Abstract—This paper presents an online optimization-based approach to compute trajectories to enable substitution of robots in formation-based deployments with durations that exceed the energy capacity of individual systems. The proposed algorithm computes trajectories in a multi-robot context to ensure a collision-free exchange, even where congestion is a concern. The quality of the resulting trajectories is determined by the amount of time spent deviating from the original plan while maintaining collision-free, speed-limited polynomial splines. The algorithm is shown through simulation and experiments to be viable with average deviation time gaps of less than 20 seconds and average computation times of under 3 minutes for the presented scenarios with varying numbers of robots and deployment specifications.

I. INTRODUCTION AND RELATED WORK

We pursue long-duration operation of aerial vehicles (quadrotors) for a theatrical endeavor (Fig. 1) that requires robots to fly in regular formations, both for visual appeal and coordinated control, with the duration of each performance extending beyond the lifetime of a single vehicle battery. We address the challenge of determining how to ensure uninterrupted execution of the desired performance trajectories given the limited energy capacity of the chosen aerial platforms.

The problem of recharging or refueling during persistent task handling is frequently addressed through scheduling and goal-reassignment [1]–[3]. These methods are well suited to managing energy capacity among vehicles when performing decoupled assignments via robot exchange, task reordering, or by changing time delays between tasks to maintain collision-free trajectories [4]–[6]. The use of reserved passing lanes and reservation systems [7] is employed to find collision-free routes to shared resources such as battery charging stations over extended operations.

In the relevant theatrical application, tasks constitute precisely spaced and timed trajectories that are executed in order to convey a visual narrative, thus precluding interruptions due to changes in task timing or order, altering of a formation’s end goal to a charging station, or delaying execution of a flight to wait for a robot to finish recharging. Altering individual trajectories within a formation is also detrimental to the visual appeal of the performance.

Therefore, we pursue a strategy to exchange active robots with depleted batteries for robots with batteries that are fully charged in a manner that minimizes changes to the formation. The requirement of close formation flight limits the availability of collision-free space for the exchange of individual robots. Further, any exchange of robots must avoid sizable gaps in the formation shape in order to prevent disrupting the visual continuity of the formation flight. Local collision avoidance methods for multiple vehicles in close configurations [8], [9] is undesirable as the resulting switching trajectories lead to changes in the formation structure. However, in the proposed application, the lifespan of the robot batteries affords sufficient time to seek globally optimal trajectories.

We formulate the problem of finding “swapping” trajectories as a Mixed Integer Quadratic Program (MIQP) that minimizes the length of time a position in the formation is unoccupied. The proposed approach computes optimal trajectories that minimize the overall exchange duration while preserving dynamic feasibility and avoiding inter-robot collisions.

The algorithm presented in this paper builds on approaches that address the rapid generation of general collision-free trajectories. Specifically, we implement an MIQP formulation to generate polynomial trajectories obeying smoothness constraints [10] and extend the collision constraints for axis-aligned cubic bounding regions [11] to handle arbitrary polyhedron to polyhedron intersections when computing coupled trajectories. We incorporate iterative generation of constraints [12] as they are discovered over repeated searches and select optimization parameters accordingly. This technique permits efficient iteration over time as well as collision parameters in order to efficiently search the space of possible problem configurations and increase the flexibility and generality of the proposed optimization formulation.
Fig. 2. Illustration of departure (red) and arrival (blue) trajectories and times, relative to the desired trajectory (black).

II. PROBLEM DEFINITION

Given a team of vehicles moving in formation, we seek to ensure persistent operation by monitoring energy capacity and preempting depletion via robot substitution by generating two trajectories, \( \gamma_d \) and \( \gamma_a \), that describe the motions of the departing and arriving robots, respectively. The controller provides the time at which a change should occur, \( t_f \), and the original trajectory the outgoing robot will follow if no change is necessary, \( \gamma_f \). Given the position of the recharging station, \( l_s \), we compute a feasible set of trajectories such that \( \gamma_d \) departs from \( \gamma_f \) as close to \( t_f \) as possible in as short a duration as possible and, similarly, \( \gamma_a \) arrives at \( \gamma_f \) with minimal deviation and duration. Here, feasible trajectories are those that avoid collisions between robots and the environment and constrain accelerations to the performance limitations of the robot.

