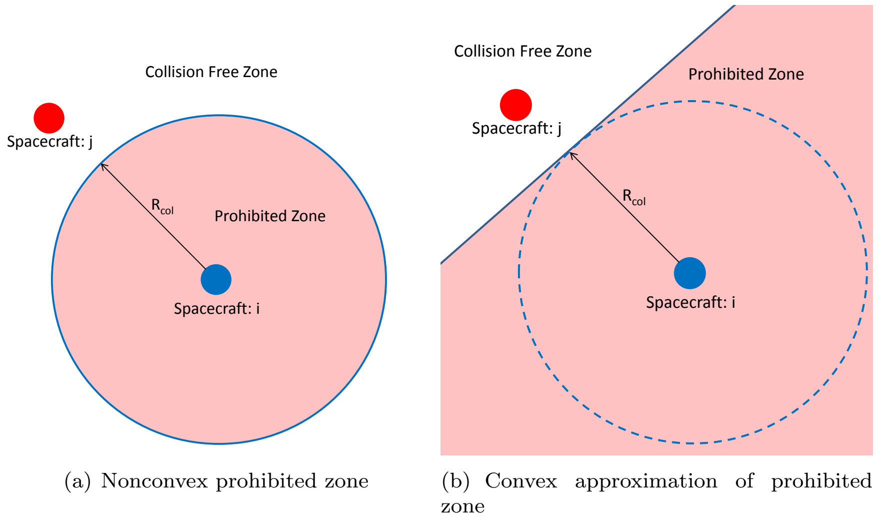
where $\mathcal{N}_j$ is the set of neighboring agents of agent j and $\mathcal{P}_j$ is the set of agents that have a higher priority than agent j meaning that j must avoid them. The set of neighboring agents ($\mathcal{N}_j$) will be further defined in Section V and the set of higher priority agents ($\mathcal{P}_j$) is further defined in Theorem 8 in Section IV.C. For now, they are defined as $\mathcal{N}_j = \{i | i \neq j\}$ and $\mathcal{P}_j = \{i | i < j\}$. It is important to note that either $i \in \mathcal{P}_j$ or $j \in \mathcal{P}_i$ must be true in order to guarantee that agents i and j do not collide.

To use convex programming to solve the trajectory optimization (Problem 4), the nonconvex collision avoidance constraint must be converted to a convex constraint. The best convex approximations of the collision avoidance constraints will be affine constraints. In other words, the circle (2-D) or sphere (3-D) which defines the prohibited region is replaced by a line (2-D) or plane (3-D) which is tangent to the circle (2-D) or sphere (3-D) and perpendicular to the line segment connecting the nominal positions ($\bar{\mathbf{x}}_j$) of the agents. The 2-D version of this idea is shown in Fig. 2.

Figure 2a shows the prohibited zone for the nonconvex collision avoidance constraint. Figure 2b demonstrates the convex approximation of the constraint. As can be seen in Fig. 2b, the new prohibited zone includes the old prohibited zone so the convex collision avoidance constraint is a sufficient condition for the original, nonconvex constraint. Therefore, collision avoidance is still guaranteed using this approximation.

The convex approximation of the nonconvex program in Problem 4 is shown below for agent j.[30]

Problem 5 (Decentralized Convex Program)**.**



(a) Nonconvex prohibited zone (b) Convex approximation of prohibited zone

Figure 2. Convexification of the 2-D collision avoidance constraint[30]

$$\min_{\mathbf{u}_j} \sum_{k=k_0}^{T-1} \|\mathbf{u}_j[k]\|_1 \Delta t \quad \text{subject to} \tag{33}$$

$$\mathbf{x}_j[k+1] = A_j[k]\mathbf{x}_j[k] + B_j[k]\mathbf{u}_j[k] + z_j[k], \qquad k = k_0, \dots, T-1 \tag{34}$$

$$\|\mathbf{u}_j[k]\|_\infty \le U_{\max} \qquad k = k_0, \dots, T-1 \tag{35}$$

$$\mathbf{x}_j[0] = \mathbf{x}_{j,0} \tag{36}$$

$$\mathbf{x}_j[T] = \mathbf{x}_{j,f} \tag{37}$$

$$(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k])^T G^T G(\mathbf{x}_j[k] - \bar{\mathbf{x}}_i[k]) \ge R_{\text{col}} \|G(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k])\|_2 \qquad k = k_0, \dots, T, \qquad i \in \{\mathcal{N}_j \cap \mathcal{P}_j\} \tag{38}$$

B. Sequential Convex Programming

The approximations used to make the trajectory optimization into a convex program, require nominal trajectories ($\bar{\mathbf{x}}_j$) for each spacecraft. Additionally, the nominal trajectories should be close to the actual state trajectories to minimize the approximation error. To ensure that the nominal vectors are good estimates of the actual state vectors, SCP is used. SCP is an iterative method, which solves a convex approximation of a nonconvex problem and uses that solution in the next iteration to convexify the problem, i.e., $\bar{\mathbf{x}}_{j,w}[k] = \mathbf{x}_{j,w-1}[k],\ \forall k, w$ where w is the SCP iteration. This process is repeated until the sequence of trajectories converges according to the following condition.

$$\|\mathbf{x}_{j,w}[k] - \mathbf{x}_{j,w-1}[k]\|_\infty < \epsilon_{\text{SCP}},\ \forall j, k \tag{39}$$

To enforce the collision avoidance constraints, each agent communicates its own nominal trajectory to its neighboring agents ($\mathcal{N}_j$).

The SCP method is described in Method 2. First, an initial trajectory is generated for each agent without considering collision avoidance (line 2). Then, the iterative process begins with each agent solving for its optimal trajectory (line 9). Next, each agent stores the current trajectory as the nominal trajectory for the next iteration (line 12) and communicates that trajectory to its neighboring agents (line 13). Finally, the iteration is repeated until the trajectories converge and the agents are collision free (line 14).

Because the convex optimizations can be solved efficiently, the run time is now on the order of a time step or two so VSDAA and SCP can be implemented using MPC by updating the future assignments and control commands based on the current state, which may have drifted off the optimal trajectories due to unmodeled disturbances or other errors. MPC can provide some robustness to disturbances and allows the communication network to be distributed and disconnected.

Method 2 Sequential Convex Programming[30]

```
1: $\bar{\mathbf{x}}_j[k] := \mathbf{0}_{6\times1},\ \forall j,k$
2: $\mathbf{x}_{j,0}[k] :=$ the solution to Problem 5 (Decentralized Convex Program) with $\mathcal{P}_j = \emptyset,\ \forall j,k$
3: $\bar{\mathbf{x}}_j[k] := \mathbf{x}_j^0[k],\ \forall j,k$
4: Communicate $\bar{\mathbf{x}}_j[k]$ to all neighboring agents ($i \in \mathcal{N}_j$)
5: $\mathcal{K} := \{1,\dots,N\}$
6: $w := 1$
7: while $\mathcal{K} \neq \emptyset$ do
8:     for all $j \in \mathcal{K}$ (run in parallel) do
9:         $\mathbf{x}_{j,w}[k] :=$the solution to Problem 5 (Decentralized Convex Program), $\forall k$
10:    end for
11:    for all $j$ (run in parallel) do
12:        $\bar{\mathbf{x}}_j[k] := \mathbf{x}_{j,w}[k],\ \forall k$
13:        Communicate $\bar{\mathbf{x}}_j[k]$ to all neighboring agents ($i \in \mathcal{N}_j$)
14:        if $\|\mathbf{x}_{j,w}[k] - \mathbf{x}_{j,w-1}[k]\|_\infty < \epsilon_{\text{SCP}}\ \forall k$ and $\|G(\mathbf{x}_{j,w}[k] - \mathbf{x}_{i,w}[k])\|_2 > R_{\text{col}}\ \forall k, \forall i \in \mathcal{N}_j$ then
15:            Remove $j$ from $\mathcal{K}$
16:        end if
17:    end for
18:    $w := w + 1$
19: end while
20: $\mathbf{x}_{j,w-1}[k]$ is the approximate solution to Problem 3
```

C. Convergence of SCP

In this subsection we will show that SCP (Method 2) converges to a point, which satisfies the KKT conditions of the nonconvex optimization in Problem 4. In order to do this, we will show that the sequence of convex programs for agent j converges in two different situations. First, the sequence converges when agent j does not have to avoid any other agents.

