Intelligent Small Satellite Swarm Control System for Avoiding in Space Debris

Evan Finnigan, Brandon Liu, Dick Stottler
Stottler Henke Associates, Inc.
1650 S Amphlett Blvd # 300
San Mateo, CA 94402
efinnigan, bliu, stottler @stottlerhenke.com

Abstract—**This paper describes an intelligent software system for controlling satellites in a swarm to avoid space debris. This software system, called Coordinated Autonomous Debris Avoidance (CADANCE), was developed with funding and direction from NASA. CADANCE uses trajectory optimization and case-based reasoning (CBR) to create a sequence of thrust controls for every satellite in the swarm, based on incoming Conjunction Data Messages (CDMs) and mission objectives. CADANCE uses a trajectory optimization algorithm called Particle Swarm Optimization (PSO) to plan the sequence of maneuvers required to execute the high-level plan. This approach can plan maneuvers to avoid space debris for satellites in diverse swarm and formation types, including but not limited to massive internet provider swarms, science missions, Earth observation swarms, and trailing formations. The main significance of this work is to prove, in simulation, the feasibility of autonomously controlling satellites in a swarm to avoid conjunctions, so we tested CADANCE in eight unique simulated scenarios with CDM data from real satellites. The scenarios were diverse, with different swarm types, mission constraints, and conjunction risks. We used a different satellite simulation software for PSO then we used for testing to avoid bias. CADANCE could find a conjunction avoidance maneuver for every scenario that reduced the Probability of Collision (P_c) to below a given threshold while obeying all mission constraints. Additionally, CADANCE's trajectory optimizer was always capable of planning a series of maneuvers to match the trajectory requested by the High-Level Planner to within 5 meters. CADANCE is computationally efficient, which will enable it to run on radiation- tolerant computers in space.**

TABLE OF CONTENTS

1. INTRODUCTION

Safely managing a swarm of SmallSats operating in Low Earth Orbit (LEO) is a challenging task that requires measuring the risk of collision with nearby objects and planning conjunction avoidance maneuvers to minimize those risks. At the same time, the conjunction avoidance maneuvers cannot increase the risk of collision with objects in nearby orbits. Additionally, to avoid mission impacts, conjunction avoidance planning should comply with mission requirements when possible. For example, the mission may require that the swarm provide coverage to view an event happening at a certain location at a certain time or may require consistent coverage of a region.

The process of managing satellite conjunctions starts when a satellite Owner/Operator (O/O) receives a Conjunction Data Message (CDM). A CDM is a standard format for exchanging spacecraft conjunction information and contains the position and velocity of both the primary and secondary object involved in the possible conjunction as well as the Probability of Collision (P_c). The CDM also contains information about the uncertainty of the position of both the primary and secondary satellites in the form of covariance volumes. Currently, managing swarms of SmallSats is a primarily manual process, where satellite operators receive a Conjunction Data Message (CDM), decide whether to avoid the conjunction, create a plan to avoid the conjunction with the owner of the other satellite, provide that maneuver to the Space Force for validation, and then execute the plan. This manual process is feasible now, but some satellite operators are planning on placing thousands of satellites into LEO which will result in an order of magnitude increase in the number of conjunctions. Within 5-10 years, performing conjunction avoidance maneuver planning manually will be a huge operational burden. One solution to this challenge is to automate the process of satellite swarm conjunction avoidance. This paper describes Coordinated Autonomous Debris Avoidance (CADANCE), an AI-based solution intended to run on each SmallSat in a swarm and collaboratively plan conjunction avoidance maneuvers.

CADANCE consists of two primary modules that achieve the automated conjunction avoidance capability. The first is the **High-Level Planner** module which processes CDMs while considering the satellite's capabilities, the swarm configuration, and the mission to develop one or more candidate high-level plans to avoid the conjunction. This module produces a constrained trajectory optimization problem for each candidate high-level plan which is then processed by the **Trajectory Optimizer** module to produce a series of actions to execute each plan. Finally, each plan is analyzed to calculate a quality score based on fuel use, amount of reduction of the P_c, and adherence to mission constraints. The plan with the highest quality score is the active plan, unless another CDM with updated information arrives. At the maneuver commitment point, if the P_c is still above the allowable threshold set by the O/O, the active plan will be executed, resulting in a series of maneuvers for one or more satellites in the swarm that will avoid conjunctions and allow the swarm to respect mission constraints.

As an illustrative example of how these modules work together, consider two satellites flying in a trailing formation that need to maintain a specified distance to achieve their mission. CADANCE receives a CDM that affects the leading satellite. The **High-Level Planner** module will create a plan to adjust the leading satellite's orbit to avoid the conjunction and then rejoin the formation after the conjunction risk has decreased to below a threshold (1E-6 could be a threshold for a high value satellite). This plan is called a phased orbit restoring plan, and will consist of an orbit raising maneuver, followed by an orbit lowering maneuver into a catch-up orbit and finally an orbit raising maneuver to return to the initial orbit. This three-step plan requires three separate constrained optimization problems, which the **High-Level Planner** will create and provide to the **Trajectory Optimizer**. In this example, the **High-Level Planner** only produced one plan, so it will by default be the active plan and will be used unless a new CDM becomes available.

There are two main benefits of using CADANCE as opposed to a manual process. The first is the reduction in human labor required, which enables managing larger swarms in orbital regimes with more obstacles. The second is that our trajectory optimization approach optimizes for plans with low fuel use, which results in longer satellite lifespans.