The four time parameters we seek to minimize here are defined as shown in Fig. 2. The approach gap, \( d_{ga} \), indicates the amount of time between \( t_f \) and the planned arrival of the incoming robot. Similarly, the departure gap, \( d_{gd} \), indicates the time between \( t_f \) and the departure of the outgoing robot. The approach and departure durations, \( d_a \) and \( d_d \), simply refer to the duration of trajectories \( \gamma_a \) and \( \gamma_d \), respectively. For convenience, we define the time at which the incoming robot arrives at \( \gamma_f \), \( t_a \), as in (1) and the time at which the outgoing robot departs \( \gamma_f \), \( t_d \), as in (2).

\[
t_a = t_f + d_{ga} \tag{1}
\]
\[
t_d = t_f - d_{gd} \tag{2}
\]

The continuity of these trajectories is preserved as described in Sect. III-A.1. The trajectories are defined as a linear combination of basis functions, so determining the optimal approach and departure trajectories amounts to finding the weights for each basis function such that the resulting trajectory minimizes the objective, defined for this problem as a minimization of trajectory roughness in Sect. III-A.

For this problem, collision avoidance is addressed by ensuring that a robot’s bounding region, represented as a polyhedron, does not intersect with any other bounding region. Collisions with the bounds of the area of operation (arena) are similarly modeled to ensure that the polyhedron is constrained to exist entirely within a region. The polyhedron is allowed to move with the robot, but is assumed to have a fixed orientation, as the bounding region will not vary significantly with the orientation of the robot.

The position and heading and their higher derivatives for all robots at all times are provided to the solver as formation trajectories. Formation trajectories are described by time-varying polynomials specified relative to a virtual formation leader. A robot’s state in the formation, or “shape” reference frame, is described as a \( 4 \times 1 \) shape vector \( s(t) \) containing the local position coordinates \( x \), \( y \), and \( z \), as well as the relative heading, \( \psi \), of robots in the local shape reference frame as \( s_i = \begin{bmatrix} x(t), y(t), z(t), \psi(t) \end{bmatrix}^T \) [5]. The virtual leader of the formation follows a time-varying polynomial trajectory in the inertial frame, \( C(t) \in \mathbb{R}^3 \), with orientation described by the time-varying rotation matrix \( \mathbf{R}(t) \in \mathbb{SO}(3) \) [13]. A trajectory in the world frame, \( \gamma(t) \), for robot \( i \) is then given by rotating \( s(t) \) by the group’s inertial rotation and offsetting the resulting rotated position from the reference trajectory as:

\[
\gamma_i(t) = \begin{bmatrix} C(t) + \mathbf{R}(t) s_i^w(t) \end{bmatrix} . \tag{3}
\]

III. SUBSTITUTION TRAJECTORIES

The presented algorithm is formulated as a multi-stage solver with an inner loop defined by a Mixed-Integer Quadratic Program (MIQP) and an outer loop that iterates over a set of four parameters. The MIQP solver is restricted to update within a finite time interval and, if unable to converge, results in a parameter update via the outer loop. The parameters explored in the outer loop serve to adjust the start and end times of the approach and departure trajectories. The construction of the program and its solution approach is detailed in Sect. III-A. The form of the parameters and iteration approach is described in Sect. III-B.

A. MIQP Formulation

The MIQP formulation seeks to minimize the roughness of the approach and departure trajectories in the presence of polyhedral obstacles [10], [11]. The objective (4)

\[
\min \int_{t_d}^{t_f+d_d} \left\| \frac{d^4 \gamma_d(t)}{dt^4} \right\|^2 dt + \int_{t_a-d_a}^{t_a} \left\| \frac{d^4 \gamma_a(t)}{dt^4} \right\|^2 dt \tag{4}
\]

takes the form of an integral of the square norm of the snap of the trajectories, here represented as a linear combination of basis functions (5).

\[
\gamma_i(t) = \sum_{i=0}^{k} [p_{ax,i}, p_{ay,i}, p_{az,i}, p_{aw,i}] b_i(t) . \tag{5}
\]