Proposition 3 (Convergence without Collision Avoidance)**.** *If $\mathcal{N}_j \cap \mathcal{P}_j = \emptyset$, the solution to the convex program (Problem 5) for agent j is equivalent to the global minimum of the nonconvex program (Problem 4).*

Proof. Since $\mathcal{N}_j \cap \mathcal{P}_j = \emptyset$, agent j does not avoid any other agents, which means that there are no collision avoidance constraints. Without collision avoidance constraints, the nonconvex program (Problem 4) and the convex program (Problem 5) are equivalent and the solution to either problem will be a global minimum to both problems. □

Next, we will show that SCP converges for agent j when all of the agents that agent j must avoid have a fixed trajectory, i.e. their trajectories do not change from one iteration to the next. First, we establish that a solution to the convex program will always be a feasible solution to the nonconvex program. To simplify the notation in the following proofs, we introduce the following definition.

Definition 1 (CP($\bar{\mathbf{x}}_j$,$\bar{\mathbf{x}}_i$))**.** We define CP($\bar{\mathbf{x}}_j$,$\bar{\mathbf{x}}_i$) to be the convex program in Problem 5 where the nominal trajectories $\bar{\mathbf{x}}_j$ and $\bar{\mathbf{x}}_i$ are used for the convexification and decoupling, respectively, of the nonconvex program in Problem 4. The sets $\mathcal{N}_j$ and $\mathcal{P}_j$ are the same as in the nonconvex program (Problem 4).

Proposition 4 (Feasible Solutions)**.** *If $(\mathbf{x}_{j,w}, \mathbf{u}_{j,w})$ is a solution to CP($\mathbf{x}_{j,w-1}$,$\bar{\mathbf{x}}_i$), then $(\mathbf{x}_{j,w}, \mathbf{u}_{j,w})$ is a feasible solution to the nonconvex program (Problem 4).*

Proof. Since $(\mathbf{x}_{j,w}, \mathbf{u}_{j,w})$ is a solution to Problem 5, it satisfies all of the constraints of Problem 5. Except for the collision avoidance constraint (Eq. (32)), the constraints of the nonconvex program (Eqs. (28)-(31)) are the same as those of the convex program (Eqs. (34)-(37)). Additionally, $\mathbf{x}_{j,w}$ satisfies Eq. (38). Therefore, the following is true for all $k = k_0, \dots, T$ and $i \in \{\mathcal{N}_j \cap \mathcal{P}_j\}$:

$$\begin{gathered}(\mathbf{x}_{j,w-1}[k] - \bar{\mathbf{x}}_i[k])^T G^T G(\mathbf{x}_{j,w}[k] - \bar{\mathbf{x}}_i[k]) \geq R_{\text{col}}\|G(\mathbf{x}_{j,w-1}[k] - \bar{\mathbf{x}}_i[k])\|_2 \\ \|G(\mathbf{x}_{j,w-1}[k] - \bar{\mathbf{x}}_i[k])\|_2\|G(\mathbf{x}_{j,w}[k] - \bar{\mathbf{x}}_i[k])\|_2 \geq R_{\text{col}}\|G(\mathbf{x}_{j,w-1}[k] - \bar{\mathbf{x}}_i[k])\|_2 \\ \|G(\mathbf{x}_{j,w}[k] - \bar{\mathbf{x}}_i[k])\|_2 \geq R_{\text{col}}\end{gathered} \tag{40}$$

The last equation in Eq. (40) shows that $\mathbf{x}_{j,w}$ satisfies the collision avoidance constraint in Eq. (32). Therefore, all of the constraints in Problem 4 are satisfied and $\mathbf{x}_{j,w}$ is a feasible solution to Problem 4. $\square$

The above proposition shows that any solution to the convex program is a feasible solution to the nonconvex program. Next, this fact will be used to establish that a sequence of optimal solutions exist.

Proposition 5 (Optimal Sequence)**.** *Let* $\mathbf{x}_i$ *be fixed trajectories for all* $i \in \{\mathcal{N}_j \cap \mathcal{P}_j\}$*. If* $(\mathbf{x}_{j,w}, \mathbf{u}_{j,w})$ *is a feasible solution to* $\text{CP}(\mathbf{x}_{j,w-1},\bar{\mathbf{x}}_i)$ *or the nonconvex program (Problem 4), then* $\text{CP}(\mathbf{x}_{j,w},\bar{\mathbf{x}}_i)$ *has an optimal solution and* $(\mathbf{x}_{j,w}, \mathbf{u}_{j,w})$ *is a feasible solution to* $\text{CP}(\mathbf{x}_{j,w},\bar{\mathbf{x}}_i)$*.*

Proof. Since $(\mathbf{x}_{j,w}, \mathbf{u}_{j,w})$ is a solution to $\text{CP}(\mathbf{x}_{j,w-1},\bar{\mathbf{x}}_i)$ or Problem 4, it satisfies Eqs. (34)-(37), which are the same as the constraints of $\text{CP}(\mathbf{x}_{j,w},\bar{\mathbf{x}}_i)$ except for the collision avoidance constraints (Eq. (38)). Additionally, $\mathbf{x}_{j,w}$ satisfies the collision avoidance constraints of $\text{CP}(\mathbf{x}_{j,w-1},\bar{\mathbf{x}}_i)$ or Problem 4. If $\mathbf{x}_{j,w}$ satisfies the constraints of Problem 4, the first line of Eq. (41) is established, otherwise, this line is established by Proposition 4.

$$\begin{aligned} \|G(\mathbf{x}_{j,w}[k] - \bar{\mathbf{x}}_i[k])\|_2 &\geq R_{\text{col}} \\ \|G(\mathbf{x}_{j,w}[k] - \bar{\mathbf{x}}_i[k])\|_2^2 &\geq R_{\text{col}}\|G(\mathbf{x}_{j,w}[k] - \bar{\mathbf{x}}_i[k])\|_2 \\ (\mathbf{x}_{j,w}[k] - \bar{\mathbf{x}}_i[k])^T G^T G(\mathbf{x}_{j,w}[k] - \bar{\mathbf{x}}_i[k]) &\geq R_{\text{col}}\|G(\mathbf{x}_{j,w}[k] - \bar{\mathbf{x}}_i[k])\|_2 \end{aligned} \tag{41}$$

The last equation in Eq. (41) shows that $\mathbf{x}_{j,w}$ satisfies the collision avoidance constraints in $\text{CP}(\mathbf{x}_{j,w},\bar{\mathbf{x}}_i)$. Therefore, $(\mathbf{x}_{j,w}, \mathbf{u}_{j,w})$ is a feasible solution to $\text{CP}(\mathbf{x}_{j,w},\bar{\mathbf{x}}_i)$ and the set of feasible solutions to $\text{CP}(\mathbf{x}_{j,w},\bar{\mathbf{x}}_i)$ is not empty. Additionally, this set is an intersection of half spaces and equality constraints, which results in a closed set, and the Eqs. (34)-(36) ensure that the set is bounded. Also, the cost of $\text{CP}(\mathbf{x}_{j,w},\bar{\mathbf{x}}_i)$ is continuous. By the Weierstrass theorem,[36] a continuous function over a closed and bounded set achieves an optimum. Therefore, an optimal solution to $\text{CP}(\mathbf{x}_{j,w},\bar{\mathbf{x}}_i)$ exists. $\square$

Proposition 5 shows that once a feasible solution to $\text{CP}(\mathbf{x}_{j,w},\bar{\mathbf{x}}_i)$ exists, all of the following convex problems have an optimal solution and a sequence of optimal solutions $\{\mathbf{x}_{j,w}\}$ exists. The next proposition shows that the optimality of this sequence is improving.