The primary contribution of this paper is to explain how constrained optimization can be used as a common representation by both the **High-Level Planner** module and the **Trajectory Optimizer** module. This is the key aspect of the work, because it is what enables an end-to-end solution that can start with CDMs and end with thrust commands for every satellite in the swarm. Constrained optimization is also beneficial because it enables CADANCE to place safety guardrails on the satellites.

The Related Work section briefly describes techniques already used to plan conjunction avoidance maneuvers and existing technologies that CADANCE builds on top of. Following that, the Methods and Results and Discussion section will cover the CADANCE software and the initial evaluation process. The Conclusion summarizes progress on CADANCE to date and outlines future work.

2. RELATED WORK

A common manual approach for planning a conjunction avoidance maneuver is to use a maneuver trade space [1]. This approach simplifies the maneuver into only prograde and retrograde maneuvers and maps out the P_c relative to both delta-v and thrust times. With this approach, a satellite O/O can then choose a delta-v and thrust time that minimizes the P_c to below their threshold while not wasting fuel. This approach only handles the conjunction avoidance maneuver itself, and further orbit restoring maneuvers would again need to be planned manually.

There are already approaches for autonomous SmallSat swarm conjunction avoidance being developed in the satellite industry. For example, the SpaceX Starlink satellites have an automated collision avoidance system. However, this software cannot be fully trusted to manage their satellite swarms. Recent examples where SpaceX could have used their autonomous conjunction avoidance system resulted in them requesting the other satellite in the conjunction make a maneuver instead [2]. Additionally, once developed, SpaceX will likely not license their software for other satellite O/Os, making it impossible for universities or small science missions to use.

CADANCE leverages several existing technologies. First, CADANCE uses Case-Based Reasoning (CBR) to implement its **High-Level Planner** module. CBR matches a current problem to a library of past problems with accompanying solutions. CBR essentially divides the decision space into different possible cases, each with a solution or solution-generating method, then analyzes each case to choose the most similar one. CBR uses a scoring function which computes a weighted sum of the differences between the features of a current case and the features of each case in the case base. CBR then adapts the solution for the most similar case to current variables like the number of satellites, their orbits, and required orbit adjustment time [3].

CADANCE uses trajectory optimization to plan the series of thrust maneuvers and attitude adjustments required to execute the high-level plan developed by the **High-Level Planner**. Trajectory optimization is a subfield of optimal control. Broadly, it is the process of finding a trajectory—with a trajectory being defined as a set of states and control inputs—that minimizes or maximizes an objective function for a dynamical system over time. There are numerous approaches for satellite trajectory optimization, such as collocation, direct shooting, and linear/non-linear programming [4]. Metaheuristic approaches, which are a kind of approximate optimization algorithm, have been increasingly popular for their computational efficiency and ability to discover high quality solutions that are approximations of the global optima. These methods include evolutionary algorithms such as Genetic Algorithms, Differential Evolution, and Particle Swarm Optimization (PSO) [5]. Metaheuristic algorithms can often be used to provide an initial solution to trajectory optimization methods that are particularly sensitive to the initial solution [5].

CADANCE uses PSO because of its potential to scale with computational resources; ability to handle nonlinear constraints and costs; ability to handle non-differentiable cost; and ease of integration with the **High-Level Planner**. Past work already used PSO to plan conjunction avoidance maneuvers [6]. However, this past work does not provide the swarm level orchestration that CADANCE does. CADANCE can maneuver one or more of the satellites in the swarm to avoid a conjunction and reconfigure the swarm to still complete mission objectives. Also, CADANCE is more computationally efficient than this past work.

3. Methods

This section first summarizes the two main components of the CADANCE software and then describes the evaluation process that proves its feasibility. The two components are the **High-Level Planner** and the **Trajectory Optimizer**. Note that the prototype version of CADANCE is a centralized algorithm, while the in-space version will be distributed across the swarm.

High-Level Planner

The **High-Level Planner** uses an implementation of CBR that considers the following features: P_c threshold, upcoming station keeping maneuvers, support from other satellites, ability to fulfill mission requirements while thrusting, breaks in the mission requirements, and mission constraints. The mission constraints that CADANCE can handle are attitude constraints, temporal constraints, and relative satellite motion constraints. An attitude constraint specifies a range of allowable pitch, roll and yaw values for the satellite while performing and/or after the maneuver. For example, an attitude constraint could be used to keep a phased array antenna pointing towards the Earth so the satellite can keep communicating while it is maneuvering. Temporal constraints prohibit the satellite from performing a maneuver in predefined time windows. For example, the mission might require that the satellites only perform maneuvers when they are over the Earth's poles because those are times when maneuvering will not interfere with mission tasks. Relative satellite motion constraints are useful for satellites in a tight formation, where the satellites need to raise their orbit in a synchronized manner.

The case in the case-base with the set of features most similar to the current scenario is the matching case, and other highly similar cases will be stored as potential secondary options. Each case in the case-base has an associated solution template which is a high-level plan for the correct maneuvers for the case. It is called a solution template because it is a general plan that requires details specific to the current scenario to be added.