This objective is further simplified by choosing basis functions with fourth order derivatives as Legendre polynomials. Here, we denote \( p_{aq,i} \) as the weight of the \( i \)th basis function, \( b_i(t) \), for each \( q \) dimension of the arrival trajectory. A representation using \( p_{dq,i} \) for the departure trajectory is similarly defined. The updated objective is

\[
\min \sum_{i=4}^{k} \sum_{q \in \{x, y, z, \psi\}} \left( p_{aq,i}^2 + p_{dq,i}^2 \right) \frac{2}{2(i-q) + 1} dt . \tag{6}
\]
The first four basis functions are chosen to be $b_i(t) = t^i$ for $i \in \{0, 1, 2, 3\}$, each of which has no contribution to the objective. The subsequent functions are increasing fourth integral Legendre polynomials of the following form:

$$ b_i(t) = \int_0^t \int_0^t \int_0^t \lambda_{i-4}(t), \tag{7} $$

where $\lambda_i(t)$ is the $i^{th}$ Legendre polynomial.

1) Initial and Final Conditions: To ensure continuity relative to the formation trajectory, the state of the outgoing robot at departure time $t_d$ must equal the initial state of the departure trajectory. Similarly, the state of the incoming robot at arrival time $t_a$ must equal the end state of the arrival trajectory. These constraints are encoded as

$$ \frac{d^j \gamma_a(t)}{dt^j} \bigg|_{t=t_a} = \frac{d^j \gamma_f(t)}{dt^j} \bigg|_{t=t_a} \quad \forall j \in \{0, 1, 2\}, \tag{8} $$

$$ \frac{d^j \gamma_d(t)}{dt^j} \bigg|_{t=t_d} = \frac{d^j \gamma_f(t)}{dt^j} \bigg|_{t=t_d} \quad \forall j \in \{0, 1, 2\}, \tag{9} $$

with derivatives taken across each of the four dimensions. The remaining endpoints of the trajectories coincide with the recharging station location, $l_s$, and yaw, $\gamma_s$, so that the robots are enforced to be static at these points:

$$ x_a(t_a - d_a) = l_s \tag{10} $$

$$ \psi_a(t_a - d_a) = \psi_s \tag{11} $$

$$ x_d(t_d + d_d) = l_s \tag{12} $$

$$ \psi_d(t_d + d_d) = \psi_s \tag{13} $$

$$ \frac{d^j \gamma_a(t)}{dt^j} \bigg|_{t=t_a-d_a} = 0 \quad \forall j \in \{2, 3\}, \tag{14} $$

$$ \frac{d^j \gamma_d(t)}{dt^j} \bigg|_{t=t_d+d_d} = 0 \quad \forall j \in \{2, 3\}, \tag{15} $$

where $x_a$ is the position component and $\psi_a$ is the yaw component of $\gamma_a$. For simplicity, (10)-(15) show the outgoing robot being sent to the recharging station from which the incoming departs.

2) Higher-order Constraints: Inequality constraints on higher-order terms help ensure that the solution trajectory is dynamically feasible. To avoid imposing non-linear constraints on the problem, these limits are applied individually to all linear spatial dimensions

$$ -\alpha_{qj} \leq \frac{d^j \gamma_a(t)}{dt^j} \bigg|_{t=t_i} \leq \alpha_{qj} \quad \forall j \in \{1, 2\}, q \in \{x, y, z, \psi\}, \tag{16} $$

$$ -\alpha_{qj} \leq \frac{d^j \gamma_d(t)}{dt^j} \bigg|_{t=t_i} \leq \alpha_{qj} \quad \forall j \in \{1, 2\}, q \in \{x, y, z, \psi\}. \tag{17} $$

Here, $\alpha_{qj}$ represents the $j^{th}$ order limit imposed on dimension $q$ and $\gamma_{aq}(t)$ is the trajectory $\gamma_a$ evaluated for dimension $q$ at time $t$. These constraints are applied at intervals, $t_i$, along both trajectories. In practice, we distribute $t_i$ evenly across each $\{t_a - d_a, t_a\}$ and $\{t_d, t_d + d_d\}$. 3) Collisions: Collision constraints are introduced as bounding regions approximated by closed, convex polyhedra with collisions determined by a point’s location relative to the half-space defined by each face plane [11]. Collisions are encoded in the MIQP in three categories: robot-boundary collisions, robot-obstacle collisions, and robot-robot collisions. Note that the constraints described here as applied to the outgoing robot are similarly applied to the incoming robot.