Proposition 6 (Decreasing Cost)**.** *If* $(\mathbf{x}_{j,w}, \mathbf{u}_{j,w})$ *is the optimal solution to* $\text{CP}(\mathbf{x}_{j,w-1},\bar{\mathbf{x}}_i)$ *and* $(\mathbf{x}_{j,w-1}, \mathbf{u}_{j,w-1})$ *is a feasible solution to* $\text{CP}(\mathbf{x}_{j,w-2},\bar{\mathbf{x}}_i)$*, then*

$$J(\mathbf{u}_{j,w}) \leq J(\mathbf{u}_{j,w-1}) \tag{42}$$

where $J(\mathbf{u}_j) = \sum_{k=k_0}^{T-1} \|\mathbf{u}_j[k]\|_1 \Delta t$*. Additionally, if* $(\mathbf{x}_{j,w}, \mathbf{u}_{j,w})$ *is the unique optimal solution, then either* $(\mathbf{x}_{j,w}, \mathbf{u}_{j,w}) = (\mathbf{x}_{j,w-1}, \mathbf{u}_{j,w-1})$ *or*

$$J(\mathbf{u}_{j,w}) < J(\mathbf{u}_{j,w-1}) \tag{43}$$

Proof. Since $(\mathbf{x}_{j,w-1}, \mathbf{u}_{j,w-1})$ is a feasible solution to $\text{CP}(\mathbf{x}_{j,w-2},\bar{\mathbf{x}}_i)$, Proposition 5 states that $(\mathbf{x}_{j,w-1}, \mathbf{u}_{j,w-1})$ is a feasible solution to $\text{CP}(\mathbf{x}_{j,w-1},\bar{\mathbf{x}}_i)$. Additionally, $(\mathbf{x}_{j,w}, \mathbf{u}_{j,w})$ is the optimal solution to $\text{CP}(\mathbf{x}_{j,w-1},\bar{\mathbf{x}}_i)$, which means that any $(\mathbf{x}_j, \mathbf{u}_j)$ that is a feasible solution of $\text{CP}(\mathbf{x}_{j,w-1},\bar{\mathbf{x}}_i)$ satisfies

$$J(\mathbf{u}_{j,w}) \leq J(\mathbf{u}_j) \tag{44}$$

Substituting $(\mathbf{x}_{j,w-1}, \mathbf{u}_{j,w-1})$ into the right hand side of the above equation establishes Eq. (42). If $(\mathbf{x}_{j,w}, \mathbf{u}_{j,w})$ is a unique optimal solution, then Eq. (44) has a strict inequality provided that $\mathbf{u}_j \neq \mathbf{u}_{j,w}$. Substituting in $(\mathbf{x}_{j,w-1}, \mathbf{u}_{j,w-1})$ establishes Eq. (43) unless $(\mathbf{x}_{j,w}, \mathbf{u}_{j,w}) = (\mathbf{x}_{j,w-1}, \mathbf{u}_{j,w-1})$. $\square$

This proposition establishes that a sequence of optimal solutions $\{\mathbf{x}_{j,w}\}$ has a nonincreasing cost and if these solutions are unique, they have a decreasing cost.

We will now use the propositions developed in this section to show that a sequence of optimal solutions exists and converges to an optimal solution of Problem 4. The following theorem establishes these claims.

Theorem 7 (Convergence of SCP)**.** *Let* $\mathbf{x}_i$ *be fixed trajectories for all* $i \in \{\mathcal{N}_j \cap \mathcal{P}_j\}$*. If* $(\mathbf{x}_{j,1}, \mathbf{u}_{j,1})$ *is a feasible solution to Problem 4, then a sequence of optimal solutions* $(\{\mathbf{x}_{j,w}\}, \{\mathbf{u}_{j,w}\})$ *exists. If each optimal solution is unique, the sequence converges to* $(\mathbf{x}_j^*, \mathbf{u}_j^*)$*, which is an optimal solution to Problem 4.*

Proof. Since $(\mathbf{x}_{j,1}, \mathbf{u}_{j,1})$ is a feasible solution to Problem 4, it follows from Proposition 5 that CP($\mathbf{x}_{j,1}$,$\bar{\mathbf{x}}_i$) has an optimal solution, which we call $(\mathbf{x}_{j,2}, \mathbf{u}_{j,2})$. Applying Proposition 5 again results in an optimal solution to CP($\mathbf{x}_{j,2}$,$\bar{\mathbf{x}}_i$) defined to be $(\mathbf{x}_{j,3}, \mathbf{u}_{j,3})$. Repeating this process yields the optimal solution sequence $(\{\mathbf{x}_{j,w}\}, \{\mathbf{u}_{j,w}\})$. Applying Proposition 6 to this yields the following condition for all w:

$$J(\mathbf{u}_{j,w}) < J(\mathbf{u}_{j,w-1}) \tag{45}$$

Since every solution in the sequence $(\{\mathbf{x}_{j,w}\}, \{\mathbf{u}_{j,w}\})$ satisfies Eqs. (34)-(37), which forms a closed and bounded set, there is an infinite subsequence $(\{\mathbf{x}_{j,w_i}\}, \{\mathbf{u}_{j,w_i}\})$ that converges. Let the convergence point be called $(\mathbf{x}_j^*, \mathbf{u}_j^*)$. Because the cost function $(J(\mathbf{u})$ is continuous, $J(\mathbf{u}_{j,w_i})$ converges to $J(\mathbf{u}_j^*)$. Additionally, the cost function is nonincreasing as seen in Eq. (45). Therefore, the cost function of the entire sequence $(J(\mathbf{u}_{j,w}))$ converges to $J(\mathbf{u}_j^*)$.

We define a mapping $T(\mathbf{x}_j)$ that represents solving CP($\mathbf{x}_j$,$\bar{\mathbf{x}}_i$). The mapping T has a fixed point at $\mathbf{x}_j^*$. We will show that this is true by contradiction. Assume that $(\mathbf{x}_j^{*+1}, \mathbf{u}_j^{*+1}) = T(\mathbf{x}_j^*)$ and $(\mathbf{x}_j^{*+1}, \mathbf{u}_j^{*+1}) \neq (\mathbf{x}_j^*, \mathbf{u}_j^*)$. In this case, $J(\mathbf{u}_{j,*+1}) < J(\mathbf{u}_j^*)$. However, $\{J(\mathbf{u}_{j,w})\} \to J(\mathbf{u}_j^*)$ so we have, for $w > *+1$, that $J(\mathbf{u}_{j,w}) < J(\mathbf{u}_j^*)$ and $\{J(\mathbf{u}_{j,w})\} \to J(\mathbf{u}_j^*)$, which is a contradiction. Therefore, $(\mathbf{x}_j^*, \mathbf{u}_j^*) = T(\mathbf{x}_j^*, \mathbf{u}_j^*)$.

Additionally, the mapping $T(\mathbf{x}_j)$ is equivalent to solving the KKT conditions of CP($\mathbf{x}_j$,$\bar{\mathbf{x}}_i$), which are continuous with respect to $\mathbf{x}_j$. Therefore, the mapping T is continuous. Since the subsequence $(\{\mathbf{x}_{j,w_i}\}, \{\mathbf{u}_{j,w_i}\}) \to (\mathbf{x}_j^*, \mathbf{u}_j^*)$ and T is continuous, the following is true.

$$\{T(\mathbf{x}_{j,w_i})\} \to T(\mathbf{x}_j^*) \tag{46}$$

Additionally, $\mathbf{x}_j^*$ is a fixed point and $\mathbf{x}_{j,w_i+1} = T(\mathbf{x}_{j,w_i})$. Therefore,

$$\{\mathbf{x}_{j,w_i+1}\} \to \mathbf{x}_j^* \tag{47}$$

This process can be repeated to show that all subsequences $\{\mathbf{x}_{j,w_i+n}\}$ converge to $\mathbf{x}_j^*$. Therefore, the sequence $\{\mathbf{x}_{j,w}\}$ converges to $\mathbf{x}_j^*$.