The **High-Level Planner** uses CBR to match the current scenario to one of the following solutions:

1. No maneuver: Empty Plan, no actions
2. Orbit raising: Single orbit raising maneuver
3. Orbit lowering: Single orbit-lowering maneuver
4. Orbit raise and restore the original orbit: Orbit raising maneuver followed by an orbit lowering maneuver of the same magnitude
5. Orbit raise and phase restore the original orbit: Orbit raise, then orbit lower into a catch-up orbit; finally, orbit raise to return to the initial orbit
6. Orbit raise with backup satellite: Orbit raise maneuver for the satellite with the conjunction and an orbit plane adjustment maneuver for a backup satellite

The **High-Level Planner** will fill in the selected solution template with additional data to limit the search space of the **Trajectory Optimizer**. The key approach for this is to use approximate formulas to calculate how much the satellite's orbit needs to be raised or lowered to avoid a conjunction. This process will use the covariance volumes of the satellite and the space debris to compute a magnitude of in-track miss distance that would reduce the P_c to below the threshold. This desired in-track miss distance is converted to a required delay time using the velocity of the satellite. Then, this delay time is added to the satellite's original period to find the desired final orbital period T_f after the maneuver. To solve for the amount of orbit raising to achieve a required final orbital period T_f, our software will find the required final semi-major axis a_f. Formula 1 shows Kepler's third law rewritten to show how to convert from desired final orbital period T_f to final desired semi-major axis a_f.

$$a_f = \sqrt[3]{\frac{GMT_f^2}{4\pi^2}} \tag{1}$$

G is the gravitational constant and M is the mass of the Earth. The required amount to raise the orbit is the difference between the final orbit a_f and the initial semi-major axis.

The **High-Level Planner** will generate one or more constrained optimization problems, depending on the number of maneuvers in the high-level plan, and send them to the **Trajectory Optimizer**. Each of these constrained optimization problems has fuel use as primary cost, and then a set of constraints on the final orbit and on the transient state of the satellite while it is maneuvering. Consider an orbit raising maneuver as an example of how to design a constraint for a certain type of maneuver. An orbit raising maneuver has a range of allowable altitudes higher than the starting altitude as one constraint on the final orbit. Refer to Figure 1 for a diagram of how the altitude constraint works. There is a target vector that specifies the point in space that the new orbit is supposed to intersect with. This point is chosen to be a safe distance above the object causing the conjunction. There is a spherical region at the end of the target vector, with radius equal to the allowable distance between the desired orbit raising vector and the actual result. In our implementation, the radius of the spherical region was 5 meters, so the final orbit always passes through a point at most 5 meters away from the target vector.

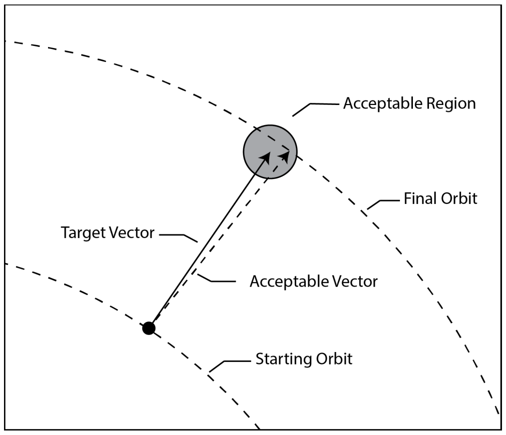

Figure 1. Diagram of the orbit raising constraint

The final orbit is also constrained to have at most a small change in eccentricity, inclination, and longitude of the ascending node. An orbit lowering maneuver has a similar set of constraints, except the final altitude has an allowable range of altitudes lower than the original orbit. A plane change maneuver is again similar, but where the target vector is perpendicular to the original orbital plane. Another constraint type is an ephemeris matching constraint, which is described in more detail in the description of the **Trajectory Optimizer**. The ephemeris-matching function is useful for returning a satellite to its original position in a formation and for enforcing a relative motion constraint between two satellites. The **High-Level Planner** can also produce multiple maneuver plans such as an orbit restoring plan and a phased orbit restoring plan. All of the maneuvers in a multiple maneuver plan have the same set of constraints as an orbit lowering and orbit raising maneuver on its own.

In addition to the constrained optimization problem, the **High-Level Planner** also creates a schedule for when to perform each maneuver. This schedule is built using a heuristic guideline. Specifically, the maneuver will be performed when the satellite is on the opposite side of the Earth from the projected location where the satellite and conjunction object will be closest. This is the location where the smallest delta-v will produce the largest increase in the distance between the two objects at the time of closest approach (TCA). The **High-Level Planner** will shift the maneuver time earlier by one or more period if needed to avoid time windows imposed by any temporal constraints.

Trajectory Optimizer

The goal of the CADANCE trajectory optimization algorithm is to discover a trajectory for each satellite which reduces the P_c with a secondary object while minimizing fuel use and following mission constraints.

CADANCE uses PSO for trajectory optimization, which stochastically generates candidate solutions within the search space and then intelligently updates the search space position of every particle in the swarm each epoch (iteration). In our case, the search space is fully parameterized by thrust duration, thrust azimuth, and thrust elevation. Here swarm refers to the swarm of particles used by PSO, not the swarm of SmallSats. Each particle uses an update function that is a weighted combination of its own best-known position and the best-known position of the entire swarm. The former's weighting is known as the "cognitive coefficient" (c_c), and the latter, as the "social coefficient" (c_s). This update function creates an effect that mimics the swarming behavior of a flock of birds, where the swarm flocks together toward the optimal solution. As a heuristic, PSO does not guarantee that the optimal solution is discovered; however, it excels at searching a large search-space for a solution that is near-optimal.