The first collision category involves the interaction between a robot’s bounding polyhedron and the arena bounds. These constraints enforce that all points belonging to the robot extents exist inside each arena polyhedron face. For a single plane, this constraint takes the form

$$ \hat{n} \cdot (x_d(t) + v_{d,i}) \geq s \quad \forall i \in \{1, \ldots, n_{d,p}\}, \tag{18} $$

where $v_{d,i}$ is the relative position of the $i^{th}$ point of a set of $n_{d,p}$ points defining the bounding region of the outgoing robot. Enforcing this constraint at discrete intervals over all possible $t$ ensures that the robot bounding region always exists outside of the plane defined by the normal, $\hat{n}$, and the scalar, $s$. Doing the same for all planes defined by the arena bounds confines the robot to the area. The assumption of constant orientation yields a reduced set of constraints

$$ \hat{n} \cdot (x_d(t) + v_{d,\text{min}}) \geq s. \tag{19} $$

Here, $v_{d,\text{min}}$ designates the relative position of the point on the robot bounding polyhedron closest to the plane. Given the constant orientation assumption, this position is readily computed for every plane at every instance in time at which this constraint is enforced. As such, we can reduce the number of constraints required per plane from $n_{d,p}$ to one.

Constraints for robot-obstacle collisions enforce that one polyhedron exists outside of another. For each plane of the robot-obstacle pair, the applied constraint takes the same form as (19). As any given point cannot exist outside of all planes defined by the faces of a closed, convex polyhedron, it is not necessary for every constraint to be active at the same time in order for a collision to be excluded. Constraints are therefore implemented using integer variables that allow only a subset to be active at any one time. For a single obstacle, the constraints are:

$$ \hat{n}_{o,i} \cdot (x_d(t) + v_{d,\text{min}}) \geq s_{o,i} + \hat{n}_{o,i} \cdot x_o(t) - M b_{o,i} \quad \forall i \in \{1, \ldots, n_{o,f}\} \tag{20} $$

$$ \hat{n}_{d,i} \cdot (x_o(t) + v_{o,\text{min}}) \geq s_{d,i} + \hat{n}_{d,i} \cdot x_d(t) - M b_{d,i} \quad \forall i \in \{1, \ldots, n_{d,f}\} \tag{21} $$

$$ \sum_{i=1}^{n_{o,f}} b_{o,i} + \sum_{j=1}^{n_{d,f}} b_{d,j} \leq n_{o,f} + n_{d,f} - 1 \tag{22} $$

$$ b_{o,i} \in \{0, 1\} \quad \forall i \in \{1, \ldots, n_{o,f}\} \tag{23} $$

$$ b_{d,j} \in \{0, 1\} \quad \forall i \in \{1, \ldots, n_{d,f}\}. \tag{24} $$

In this formulation, the outgoing robot’s bounding polyhedron is defined by $n_{d,f}$ planes, where the $i^{th}$ plane is defined
by a normal \( \hat{n}_{a,i} \) and scalar \( s_{d,i} \). The obstacle polyhedron, in turn, is defined by \( n_{o,f} \) planes, with normals \( \hat{n}_{o,i} \) and scalars \( s_{o,i} \). The first pair of equations, (20) and (21), constrain robot points to lie outside of obstacle planes and obstacle points to remain outside of robot planes, respectively. The binary variables, \( b_{a,i} \) and \( b_{d,i} \), coupled with the large positive constant \( M \) dictate the enforcement constraint for plane \( i \) [14]. For example, if \( b_{a,i} = 1 \), \( x_a(t) \) can take any value without violating (20). Given this formulation, (22-24) ensure that when at least one of these constraints is enforced, a collision is properly excluded.