Finally, we will show that $\mathbf{x}_j^*$ is a KKT point of Problem 4. Since $\mathbf{x}_j^*$ is a fixed point of T, it is a solution to CP($\mathbf{x}_j^*$,$\bar{\mathbf{x}}_i$) and from Proposition 4, it is a feasible solution to Problem 4. Additionally, $\mathbf{x}_j^*$ is a KKT point of Problem 5 so it satisfies the following equations:

$$0 = \partial J(\mathbf{u}_j^*) + \mu_1^T \partial g_1(\mathbf{x}_j^*, \mathbf{u}_j^*) + \mu_2^T \partial g_2(\mathbf{x}_j^*, \mathbf{u}_j^*) + \lambda^T \partial h(\mathbf{x}_j^*, \mathbf{u}_j^*) \tag{48}$$
$$0 = \mu_1^T g_1(\mathbf{x}_j^*, \mathbf{u}_j^*) + \mu_2^T g_2(\mathbf{x}_j^*, \mathbf{u}_j^*) \tag{49}$$
$$\mu_1 \geq \mathbf{0} \tag{50}$$
$$\mu_2 \geq \mathbf{0} \tag{51}$$

where ∂f is the subgradient of a function f and

$$g_1(\mathbf{x}_j, \mathbf{u}_j) = \|\mathbf{u}_j[k]\|_\infty - U_{\max}, \quad \forall k = k_0, \ldots, T \tag{52}$$
$$g_2(\mathbf{x}_j, \mathbf{u}_j) = R_{\text{col}} \|G(\mathbf{x}_j^*[k] - \bar{\mathbf{x}}_i)\|_2 - (\mathbf{x}_j^*[k] - \bar{\mathbf{x}}_i)^T G^T G(\mathbf{x}_j[k] - \bar{\mathbf{x}}_i), \quad \forall k = k_0, \ldots, T \tag{53}$$
$$h(\mathbf{x}_j, \mathbf{u}_j) = \begin{bmatrix} \mathbf{x}_j[k+1] - A_j[k]\mathbf{x}_j[k] - B_j[k]\mathbf{u}_j[k] - z_j[k], \quad \forall k = k_0, \ldots, T-1 \\ \mathbf{x}_j[0] - \mathbf{x}_{j,0} \\ \mathbf{x}_j[T] - \mathbf{x}_{j,f} \end{bmatrix} \tag{54}$$

We note that the cost J and the constraints in g_1 and h are the same in both Problems 4 and 5. Therefore, we will only substitute Eq. (53) for g_2 in the KKT conditions. The resulting conditions are

$$0 = \partial J(\mathbf{u}_j^*) + (\mu_1^*)^T \partial g_1(\mathbf{x}_j^*, \mathbf{u}_j^*) - (\mu_2^*)^T (\mathbf{x}_j^*[k] - \bar{\mathbf{x}}_i)^T G^T G + (\lambda^*)^T \partial h(\mathbf{x}_j^*, \mathbf{u}_j^*) \tag{55}$$
$$0 = (\mu_1^*)^T g_1(\mathbf{x}_j^*, \mathbf{u}_j^*) + (\mu_2^*)^T (R_{\text{col}} - \|G(\mathbf{x}_j^*[k] - \bar{\mathbf{x}}_i)\|_2) \|G(\mathbf{x}_j^*[k] - \bar{\mathbf{x}}_i)\|_2 \tag{56}$$
$$\mu_1^* \geq \mathbf{0} \tag{57}$$
$$\mu_2^* \geq \mathbf{0} \tag{58}$$

Now, consider the KKT conditions of the nonconvex program (Problem 4) as shown below.

$$0 = \partial J(\mathbf{u}_j^*) + (\mu_1)^T \partial g_1(\mathbf{x}_j^*, \mathbf{u}_j^*) - (\mu_2)^T \frac{(\mathbf{x}_j^*[k] - \bar{\mathbf{x}}_i)^T G^T G}{\|G(\mathbf{x}_j^*[k] - \bar{\mathbf{x}}_i)\|_2} + (\lambda)^T \partial h(\mathbf{x}_j^*, \mathbf{u}_j^*) \tag{59}$$

$$0 = (\mu_1)^T g_1(\mathbf{x}_j^*, \mathbf{u}_j^*) + (\mu_2)^T (R_{\text{col}} - \|G(\mathbf{x}_j^*[k] - \bar{\mathbf{x}}_i)\|_2) \tag{60}$$

$$\mu_1 \geq \mathbf{0} \tag{61}$$

$$\mu_2 \geq \mathbf{0} \tag{62}$$

Now let $\mu_1 = \mu_1^*$, $\mu_2 = \mu_2^* \|G(\mathbf{x}_j^*[k] - \bar{\mathbf{x}}_i)\|_2$, and $\lambda = \lambda^*$ where $(\mu_1^*, \mu_2^*, \lambda^*)$ are the KKT multipliers that satisfy the KKT conditions for the convex program. When (μ_1, μ_2, λ) are substituted into Eqs. (59)-(62), these equations reduce to Eqs. (55)-(58), which are satisfied by $(\mathbf{x}_j^*, \mathbf{u}_j^*)$. Therefore, $(\mathbf{x}_j^*, \mathbf{u}_j^*)$ satisfies the KKT conditions for the nonconvex program (Problem 4). □

Proposition 3 and Theorem 7 show that the SCP algorithm (Method 2) applied to the convex program (Problem 5) converges to a trajectory that satisfies the KKT conditions of the nonconvex program (Problem 4). Proposition 3 applies when there are no collision avoidance constraints ($\mathcal{N}_j \cap \mathcal{P}_j = \emptyset$) and Theorem 7 applies when the agents that need to be avoided have a fixed trajectory ($\mathbf{x}_{i,w} = \mathbf{x}_i^*,\ \forall w, \forall i \in \mathcal{N}_j \cap \mathcal{P}_j$). In order to guarantees that the SCP algorithm for every agent converges, a priority value (ρ_j) is defined for each agent and is used to construct $\mathcal{P}_j$.

Theorem 8 (Convergence of All Agents)**.** *Let each agent j have a priority value ρ_j such that $\rho_i \neq \rho_j$ for any $i \neq j$. Define the priority set as follows:*

$$\mathcal{P}_j = \{i | \rho_i < \rho_j\} \tag{63}$$

If there is a feasible solution to Problem 4 for each agent and $\mathcal{N}_j = \{i | i \neq j\}$, all of the agents will converge to a KKT solution of Problem 4 and no collisions occur.

Proof. Since $\rho_i \neq \rho_j$ for any $i \neq j$, there exists an agent j_1 such that $\rho_{j_1} < \rho_j$ for all $j \neq j_1$. This will result in $\mathcal{P}_{j_1} = \emptyset$. Using Proposition 3, agent j_1 converges to a solution that is a global minimum of Problem 4 and therefore, satisfies the KKT conditions. This solution is defined as $\mathbf{x}_{j_1}^*$.

Now there exists an agent j_2 such that $\rho_{j_2} < \rho_j$ for all $j \neq \{j_1, j_2\}$ and $\rho_{j_1} < \rho_{j_2}$. This implies that $\mathcal{P}_{j_2} = j_1$. Now, Theorem 7 can be applied, using the assumption that there is a feasible solution and the fact that $\mathbf{x}_{j_1}^*$ is fixed, to show that the SCP algorithm for agent j_2 converges to a solution $(\mathbf{x}_{j_2}^*)$ that is a KKT point of Problem 4. This step can be repeated for all of the remaining agents to show that every agent's trajectory converges to a KKT point.

Additionally, since $i \in \mathcal{P}_j$ if and only if $\rho_i < \rho_j$ we can state that exactly one of the following statements is true for all pairs (i, j): $i \in \mathcal{P}_j$ or $j \in \mathcal{P}_i$. Since either i or j will avoid the other one, the collision is accounted for in the optimizations. Since this is true for all agent pairs, every possible collision is considered in the optimizations so no collisions will occur. □

Remark 5 (Priority Value (ρ_j))**.** The priority values (ρ_j) described in Theorem 8 should be defined by a physical parameter that reflects the ability of each agent to avoid other agents. The most obvious choice for this value is the fuel/battery remaining for each agent. Using this quantity, the agent with more fuel/battery remaining would avoid the collision.

V. Swarm Assignment and Trajectory Optimization

In this section, MPC is used to implement the VSDAA and SCP methods in real time creating the swarm assignment and trajectory optimization (SATO) algorithm. Additionally, the communication requirement in the swarm is reduced so that the network does not need to be centralized or even connected for the algorithm to work. MPC uses a receding horizon to update the optimal assignments and trajectories based on the current state information. Once an optimal control sequence or optimal assignment is calculated, those values are used until updated values are calculated.