Below is pseudo-code that outlines the general approach of PSO as implemented in CADANCE [7]:

Algorithm 1 Particle Swarm Optimization

```
1: procedure PSO($c_c, c_s, w, N$)
2:     $f(g) \leftarrow +\infty$                          ▷ Set cost for best known position to infinity
3:     for each particle $i = i_1, i_2, ..., i_N$ do
4:         $x_i \sim U(b_{lo}, b_{up})$                  ▷ Initialize particle position uniformly randomly
5:         $p_i \leftarrow x_i$                           ▷ Initialize best known to its current position
6:         if $f(p_i) < f(g)$ then
7:             $g \leftarrow p_i$                         ▷ Update swarm's best known position
8:         end if
9:         $v_i \sim U(-|b_{up} - b_{lo}|, |b_{up} - b_{lo}|)$   ▷ Initialize the particle's velocity
10:    end for
11:    while termination criterion is not met do
12:        for each particle $i = i_1, i_2, ..., i_N$ do
13:            for each dimension $d = 1, ..., D$ do
14:                $r_c, r_s \sim U(0, 1)$                ▷ Pick random numbers
15:                $v_{i,d} \sim wv_{i,d} + c_c r_c(p_{i,d} - x_{i,d}) + c_s r_s(g_d - x_{i,d})$
16:            end for
17:            $x_i \sim x_i + v_i$
18:            if $f(x_i) < f(p_i)$ then
19:                $p_i \sim x_i$                         ▷ Update the particle's position
20:                if $f(p_i) < f(g)$ then
21:                    $g \sim p_i$                       ▷ Update the swarm's best know positions
22:                end if
23:            end if
24:        end for
25:    end while
26: end procedure
```

Note the velocity and position update functions. These comprise the weighted update that determines the particle's next values in the search process. Consider the velocity update function, Formula 2.

$$v_{i,d} \sim \omega v_{i,d} + c_c r_c(p_{i,d} - x_{i,d}) + c_s r_s(g_d - x_{i,d}) \quad (2)$$

Note the variable ω, which is a predefined constant for inertia that specifies the weight placed on the previous velocity value. Also note the variables $r_1$and r_2. These are simply uniform random variables between 0 and 1 that help facilitate the randomness in the stochastic search process [6]. The value $p_{i,d}$ is the best solution the particle has encountered while g_d is the best solution the entire swarm has encountered. The values b_{lo} and b_{up} are the lower and upper boundaries of the search-space respectively.

To simplify the optimization problem as much as possible, we assume that the spacecraft operates at maximum thrust, and that the **High-Level Planner** plans the start time for each maneuver. PSO uses the latest values for duration, azimuth, and elevation for each particle to define a thrust maneuver which is dynamically propagated forward toward the TCA using orbital propagation. The resulting trajectory is evaluated using the objective function, and the epoch is complete once every particle has performed this process. Each particle then updates its position using the PSO update function, repeating until a desired number of epochs is reached.

For impulsive maneuvers, pure analytic approximations exist which are sufficient to propagate the spacecraft forward in time. Pure analytical propagation approximates the equations of motion with closed-form expressions, instead of numerically integrating the forces, which is much faster, but *can* be less accurate over extended periods of time [8]. However, in our case—that is, low-thrust continuous maneuvers—the spacecraft requires numerical, or semi-analytic, propagation. This is because the multiple forces need to be modeled over an extended period of time, and there are no good analytical models for this. Therefore, CADANCDE uses the Orekit implementation of the 8th order Dormand-Prince numerical integrator for propagation, which uses the Runge-Kutta iterative method for integration. The Runge-Kutta method is widely used in space applications today [9]. The Runge-Kutta method offers adaptive step-size integration, which resulted in shorter maneuvers requiring longer to integrate because they required a smaller step size.

The most important design consideration for trajectory optimization is the objective function which needs to be designed such that a desired trajectory is the global minima. The CADANCE trajectory optimizer uses an objective function that follows the Bolza form, which is shown in Formula 3. The Bolza form accepts the control state function x (vector describing both state and control) [10].

$$J(x) = l\left(x(t_i), x(t_f)\right) + \int_{t_0}^{t_f} L\left(t, x(t), \dot{x}(t)\right)dt \qquad (3)$$

Note the two terms: the first is a boundary term at t_f, while the second is a continuous term across the entire time span t_0 to t_f. l represents a cost associated with the final state and L represents a cost throughout the entire trajectory. The first term is often called the Mayer term, and the latter term is referred to the Lagrange term (which is not to be confused with Lagrange Multipliers, which will be discussed later).

CADANCE contains implementations of two primary objective functions: **target vector** and **ephemeris matching** which correspond to different constraints created by the **High-Level Planner**.

The **target vector** function corresponds to a constraint created by the **High-Level Planner** to ensure that the final orbit passes through a given position in space. The **target vector** function has a global minimum at a spacecraft trajectory that passes through a specified range of positions near a target position vector defined in the Earth-centered Inertial (ECI) frame. This position vector is provided to the function as a relative position vector defined in the spacecraft UVW frame (a coordinate frame relative to the spacecraft) and then converted to the ECI frame. For instance, a target vector representing an orbit raising maneuver of 1 kilometer can simply be defined in the spacecraft frame as [0, 0, 1000]. This way of defining desired orbits makes it simpler for the **High-Level Planner** to generate human-understandable inputs to PSO as well as for the **High-Level Planner** to allow for a range of possible satisfying orbits that can be narrowed down with other constraints. The objective function is structured primarily as a penalty that correlates with the minimum distance between the spacecraft's new orbit and the target point. Since this is a cost that needs to be evaluated across the trajectory, this can be considered as a part of the Lagrange term of the objective function.