For the defined problem, obstacles in the MIQP formulation include any robots with fixed trajectories at a time, \( t \), excluding the outgoing and incoming robots, for the trajectories being solved in the MIQP formulation. Thus, it is possible for the incoming robot to be interpreted as an obstacle with a fixed \( x_o(t) \) if \( t > t_a \) (i.e. outside of the trajectory duration).

Robot-robot collisions, on the other hand, occur when both the outgoing and incoming trajectories are evaluated within their durations. The resulting set of constraints are of similar form to (20)-(24). However, in the case of robot-robot collisions, both \( x(t) \) variables contain the basis function weights:

\[
\hat{n}_{a,i} \cdot (x_a(t) + v_{a,\min} \hat{n}_{a,i}) \geq s_{a,i} + \hat{n}_{a,i} \cdot x_a(t) - M b_{a,i} \\
\forall i \in \{1, \ldots, n_a, f\} \tag{25}
\]

\[
\hat{n}_{d,i} \cdot (x_d(t) + v_{d,\min} \hat{n}_{d,i}) \geq s_{d,i} + \hat{n}_{d,i} \cdot x_d(t) - M b_{d,i} \\
\forall i \in \{1, \ldots, n_d, f\} \tag{26}
\]

\[
\sum_{i=1}^{n_a, f} b_{a,i} + \sum_{j=1}^{n_d, f} b_{d,j} \leq n_{a,f} + n_{d,f} - 1 \\
b_{a,i} \in \{0, 1\} \quad \forall i \in \{1, \ldots, n_a, f\} \tag{27}
\]

\[
b_{d,i} \in \{0, 1\} \quad \forall i \in \{1, \ldots, n_d, f\} \tag{28}
\]

By repeating these constraints for discrete values of \( t \) over all values for which both the arrival and departure trajectories are active, we ensure that the resulting trajectories do not contain collisions between the outgoing and incoming robots. However, arbitrary addition of a large number of complex constraints significantly increases the computation time for each inner loop cycle. To mitigate this issue, the proposed algorithm iteratively computes solutions, checks for collisions, and adds collision constraint at values of \( t \) for which collisions are found in a manner similar to techniques that iteratively add collision constraints to a Sequential Convex Program [12] with a different interpretation of trajectories and obstacle bounds. This approach yields collision-free, coupled trajectories for scenarios with the level of congestion arising in our application. However, this approach requires more time than the standard application of constraints at discrete time intervals given a large number of obstacles.

### B. Parameter Iteration

The inner loop, described in Sect. III-A, assumes predefined values of \( d_{ga}, d_{gd}, d_a, \) and \( d_d \). Should the initial choice of these values fail to produce a viable solution within the