A. Model Predictive Control Formulation

SATO updates the state ($\mathbf{x}_{j,k_0}$) and time (k_0) and uses this information as the initial conditions in the optimization. Additionally, a receding horizon, which is T_H time steps long, is used to reduce the size of the optimizations so that they can be run more frequently. Additionally, we will consider communication networks that are distributed and disconnected. With these ideas in mind, each agent will only consider collision avoidance with their neighboring agents and only for the length of the horizon. This allows us to reduce the size of the optimizations without hindering the safety of the agents by considering future collision avoidance in future optimizations. To incorporate these ideas into SATO we modify the collision avoidance constraint in Eq. (38) as follows.

$$(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k])^T G^T G(\mathbf{x}_j[k] - \bar{\mathbf{x}}_i[k]) \geq R_{\text{col}} \|G(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k])\|_2 \qquad (64)$$
$$k = k_0, \ldots, \min\{k_0 + T_H, T\}, \qquad i \in \mathcal{N}_j \cap \mathcal{P}_j$$

where

$$\mathcal{N}_j = \{\|\mathbf{x}_j[k_0] - \mathbf{x}_i[k_0]\|_2 \leq R_{\text{comm}}\} \qquad (65)$$

and R_{comm} is the communication radius of each agent.

Since we are only considering collisions between neighboring agents, it is important to guarantee that no agents can start in positions where they cannot communicate and end up colliding before a new trajectory is generated. To do this, an artificial velocity constraint is introduced so that a bound can be placed on the distance two agents can move relative to each other in a limited amount of time. This constraint is shown below.

$$\|H\mathbf{x}_j[k]\|_2 \leq V_{\max} \qquad k = k_0, \ldots, T \qquad (66)$$

where $H = [0_{3\times3} \quad I_{3\times3}]$.

Next, we define two modifications of Problem 5. The first modification is used in SATO to estimate the costs used in VSDAA. The problem is defined as follows.

Problem 6 (Auction Cost)**.**

$$\min_{\mathbf{u}_j} \sum_{k=k_0}^{T-1} \|\mathbf{u}_j[k]\|_1 \Delta t \quad \text{subject to} \quad \{(10), (12), (14), (66)\}\text{, and } \mathbf{x}_j[T] = \mathcal{X}_f(j) \qquad (67)$$

It is important to note that this problem does not include collision avoidance. This is due to the fact that collisions are highly dependent on the assignment, which has not been determined when this problem is used. Additionally, the number of collisions will generally be small for the optimal assignment so excluding them should not have a large effect on the cost.

The other modified problem that will be introduced will be used when SATO uses SCP. This problem is defined below.

Problem 7 (Limited Horizon SCP)**.**

$$\min_{\mathbf{u}_j} \sum_{k=k_0}^{T-1} \|\mathbf{u}_j[k]\|_1 \Delta t \quad \text{subject to} \quad \{(10), (12), (14), (18), (64), (66)\} \qquad (68)$$

This problem uses the new collision avoidance constraint, which considers collisions only in the short term, with respect to both position and time.

SATO is described in Method 3. First, the auction costs are generated for each agent going to each target position (line 5). Then, VSDAA is used to compute the optimal assignment (line 9). Next, SCP is used to generate the optimal trajectory and corresponding control sequence (line 14). Finally, the optimal control sequence is applied until a new optimal control is generated (line 16). The current time and state are then updated and the process is repeated until the terminal time is reached.

After the optimal assignment is computed, the algorithm checks to see if the number of targets that have been bid upon has changed from the previous iteration. If the number of bids has changed, that means

Method 3 Swarm Assignment and Trajectory Optimization (SATO)

```
1: $k_0 = 0$
2: while $k_0 \leq T$ do
3:    for all $i = 1, \ldots, N$ (parallel) do
4:       for all $j = 1, \ldots, M$ do
5:          Solve Problem 6 using SCP (Method 2)
6:          $\mathbf{c}^i(j)$ = cost of optimal solution to Problem 6
7:       end for
8:    end for
9:    Solve Problem 2 using VSDAA (Method 1)
10:   $\mathbf{x}_{j,f}$ = solution to Problem 2, $\forall j$
11:   if # of bids has changed then
12:      $k_0 = 0$
13:   end if
14:   Solve Problem 7 using SCP (Method 2)
15:   $\mathbf{u}_j[k]$ = control solution to Problem 7, $\forall j$, $k = k_0 \ldots k_0 + T_H - 1$
16:   Apply $\mathbf{u}_j[k]$ for $k = k_0 \ldots k_0 + T_H - 1$
17:   Update $k_0$ and $\mathbf{x}_{j,k_0}$ to current time
18: end while
```

the number of agents in the communication network has changed. This can result in significantly different assignments due to new information about other agents. To prevent the new assignment from creating an infeasible trajectory optimization, the current time is reset to give each agent enough time to go to its new target.

B. Disconnected Communication Network

A typical auction algorithm cannot achieve a proper assignment when the communication network is disconnected. Without the assumption of a connected network, the proof of Proposition 2 breaks down. In fact, we cannot even guarantee that every target has at most one agent assigned to it since the bid of one agent cannot reach all of the other agents. However, we will show that using the MPC implementation contained in SATO allows the assignment algorithm to achieve an optimal assignment. First, we develop a condition that guarantees that two spacecraft, which cannot communicate, will not collide before the end of the MPC horizon.

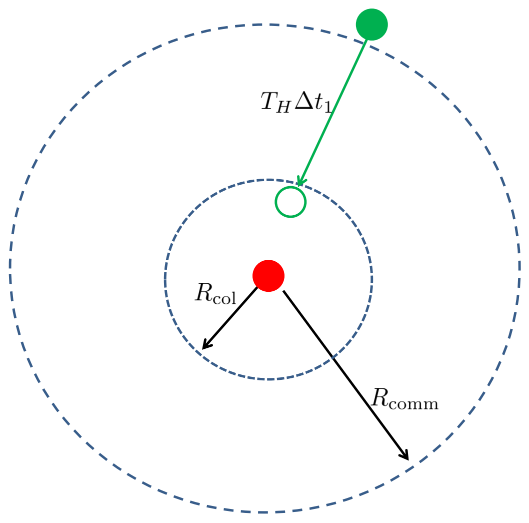
Proposition 9 (Detectable Collisions[30]). *If two agents cannot communicate, they will not collide before the end of the current horizon if*

$$R_{\text{comm}} \geq 2V_{\max} T_H \Delta t + R_{\text{col}} \tag{69}$$

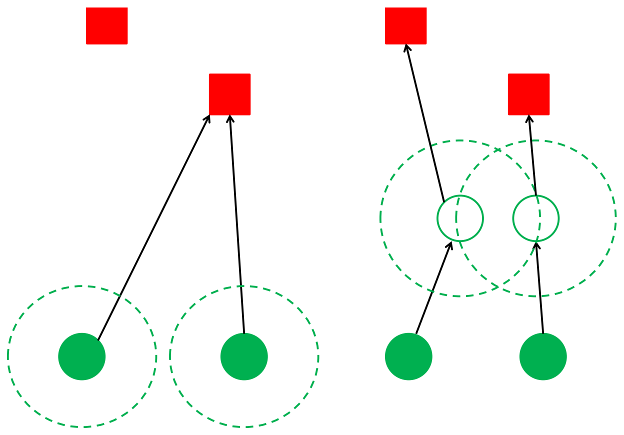
Proof. The amount of time in the current horizon is the number of time steps (T_H) multiplied by the time step size (Δt). Since the relative velocity is bounded by $2V_{\max}$, the maximum change in the relative distance between two agents is $2V_{\max} T_H \Delta t$. Therefore, this distance must be less than the difference between the communication radius (R_{comm}) and the collision radius (R_{col}). This establishes Eq. (69). □

This condition guarantees that any agent that could potentially cause a collision before the end of the MPC horizon is detected and therefore considered in the optimization. An illustration of a pair of spacecraft that violate this condition is shown in Fig. 3a.

While Proposition 9 was created in our prior work[30] with respect to trajectory optimizations, it also applies to the assignment part of SATO. This condition guarantees that before two agents, which are in disconnected communication networks and assigned to the same target, collide, a new assignment will be run with those agents being able to communicate. At this point, VSDAA guarantees that they will be assigned to different targets and that within their connected network, the assignment will be optimal. This idea allows each connected network to perform its own assignment, which will be optimal. If there is no conflict between the networks, no two agents in different networks have bid on the same target so all of the current bids are the highest bids and the solution is optimal. On the other hand, if there is a conflict, the conflicting agents will go to the same target resulting in them being able to communicate before they



(a) Illustration of an undetectable collision



(b) Resolution of an assignment conflict

Figure 3. Visualization of an undetectable collision and an assignment conflict being resolved

collide. At this point, they are on the same communication network and will resolve the conflict. Figure 3b shows an assignment conflict being resolved once the agents can communicate. When the agents are at the solid circles, they cannot communicate and they choose the same target. However, as they move towards the target, they reach the open circles and become connected. At this point, they will be assigned to different target, which resolves the conflict.