The **ephemeris-matching** function has a global minimum for a trajectory that matches an ephemeris provided by the **High-Level Planner**. This function is implemented by *sampling points at regular time intervals* in the ephemeris and finding the maximum separation between a particle's position and the ephemeris over those sampled times. Sampling points across the trajectory is a discretized estimate of the Lagrange term in the objective function, chosen for its simplicity and speed of evaluation. The global minimum thus corresponds to a trajectory where this separation is minimized.

Both classes of objective function also ensure their global minimum lies where fuel-use is minimized. This is achieved with a fuel penalty on top of the objective function. In practice, this penalty leads the optimizer toward the theoretically most efficient maneuver. On top of finding the most energy-efficient maneuver, the fuel penalty reduces the number of local minima in the objective function. This is because for two solutions with otherwise equivalent objective costs, the fuel penalty penalizes the solution that uses more fuel. In reducing the number of local minima, the optimizer outputs more predictable results.

PSO does not natively handle constraints. As such the trajectory optimizer uses a Lagrangian Relaxation for constraints on the final orbital elements. Lagrangian Relaxation is a common technique in optimization that converts a constrained optimization problem into an unconstrained optimization problem by adding costs for constraint violations. This new unconstrained optimization problem will have a global minimum that is close to the global minimum of the original constrained optimization problem. CADANCE matches Lagona et al.'s form of Lagrangian Relaxation [6]. In their example, their constraints on the orbital elements takes the form in Formula 4.

$$J = J_* + \omega_L\left(\frac{p(t_f) - p_f}{p_i}\right) + \omega_L\left(\frac{g(t_f) - g_f}{g_i}\right) \qquad (4)$$

J_* is the unconstrained objective function, $\omega_L$is a weighting on the constraints, and p and g represent a function of orbital elements over time for the evaluated maneuver. p_f, g_f and p_i, g_i represent the final and initial values for those orbital elements, assuming no maneuver is performed. Note how each Lagrange Multiplier is normalized by its initial value. This is critical for the optimization to work on our objective functions, since each orbital element can vary by orders of magnitude.

These Lagrange Multipliers are primarily used for orbital constraints; they make sure the satellite stays within operating ranges after the maneuver is performed. Since these constraints are only evaluated after the maneuver is completed, for our purposes, these constraints can be considered a part of the Mayer term of the objective function. In Formula 4, each constraint is weighted with ω_L but each term can also be assigned a different weight if some constraints are more important than others.

For simple, search-space limiting constraints, such as hard boundaries on thrust duration and angle, a simple solution is to use the largest penalty value possible in the programming language when the constraint is violated. While this solution does not guarantee continuity on the objective function, PSO does not require continuity, and in practice this is not an issue.

Evaluation Process

An important part of our evaluation methodology was to avoid bias by testing the CADANCE approach with simulation software that was not used inside of CADANCE itself. To this end, we used Orekit in our software for the trajectory optimization step and we used GMAT to evaluate the results. We tested CADANCE in the following eight diverse scenarios to demonstrate its capability to handle different satellite swarm types and constraints.

1. No Maneuver: Satellite in LEO with no constraints, which has a conjunction but with a P_c below the threshold. Satellite will not perform any maneuvers.
2. Orbit Raising: Satellite in LEO with no constraints, which has a possible conjunction with P_c above the threshold. Satellite will perform an orbit raising maneuver, where the amount to raise the orbit will depend on the uncertainty of the location and velocity of the secondary object.
3. Orbit Lowering: Satellite in LEO with a flight ceiling constraint, and a possible conjunction with P_c above the threshold. Satellite will perform an orbit lowering maneuver, where the amount to lower the orbit will depend on the uncertainty of the location and velocity of the secondary object.
4. Landsat with Orbit Coordination: The two active Landsat satellites, Landsat 8 and Landsat 9, are in the same orbit but on opposing sides of the Earth. The Landsat satellites are constrained to a fixed orbit. The satellite with a possible conjunction with P_c above the threshold will perform an orbit raising maneuver followed by an orbit lowering maneuver to return to its original orbit.
5. Trailing Satellites: Two satellites flying in a formation where one satellite is supposed to follow in the same orbit but with a separation distance from the leading satellite. The satellites are constrained to a set orbit and range of allowable separation distances. The satellite with the possible conjunction with a P_c above the threshold will perform an orbit raising maneuver and then a multi-step maneuver to return to the formation.
6. Massive Internet Provider Swarm with No Constraints: Swarm with many satellites, including a satellite with a possible conjunction with P_c higher than the threshold and a satellite in a nearby orbit that can provide backup coverage of an important region. The satellite with the possible conjunction will perform an orbit raising maneuver, which will result in an important region not having coverage during a time window. The backup satellite will then adjust its orbit slightly to provide coverage.
7. Massive Internet Provider Swarm with Attitude Constraint: Like scenario (6) with the addition of an attitude constraint restricting the allowable pitch of all satellites in the swarm. The satellite with the possible conjunction with P_c higher than the threshold will again perform an orbit raising maneuver, but now, the trajectory optimizer will have an added constraint forcing the pitch to stay in the allowable range. The satellite providing backup coverage will have the same attitude constraint.
8. Massive Internet Provider Swarm with Temporal Constraint: Like scenario (6) but with the addition of a time window constraint. The satellite with the possible conjunction will therefore need to wait until after this window to perform the thrust maneuver. CADANCE will still try to schedule the maneuver for a time when the satellite is on the opposite side of the Earth from where it will be at the TCA but may have to choose another time depending on the time window.