<table>
<thead>
<tr>
<th>Algorithm 1: Substitution Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Swap time ( t_f ), Duration increment size ( dt_d ), Gap increment size ( dt_g ), Gap closeness parameter ( g_c ), Formation trajectories ( F ), Polyhedral bounds ( B ), Polynomial order ( n ), Speed limit ( v_{\max} ), MIQP computation time limit ( t_{miqp} ), Inner loop computation time limit ( t_{solver} )</td>
</tr>
<tr>
<td><strong>Output:</strong> Arrival trajectory ( \gamma_{a} ), Departure trajectory ( \gamma_{d} ) begin</td>
</tr>
<tr>
<td>( d_{gd} = 0 )</td>
</tr>
<tr>
<td>( d_d = d_t )</td>
</tr>
<tr>
<td>( d_{ga} = 0 )</td>
</tr>
<tr>
<td>( d_a = d_t )</td>
</tr>
<tr>
<td>success = false</td>
</tr>
<tr>
<td>while ( \neg success ) do</td>
</tr>
<tr>
<td>Collision set, ( C \leftarrow 0 )</td>
</tr>
<tr>
<td>Computation time, ( t_{compute} = 0 )</td>
</tr>
<tr>
<td>setStartTime(( \gamma_{a}, t_f + d_{ga} - d_a ))</td>
</tr>
<tr>
<td>setEndTime(( \gamma_{a}, t_f + d_{ga} ))</td>
</tr>
<tr>
<td>setStartTime(( \gamma_{d}, t_f - d_{gd} ))</td>
</tr>
<tr>
<td>setEndTime(( \gamma_{d}, t_f - d_{gd} + d_d ))</td>
</tr>
<tr>
<td>while ( \neg success ) do</td>
</tr>
<tr>
<td>( [\gamma_{d}, \gamma_{a}, t_{solution}, success] = )</td>
</tr>
<tr>
<td>MIQPsolver(( \gamma_{d}, \gamma_{a}, B, n, F, C, t_{miqp} ))</td>
</tr>
<tr>
<td>if checkForCollisions(( \gamma_{d}, \gamma_{a}, B, F )) then</td>
</tr>
<tr>
<td>( nC = \text{findFirstCollision}(\gamma_{d}, \gamma_{a}, B, F) )</td>
</tr>
<tr>
<td>( C \leftarrow C \cup nC )</td>
</tr>
<tr>
<td>( success = false )</td>
</tr>
<tr>
<td>( t_{compute} += t_{solution} )</td>
</tr>
<tr>
<td>if ( t_{compute} &gt; t_{solver} ) then</td>
</tr>
<tr>
<td>( \text{break} )</td>
</tr>
<tr>
<td>if ( \neg success ) then</td>
</tr>
<tr>
<td>if (</td>
</tr>
<tr>
<td>( d_d += dt_d )</td>
</tr>
<tr>
<td>( d_a += dt_d )</td>
</tr>
<tr>
<td>if countCollisions(( C, \gamma_{d} )) &gt; 0 then</td>
</tr>
<tr>
<td>if aveCollisionTime(( C, \gamma_{d} )) &lt; ( t_d + g_c d_d ) then</td>
</tr>
<tr>
<td>( d_d += dt_d )</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>( d_{gd} += dt_g )</td>
</tr>
<tr>
<td>if countCollisions(( C, \gamma_{a} )) &gt; 0 then</td>
</tr>
<tr>
<td>if aveCollisionTime(( C, \gamma_{a} )) &gt; ( t_a - g_c d_a ) then</td>
</tr>
<tr>
<td>( d_a += dt_d )</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>( d_{ga} += dt_g )</td>
</tr>
<tr>
<td>return ( \gamma_{a}, \gamma_{d} )</td>
</tr>
</tbody>
</table>
required time limit, a new set of values must be selected. To avoid searching through all possible permutations of these parameters, the proposed algorithm analyzes the result of the MIQP solver and increments parameters as a function of the inner-loop optimization progression (and subsequent failure to converge).

The process to efficiently iterate through permutations of the parameters is shown in Alg. 1. The approach hinges entirely on the ability of the MIQP solver to generate a solution. In practice, this function is a CPLEX class structure that attempts to solve the problem described in Sect. III-A relative to the start and end times of $\gamma_a$ and $\gamma_d$, using the bounds defined by $B$, given a solution of order $n$, avoiding collisions with robots following the formation trajectories $F$, avoiding previously detected collisions in $C$, within the given time limit, $t_{miqp}$. Once a solution is provided, it is checked for any new collisions that have not already been added to $C$. If any are found, the first is added to $C$ and the process repeats. The inner while loop exits if a solution is found with no collisions, if the time spent in this loop exceeds $t_{solver}$, or if the solver determines that the problem is infeasible.

Based on the outcome of the inner while loop, logical conditions determine how the parameters are adjusted. If no collisions are detected but a solution is not found, then the durations are not long enough to account for the velocity and acceleration limits. As the trajectories are designed to directly exchange robot locations, it is sufficient to increment both durations simultaneously until a feasible solution can be found. If collisions are found, then the gap parameter, $d_{ga}$, and $d_{gd}$, are incremented only if the average time of collision occurs near the gap, $t_a$ for the incoming robot and $t_g$ for the outgoing robot. Closeness, in this case, is determined by the parameter $g_c$, for which a value of 0.25 is found to work in practice. The reason for the increase in gap time is to ensure that the incoming and outgoing robots have enough room to enter and exit $\gamma_f$ without colliding. Collisions outside of this regime can be handled by incrementing the durations.