VI. Simulation Results

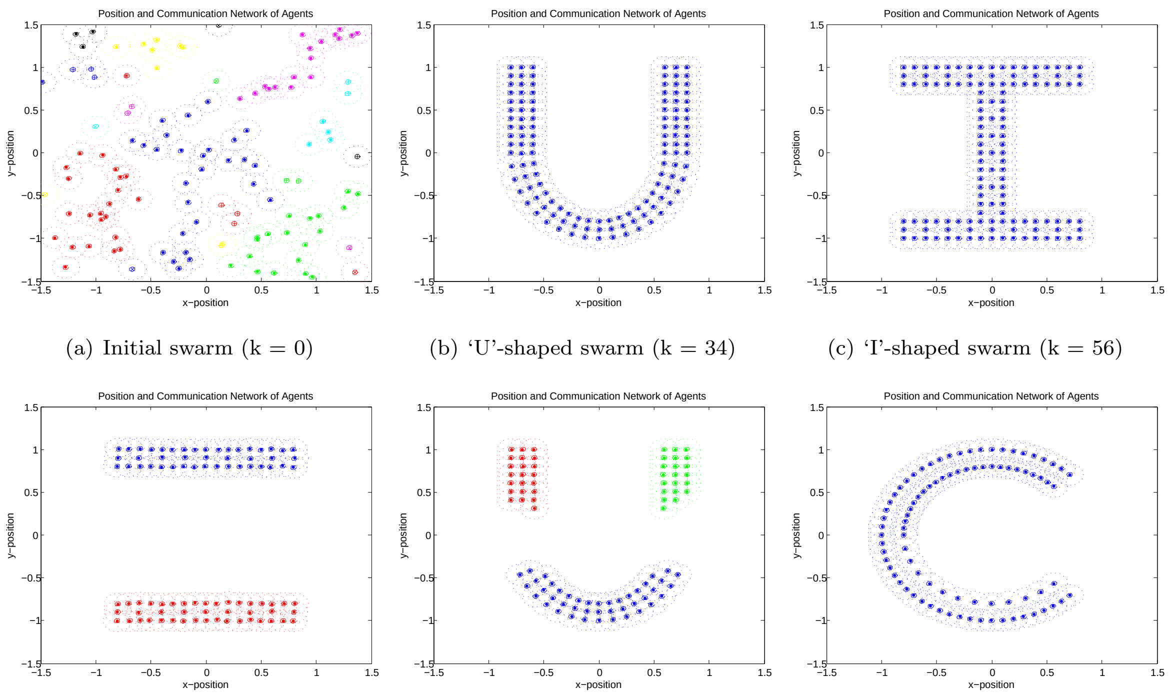
In this section, we apply SATO to swarms in two different dynamic environments. First, we apply SATO to a 2-D environment with double integrator dynamics. This simulation represents the most common problem in the swarm robotics community and can be used to compare SATO to the existing algorithms in that field. In the second simulation, we apply SATO to a swarm of spacecraft in 3-D. This simulation shows that SATO can be used even when the dynamics are nonlinear and state dependent.

A. 2-D Double Integrator Dynamics

In the 2-D double integrator simulation, a swarm of 123 agents is randomly initialized and undergoes four reconfigurations forming the letters ‘U’, ‘I’, ‘U’, and ‘C’, respectively. During the first reconfiguration, the communication network is initially disconnected since the swarm is randomly distributed. As the agents begin to form the ‘U’, they move closer to each other and the communication network becomes more connected. As the agents become more connected conflicting assignments are resolved and the swarm achieves the desired shape. After the second reconfiguration (‘I’-shape), a significant number of agents are lost causing the communication network to become disconnected. As the agents reconfigure from ‘I’ to ‘U’, they still believe there are 123 agents in the swarm, which results in some holes in the target shape. However, after completing the ‘U’, the agents adjust the number of targets in the assignment based on the number of bids they received while reconfiguring from ‘I’ to ‘U’. This is seen when the swarm successfully achieves a ‘C’ without any holes. The results of the simulation are shown in Figs. 4 and 5.

Figure 4 shows the swarm shape after each reconfiguration. Each agent is shown with a given marker shape and color, which denotes its communication network. Agents with the same shape and color are connected and agents with different shapes or colors are not connected. Additionally, the dotted circle around each agent represents half of the agents communication radius. Therefore, any agents whose communication circles touch can communicate with each other.

Figure 4a shows the initial swarm distribution and the disconnected communication networks due to the random distribution. Figure 4b shows the ‘U’-shaped swarm after the first reconfiguration. In Fig. 4c,



(a) Initial swarm (k = 0) (b) ‘U’-shaped swarm (k = 34) (c) ‘I’-shaped swarm (k = 56)

(d) ‘I’-shaped swarm after loss of agents (k = 57) (e) ‘U’-shaped swarm (k = 76) (f) ‘C’-shaped swarm (k = 104)

Figure 4. Various time instances in the UIUC reconfigurations

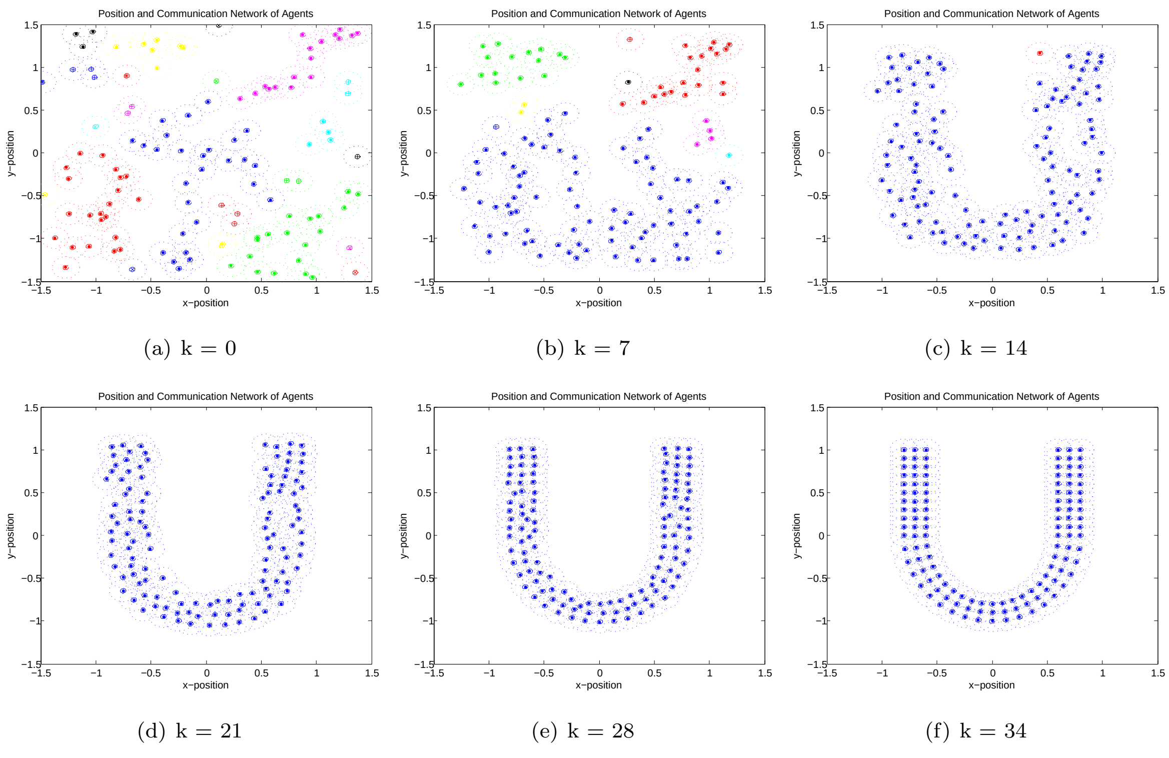
the ‘I’-shaped swarm that occurs after the second reconfiguration is shown. Figure 4d shows the ‘I’-shaped formation after many of the agents have been lost. This loss of agents results in two disconnected groups of agents (red and blue). In Fig. 4e, the swarm forms a ‘U’ shape but due to the loss of agents there are holes in the formation. Finally, Fig. 4f shows the ‘C’ shape swarm after the final reconfiguration. Although the ‘C’ is not complete due to the loss of agents, VSDAA adjusts the number of targets in the assignment so that there are no large gaps in the desired shape.