For each scenario, we ran CADANCE on real CDM data, which contained positions, velocities and covariance volumes for both the primary satellite and the secondary object causing the possible conjunction. For each scenario, our integrated pipeline first ran the **High-Level Planner** to produce a series of constrained trajectory optimization problems. Then the **Trajectory Optimizer** produced a series of thrust maneuvers based on the constrained trajectory optimization problems. The **Trajectory Optimizer** produces a satellite trajectory and an estimated fuel use. To verify the quality of the result, we simulated the new trajectory for the satellite and the trajectory for the secondary object in GMAT. We used the GMAT simulation to estimate the new distance at TCA and new P_c and verified that the new P_c was below the threshold.

Figure 2 shows how orbit raising affects a satellite's orbit. Before the maneuver, the debris and the satellite are approximately 348 meters apart which produces a P_c of 7.947E-6. After the maneuver, the satellite and debris have a

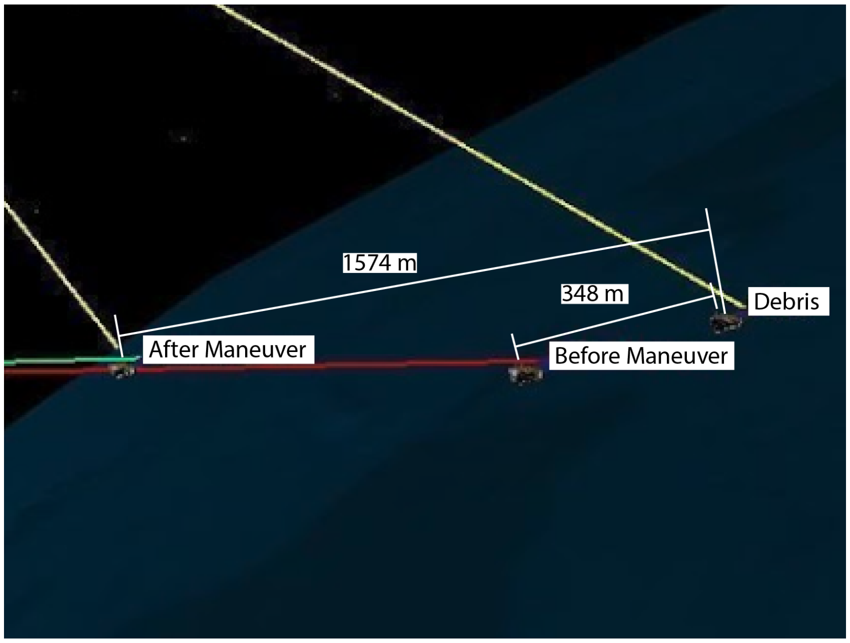

Figure 2. Visualization of the position of the space debris and the satellite after the maneuver and if the maneuver did not happen. Note this image is a screenshot of the 3D GMAT software with some additional overlays to show distances in 3D space. The 2D distances in the image are not to scale with distances in 3D space.

much larger in-track distance between them, which produces a distance of 1574 meters and a P_c of 5.240E-7. Appendix A shows the P_c's and miss distances at TCA if a conjunction avoidance maneuver is performed and if one is not performed for all scenarios.

4. Results and Discussion

This section summarizes the results from our simulation-based testing of CADANCE. First, the **Trajectory Optimizer** in CADANCE should produce the most fuel efficient thrust controls for basic maneuvers. For example, for a simple orbit raising maneuver with no other constraints on the resulting orbit, the **Trajectory Optimizer** finds a thrust vector that is almost in-track and prograde. Specifically, the **Trajectory Optimizer** produced a thrust maneuver where the satellite has pitch and yaw angles equal to -8.457E-4 and 7.451E-4 respectively, which is very close to the expected in-track prograde burn. Likewise, for an orbit-lowering maneuver with no other constraints on the resulting orbit the **Trajectory Optimizer** finds a thrust vector that is very close to in-track and retrograde.

Appendix A provides a summary of P_c's and miss distances at TCA with and without the conjunction avoidance maneuver for all eight scenarios. Each scenario uses a different type of satellite swarm with different mission constraints. Additionally, each scenario uses a different covariance volume, which is why different scenarios need different amount of orbit raising to achieve similar reductions in P_c. In every scenario, CADANCE could plan a satellite maneuver that allowed the satellite to reduce the P_c to below the required threshold of 1E-6. Additionally, the final P_c range is between 5.070E-7 and 9.481E-7, which is less than half an order of magnitude away from the threshold. This is important because it would often be possible to use an excessive amount of delta-v to create an unnecessarily large miss distance at TCA, which would reduce the P_c to nearly zero. However, this would use more fuel than was necessary, which would ultimately reduce the lifespan of the satellite.

Computational efficiency is an important aspect of CADANCE, to ensure that it can run on radiation-tolerant computers in space. The total amount of wall clock time required to run each scenario on an Apple M1 Max processor with a clock speed of 3.2 GHz and 64 GB of memory was less than 142 seconds and some runs took as little as 30 seconds (Appendix B contains the runtimes for all the scenarios). This is a runtime that is more efficient than a similar approach that also uses PSO and takes 30.5 minutes to plan a single orbit raising maneuver [6]. Our approach is more efficient, because our implementation of PSO searches a smaller search space confined by the constraints produced by the **High-Level Planner.**

Additionally, PSO is easily and efficiently parallelizable by design. Clusters of particles can be optimized basically independently, only transferring the best-known set of parameters (social values) between cores. This means that there is very little overhead required for the parallelization and that the runtime could be significantly reduced by fully utilizing multiple processor cores.