IV. Simulation and Experimental Evaluation

The approach is tested on three hour long formation flight trajectories. The trajectories describe random transitions between line and triangle shape formations consisting of three robots. The formation is also driven through randomly generated lateral transitions and rotations at accelerations up to the physical limitations of the quadrotors. Substitution times, $t_f$, are chosen along these trajectories such that no exchanges occur simultaneously and no robot has a continuous flight time greater than its’ capacity. This results in a total of 138 substitutions across the three flights. Substitutions are computed in simulation and evaluated for computation time and the values of the four parameters listed in Sect. III-B. Simulations are evaluated on a Dell PowerEdge T420 with an Intel Xeon E52450 v2 2.50GHz processor and 16 GB of RAM using the CPLEX convex optimization library to solve each MIQP.

The parameters chosen for testing are: polynomial order $n = 8$, MIQP timeout $t_{miqp} = 3$ s, collision solver timeout $t_{solver} = 120$ s, max speed $v_{max} = 2$ m/s, max acceleration $a_{max} = 1$ m/s$^2$, duration step $d_t = 1$ s, and time gap step $d_g = 0.5$ s. The robot bounding volumes are chosen to be vertically oriented octagonal prisms with a width of 0.5 m and a height of 1.2 m. The robots operate in an arena of $3.75 \times 5 \times 4.5$ m. The maximum lateral speed experienced by robots performing the original formation trajectory varied among robots, between $1.6 - 1.8$ m/s.

Results from the experiments shown in Fig. 3 suggest that, under these conditions, trajectories are computed on average within 3 minutes. Although there is a realistic possibility of certain transition states requiring up to 10 minutes of computation time, at least 88% of the time a solution is available within 6 minutes, which is more than sufficient for quadrotor platforms with a battery life of 10 minutes.

In Fig. 4, it is evident that the resulting gaps and trajectory durations are under 20 s , with 87% of solutions presenting approach durations under 10 s. From the graphs, we observe a total deviation from the original formation specification under 4 s for 82% of trials. Similarly, any substitution robot will expend at most 16 seconds of excess energy in its approach and departure 86% of the time.

Table I provides an analysis of the parameter data. The

<table>
<thead>
<tr>
<th>Table I Analysis of MIQP Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approach Gap (s)</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>mean</td>
</tr>
<tr>
<td>std dev</td>
</tr>
<tr>
<td>max</td>
</tr>
<tr>
<td>min</td>
</tr>
</tbody>
</table>

Fig. 3. Cumulative percentage of computation time.

Fig. 4. Cumulative percentage of parameter values.
maximum bounds of the parameters are within reasonable values for robots that have a battery life of 10 minutes. In particular, we note that the maximum approach and departure duration of 16 seconds as well as the average total gap of roughly 3 seconds.

We evaluate the applicability of this approach experimentally using a team of four quadrotors. Over a representative seven minute long theatrical performance, a charging robot exchanges places in turn with members of a three-robot team flying in formation within a motion capture arena. The formation trajectories considered in this performance consist of continuous transitions between line and triangle shapes, while constantly translating across the flight volume in three dimensions while the formation itself is directed to smoothly spin at randomly chosen rates about the inertial vertical axis. Still images from the experiment flight are shown in Fig. 5.

V. CONCLUSION AND FUTURE WORK

The optimization-based approach detailed in this work enables the online substitution of robots moving in close formation though the generation of trajectories that exchange robots with depleted energy stores for robots with full reserves and minimizes the amount of time spent deviating from the original plan while obeying collision and actuator constraints. While the solver cannot produce fast solutions for any given problem, it produces high-quality solutions on an application-appropriate time scale allowing us to use this approach reliably in an online setting. To reduce computation time, we will explore further ways in which to draw intuition from failed solutions to guide the search for optimal parameters, as well as investigate the applicability of algorithms such as a backtracking line search for parameter optimization. For the cases in which a solution cannot be found in a timely manner, we are interested in developing both “safety” behaviors, which may be suboptimal but can be employed using limited computation time in the event that we cannot find optimal online transitions, as well as providing optimal solutions in the event that we are able to relax the constraint of altering formation trajectories. Incorporating energy-based optimization considerations into the original generation of formation trajectories may serve to allow formation shapes to remain close to the originally intended plans but permit easy online swapping calculations to quickly and fluidly enable persistent theatrical performances.

REFERENCES