Figure 5 shows several time instances of the first reconfiguration. In this reconfiguration, a random swarm transforms into a ‘U’ shape. As with Fig. 4, the connected agents are shown with the same color and shape while the dotted circle represents half of the communication radius. Figure 5 shows the evolution of the communication network as the swarm converges to its desired shape.

Figure 5a shows the initial swarm with many, disconnected communication networks. In Fig. 5b, over half of the swarm is connected (blue) and the agents are roughly starting to form a ‘U’. In Fig. 5c, all but one of the agents are connected so the assignment is nearly complete and the agents are moving towards their targets. Figures 5d-f show the movement of the swarm once the agents are all connected and the assignment is complete. These figures show the trajectory optimization part of SATO as the agents reach their desired positions in Fig. 5f.

B. 3-D Relative Orbital Dynamics with J_2

In the 3-D spacecraft swarm simulation, the agents are randomly initialized and reconfigured to form a circle. In this simulation, the relative orbital dynamics with J_2[37] are used as the dynamics constraints when SATO calls Problems 6 and 7. The initial and terminal positions are random and circular, respectively, and the velocities are generated using J_2-invariant conditions from our prior work.[37] Additionally, the random initial positions and the target circle are both located in the $x - y$ plane, but the trajectories are allowed to move out of plane. We place the initial and terminal conditions with this constraint so that the plots can be more easily interpreted. However, SATO is equally effective when the initial and terminal positions also have out-of-plane components.



(a) k = 0 (b) k = 7 (c) k = 14

(d) k = 21 (e) k = 28 (f) k = 34

Figure 5. Various time instances in the reconfiguration from a random swarm to a 'U' shape

Figure 6 shows the reconfiguration of a swarm of 48 spacecraft from a random distribution to a circular formation. The reference orbit used for this simulation is a circular orbit with a semimajor axis of 6878 km, and an inclination of 45 degrees. The reconfiguration occurs in a quarter of an orbit, which is equal to about 24 minutes. The time step size (Δt) is one minute, the communication radius (R_{comm}) is 2 km, and the collision radius (R_{col}) is 50 m.

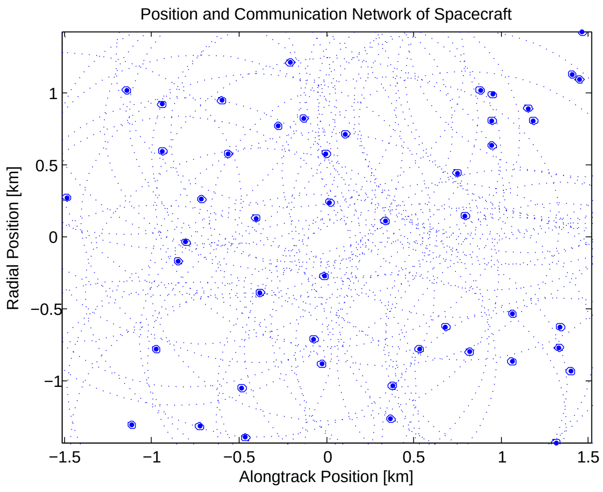
Figure 6a shows the initial swarm distribution where the spacecraft are randomly distributed. Figures 6b-d show the reconfiguration after 8, 16, and 24 minutes, respectively. In Fig. 6d, the swarm has achieved its desired formation of a circle. The results of this simulation show that SATO still achieves the optimal assignment and trajectory generation when the dynamics are as complicated as the relative dynamics of a spacecraft in low Earth orbit.

VII. Conclusion

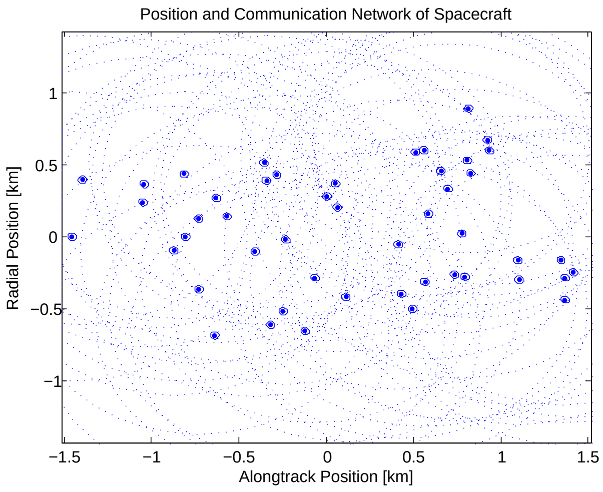
In this paper, we developed a new variable-swarm, distributed auction algorithm to solve the optimal assignment problem for a swarm of agents. The variable-swarm quality allowed the algorithm (Method 1) to adjust the number of targets in the assignment to match the number of agents in the swarm that are bidding on targets. This ultimately resulted in an assignment with the same number of agents and targets. This quality was especially useful when the number of agents in the swarm changed or when there were initially more agents than targets. Additionally, VSDAA (Method 1) can be implemented on a swarm with distributed communications and computations while still terminating in a finite number of iterations and achieving the optimal assignment.

We also introduced SATO (Method 3) as an algorithm that integrates the optimal assignment solver, VSDAA, and the trajectory optimizer, SCP. This integrated approach used MPC to update the optimal assignment and trajectory in real time so that changes in the state of each agent and changes in the communication network were included in the new solution. This allowed SATO to ultimately achieve the optimal assignment and trajectory even when the communication network was disconnected at the initial time or when errors prevented the agents from following their optimal trajectories.

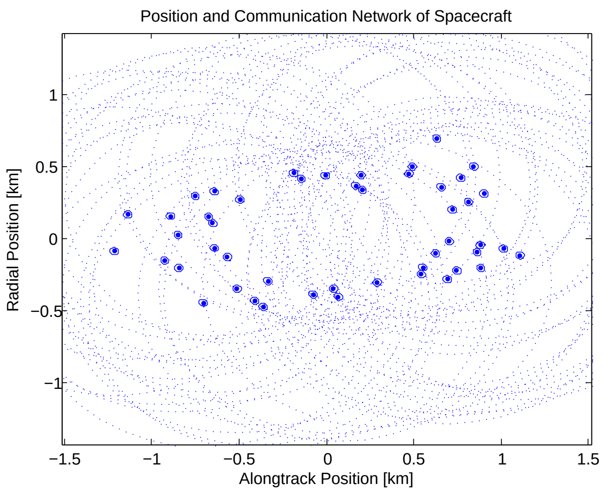
Finally, we showed in simulation that SATO can be implemented in a variety of situations. A 2-D



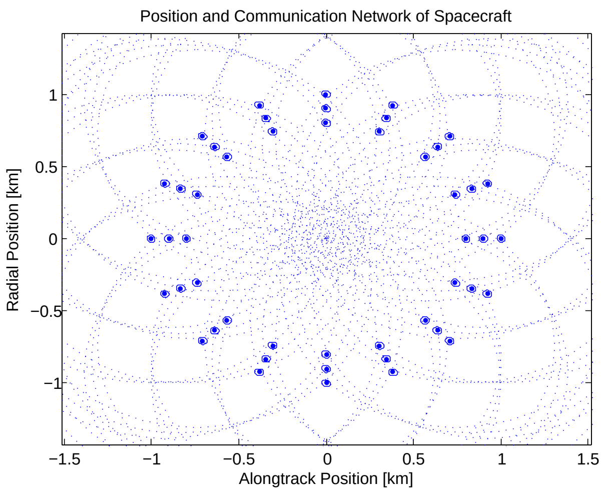
(a) k = 0



(b) k = 8



(c) k = 16



(d) k = 24

Figure 6. Various time instances in the reconfiguration of a swarm of spacecraft

simulation with double integrator dynamics showed that SATO can overcome an initially disconnected communicateion network and a loss of a significant number of agents, which caused the communication network to become disconnected. Additionally, a 3-D simulation using relative orbital dynamics showed that SATO can be run on a swarm of spacecraft even when the dynamics are complicated. The results of these simulations showed that SATO can be used to solve for the optimal assignment and trajectory in a variety of undesirable conditions.