Transitioning CADANCE for use on SmallSats in LEO will require running the software on a flight computer. A common computer for SmallSats operating in LEO is the

Versalogic Sabertooth, which is a multi-core rugged embedded computer utilizing commercial off-the-shelf (COTS) components [11]. The Sabertooth can be based on either the 4 core Intel i3 or the 6 core Xeon E processor, which have clock speeds of 1.6 GHz and 2.0 GHz respectively. The individual cores of each processor will be slower than the computer we tested on, but if the algorithm is parallelized, the overall runtime will be less. SmallSat and CubeSat Operations in LEO swarms tend to utilize more COTS components than satellites in higher orbits. This is acceptable because there are generally lower radiation levels in LEO compared to higher orbits [12] and LEO missions tend to be shorter so long duration reliability is not as much of a requirement. This means that LEO missions can take advantage of the lower cost, and higher performance of COTS components [13].

For longer term missions, CADNACE is also efficient enough for radiation-hardened and radiation-tolerant computers. Radiation-hardened computers are specifically designed to withstand intense radiation levels while radiation-tolerant computers can resist radiation but only when it is not as intense or long-lasting. The RAD750 radiation-hardened processor has a single 220 MHz core which is an order of magnitude slower than the Apple M1 Max processor we tested CADANCE on. While the RAD750 is not typically used on low-cost SmallSats in LEO, it serves as a good baseline for radiation-tolerant computer speeds. Even with an order of magnitude slowdown the system will still be able to react fast enough considering that the maneuver commitment point is typically 1 to 0.5 days before the TCA [1]. In addition, multi-core processors are becoming more ubiquitous, and CADANCE is well positioned to take advantage due to easy parallelization of the PSO algorithm. In fact, the newer version of the RAD750, the RAD5545, has 4 cores and has 10 times higher throughput than the RAD750 [14]. Lastly, CADANCE's design can be easily adapted to the specific computational requirements of the mission. The **High-Level Planner** can be adapted to provide tighter or looser search space constraints to reduce or increase reliance on the more computationally taxing trajectory optimizer. Tighter search constraints will increase computational efficiency but may result in the less optimal results.

It is important to also consider how the efficiency of CADANCE will change with different types of maneuvers and different swarm sizes. We found that orbital plane adjustment maneuvers require more epochs of PSO to reach a good solution than simpler orbit raising or lowering maneuvers. Also, optimizing smaller maneuvers is more time consuming because small maneuvers require smaller step sizes for propagation which makes the simulation take longer. Therefore, if CADANCE needs to perform many small maneuvers, or many orbital plane adjustment algorithms, its runtime may increase.

The CADANCE runtime scales linearly with both the number of satellites in the swarm and with the number of maneuvers required by the conjunction avoidance plan. This is because orbital propagation dominates the runtime, and one orbital propagation is required for each maneuver of each satellite. However, these runtimes are not negatively affected by relative motion constraints between satellites. This is because CADANCE will optimize one satellite first and then optimize the trajectory for the other satellite using the single satellite constraints (e.g. on attitude) and the produced ephemeris of the other satellite. In other words, CADANCE does not perform a joint optimization which would have twice the number of parameters and therefore would be much slower.

5. CONCLUSION

CADANCE was able to reduce P_c to below the required threshold for a diverse set of scenarios with different mission objectives, covariance volumes, orbital regimes, and types of mission constraints. In other words, these scenarios were varied enough to give us confidence that the CADANCE concept can be generalized to most SmallSat swarms. In addition to correctness, CADANCE is also very efficient, taking less than 142 seconds to plan multi-maneuver trajectories.

While our prototype demonstrates the correctness and efficiency of CADANCE, there are several items of future work that are required. First, CADANCE needs to be tested against real conjunction avoidance scenarios and the conjunction avoidance maneuver that CADANCE produces needs to be compared against the actual maneuver that was used in that scenario. If CADANCE can produce a more fuel-efficient conjunction avoidance maneuver or a maneuver with less mission impact, it would provide extra evidence of CADANCE being able to produce better solutions in addition to handling conjunctions autonomously.

In addition, future work is required to create an in-space version of CADANCE. The first step is to port the system to NASA's core Flight System (cFS), which is a middleware framework for flight software used on the types of academic and small science mission satellites that CADANCE is targeted towards. This would require porting the software from Java to C++ and integrating it as a cFS application. Porting CADANCE to C++ would also allow for tightly optimized code that can better utilize the limited cache sizes and memory for radiation-tolerant and radiation-hardened computers. In addition, CADANCE needs to be tuned with a higher-fidelity orbital propagation model. The current prototype uses Orekit's default orbital models while flight software typically uses custom designed models. CADANCE will also need to be configured for the real satellite swarm with information about the mission constraints and satellite capabilities. Finally, in future work, the centralized controller that the current version of CADANCE uses will be replaced with a decentralized controller based on Consensus-Based Bundle Algorithm (CBBA). CBBA is a decentralized market-based protocol that iterates between a bundle building phase where each SmallSat will greedily generate a plan meeting its own mission and conjunction avoidance requirements and a consensus phase where conflicting plans are identified and resolved with SmallSats nearest to each other [15].