Acknowledgments

This research was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. ©2015 California Institute of Technology. This work was supported by a NASA Office of the Chief Technologist Space Technology Research Fellowship. Government sponsorship acknowledged. Additional thanks to Saptarshi Bandyopadhyay for stimulating discussions and constructive comments.

References

[1]Murray, R. M., "Recent Research in Cooperative Control of Multivehicle Systems," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 51, 2007.

[2]Jadbabaie, A., Lin, J., and Morse, A. S., "Coordination of Groups of Mobile Autonomous Agents Using Nearest Neighbor Rules," *IEEE Transactions on Automatic Control*, Vol. 48, No. 6, 2003, pp. 2976–2984.

[3]Earl, M. G. and D'Andrea, R., "Iterative MILP Methods for Vehicle-Control Problems," *IEEE Transactions on Robotics*, Vol. 21, No. 6, 2005, pp. 1158–1167.

[4]Kingston, P. and Egerstedt, M., "Index-free multiagent systems: An Eulerian approach," *2nd IFAC Workshop on Distributed Estimation and Control in Networked Systems*, Annecy, France, 2010, pp. 215–220.

[5]Milutinović, D. and Lima, P., "Modeling and optimal centralized control of a large-size robotic population," *IEEE Trans. Robotics*, Vol. 22, No. 6, 2006, pp. 1280–1285.

[6]Cheah, C. C., Hou, S. P., and Slotine, J. J. E., "Region-based shape control for a swarm of robots," *Automatica*, Vol. 45, No. 10, 2009, pp. 2406–2411.

[7]Zhao, S., Ramakrishnan, S., and Kumar, M., "Density-based control of multiple robots," *Amer. Control Conf.*, San Francisco, USA, 2011.

[8]Hsieh, M. A., Kumar, V., and Chaimowicz, L., "Decentralized controllers for shape generation with robotic swarms," *Robotica*, Vol. 26, 2008, pp. 691–701.

[9]Alonso-Mora, J., Breitenmoser, A., Rufli, M., Siegwart, R., and Beardsley, P., "Image and Animation Display with Multiple Robots," *Int. Journal of Robotics Research*, Vol. 31, No. 6, May 2012, pp. 753–773.

[10]Scharf, D. P., Hadaegh, F. Y., and Ploen, S. R., "A Survey of Spacecraft Formation Flying Guidance and Control (Part I): Guidance," *Proceedings of the American Control Conference*, Jun. 2003, pp. 1733–1739.

[11]Scharf, D. P., Hadaegh, F. Y., and Ploen, S. R., "A Survey of Spacecraft Formation Flying Guidance and Control (Part II): Control," *Proceedings of the American Control Conference*, Jun. 2004, pp. 2976–2984.

[12]Alfriend, K. T., Vadali, S. R., Gurfil, P., How, J. P., and Breger, L., *Spacecraft Formation Flying: Dynamics, Control and Navigation*, Elsevier, Oxford, UK, 2009.

[13]Breger, L. and How, J. P., "Gauss's Variational Equation-Based Dynamics and Control for Formation Flying Spacecraft," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 2, Mar.-Apr. 2007, pp. 437–448.

[14]Vaddi, S. S., Alfriend, K. T., Vadali, S. R., and Sengupta, P., "Formation Establishment and Reconfiguration Using Impulsive Control," *Journal of Guidance, Control, and Dynamics*, Vol. 28, 2005, pp. 262–268.

[15]Campbell, M. E., "Planning Algorithm for Multiple Satellite Clusters," *Journal of Guidance, Control, and Dynamics*, Vol. 26, No. 5, 2003, pp. 770–780.

[16]Campbell, M. E., "Collision Monitoring within Satellite Clusters," *IEEE Transactions on Control Systems Technology*, Vol. 13, No. 1, 2005, pp. 42–55.

[17]Zanon, D. J. and Campbell, M. E., "Optimal Planner for Spacecraft Formations in Elliptical Orbits," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 1, 2006, pp. 161–171.

[18]Chung, S.-J., Bandyopadhyay, S., Chang, I., and Hadaegh, F. Y., "Phase Synchronization Control of Complex Networks of Lagrangian Systems on Adaptive Digraphs," *Automatica*, Vol. 49, No. 5, 2013, pp. 1148–1161.

[19]Hadaegh, F. Y., Chung, S.-J., and Manaroha, H. M., "On Development of 100-gram-class Spacecraft for Swarm Applications," *IEEE Systems Journal*, 2014, DOI:10.1109/JSYST.2014.2327972.

[20]Kuhn, H. W., "The Hungarian Method for the Assignment Problem," *Naval Research Logistics*, Vol. 2, No. 1-2, 1955, pp. 83–97.

[21]Bertsekas, D. P., "A New Algorithm for the Assignment Problem," *Mathematical Programming*, Vol. 21, 1981, pp. 152–171.

[22]Hung, M., "A Polynomial Simplex Method for the Assignment Problem," *Operations Research*, Vol. 31, 1983, pp. 595–600.

[23]Bertsekas, D. P. and Castanon, D. A., "Parallel Synchronous and Asynchronous Implementations of the Auction Algorithm," *Parallel Computing*, Vol. 17, 1991, pp. 707–732.

[24]Bertsekas, D. P., "Auction Algorithms for Network Flow Problems: A Tutorial Introduction," *Computational Optimization and Applications*, Vol. 1, 1992, pp. 7–66.

[25]Zavlanos, M. M., Spesivtsev, L., and Pappas, G. J., "A Distributed Auction Algorithm for the Assignment Problem," *IEEE Conference on Decision and Control*, Cancun, Mexico, December 2008.

[26]Choi, H.-L., Brunet, L., and How, J. P., "Consensus-Based Decentralized Auctions for Robust Task Allocation," *IEEE Transactions on Robotics*, Vol. 25, No. 4, August 2009, pp. 912–926.

[27]Richards, A., Kuwata, Y., and How, J., "Experimental Demonstration of Real-time MILP Control," *AIAA Guidance, Navigation, and Control Conference*, Austin, Texas, August 2003, AIAA-2003-5802.

[28]Ross, I. M. and Fahroo, F., "Legendre Pseudospectral Approximations of Optimal Control Problems," *Lecture Notes in Control and Information Systems*, Vol. 295, 2003.

[29]Schulman, J., Ho, J., Lee, A., Awwal, I., Bradlow, H., and Abbeel, P., "Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization," *Robotics: Science and Systems*, Berlin, Germany, June 2013.

[30]Morgan, D., Chung, S.-J., and Hadaegh, F. Y., "Model Predictive Control of Swarms of Spacecraft Using Sequential Convex Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 6, 2014, pp. 1725–1740.

[31]Bemporad, A. and Morari, M., "Robust Model Predictive Control: A Survey," *Robustness in Identification and Control*, 1999, pp. 207–226.

[32]Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O. M., "Constrained model predictive control: Stability and optimality," *Automatica*, Vol. 36, 2000, pp. 789–814.

[33]Turpin, M., Michael, N., and Kumar, V., "CAPT: Concurrent Assignment and Planning of Trajectories for Multiple Robots," *The International Journal of Robotics Research*, Vol. 33, No. 1, 2014, pp. 98–112.

[34]Boyd, S. and Vandenberghe, L., *Convex Optimization*, Cambridge Univ. Press, Cambridge, U.K., 2004.

[35]Ruszczynski, A., Princeton University Press, Princeton, NJ, 2006.

[36]Jongen, H. T., Meer, K., and Triesch, E., Kluwer Academic, Norwell, MA, 2004.

[37]Morgan, D., Chung, S.-J., Blackmore, L., Acikmese, B., Bayard, D., and Hadaegh, F. Y., "Swarm-Keeping Strategies for Spacecraft Under J_2 and Atmospheric Drag Perturbations," *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 5, 2012, pp. 1492–1506.