APPENDICES

A. P_C WITH AND WITHOUT MANEUVER

Table 1 shows the distance at TCA and P_c if the conjunction avoidance maneuver is performed and if no maneuver is performed. The key feature to note in this data is that the P_c starts at above the threshold of 1E-6 for all scenarios except the first (No Maneuver) scenario. Then, the maneuver always creates a larger distance at TCA then would have occurred with no maneuver, resulting in a P_c below the threshold. However, the P_c is always only slightly below the threshold to avoid wasting fuel.

Table 1. Distance at TCA and P_C values

Scen-ario	No Maneuver		Maneuver	
	Distance (m)	P_c	Distance(m)	P_c
1	398.529	1.464E-11	N/A	N/A
2	348.713	7.947E-6	1574.000	5.240E-7
3	87.420	9.414E-4	5127.089	6.315E-7
4	249.967	7.607E-5	2967.424	5.070E-7
5	350.520	9.609E-6	1996.000	5.321E-7
6	279.362	2.620E-5	12011.452	6.235E-7
7	314.038	3.142E-5	7912.969	9.481E-7
8	210.012	9.663E-5	9690.403	8.573E-7

Scenarios: 1=No Maneuver; 2=Orbit Raising; 3=Orbit Lowering; 4=Landsat with Orbit Coordination; 5=Trailing Satellites; 6= Massive Internet Provider Swarm with No Constraints; 7= Massive Internet Provider Swarm with Attitude Constraint; 8= Massive Internet Provider Swarm with Temporal Constraint

B. RUNTIMES FOR SCENARIOS

Table 2 shows the runtimes required to complete the entire high-level planning and trajectory optimization process, starting from the CDM and ending with thrust and attitude controls. These runtimes were recorded on an Apple M1 Max processor with 64 GB of memory. There is no runtime recorded for the "No Maneuver" scenario as it does not actually do any optimization, which dominates the runtime. Note that these runtimes reflect CADANCE operating with single-threaded trajectory optimization, and the runtimes should scale down with multiple threads distributed across over multiple processor cores.

Table 2. Runtimes for each scenario

Scenario	Runtime (s)
1	N/A
2	39.265624
3	116.021613
4	51.123566799
5	86.410465695
6	141.489585273
7	58.386330
8	19.366113243

Scenarios: 1=No Maneuver; 2=Orbit Raising; 3=Orbit Lowering; 4=Landsat with Orbit Coordination; 5=Trailing Satellites; 6= Massive Internet Provider Swarm with No Constraints; 7= Massive Internet Provider Swarm with Attitude Constraint; 8= Massive Internet Provider Swarm with Temporal Constraint

Note that the attitude and temporal constraints in the "Massive Internet Provider Swarm with Attitude Constraints" and "Massive Internet Provider Swarm with Temporal Constraints" scenarios decrease the runtime when compared to the unconstrained "Massive Internet Provider Swarm with No Constraints" scenario. This is because the constraints limit the search space for the trajectory optimizer, enabling it to find a solution faster.

ACKNOWLEDGEMENTS

This material is based upon work supported by NASA under Contract No. 80NSSC23PB595. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official view of policies, either expressed or implied, of NASA.

References

[1] "Step 3 Close Approach Risk Mitigation - NASA." Accessed: Sep. 18, 2024. [Online]. Available: https://www.nasa.gov/cara/step-3-close-approach-risk-mitigation/

[2] P. M. Sutter and U. Today, "Starlink and OneWeb have their first avoidance maneuver with each other's constellations." Accessed: Sep. 24, 2024. [Online]. Available: https://phys.org/news/2021-05-starlink-oneweb-maneuver-constellations.html

[3] A. Aamodt and E. Plaza, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches," *AI Commun.*, vol. 7, no. 1, pp. 39–59, 1994, doi: 10.3233/AIC-1994-7104.

[4] M. Kelly, "Transcription Methods for Trajectory Optimization: a beginners tutorial," *ArXiv Optim. Control*, Jul. 2017, Accessed: Sep. 18, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Transcription-Methods-for-Trajectory-Optimization%3A-Kelly/0e36e16a09ede112a99370e8d669eb066a631bdc

[5] A. Shirazi, J. Ceberio, and J. A. Lozano, "Spacecraft trajectory optimization: A review of models, objectives, approaches and solutions," *Prog. Aerosp. Sci.*, vol. 102, pp. 76–98, Oct. 2018, doi: 10.1016/j.paerosci.2018.07.007.

[6] E. Lagona, S. Hilton, A. Afful, A. Gardi, and R. Sabatini, "Autonomous Trajectory Optimisation for Intelligent Satellite Systems and Space Traffic Management," *Acta Astronaut.*, vol. 194, pp. 185–201, May 2022, doi: 10.1016/j.actaastro.2022.01.027.

[7] M. Clerc, "Standard Particle Swarm Optimisation".

[8] D. Vallado, *Fundamentals of Astrodynamics and Applications*, 4th Edition. Microcosm Press, 2013.

[9] J. Aristoff and A. Poore, "Implicit Runge-Kutta Methods for Orbit Propagation," Aug. 2012. doi: 10.2514/6.2012-4880.

[10] P. D. Loewen and R. T. Rockafellar, "New Necessary Conditions for the Generalized Problem of Bolza," *SIAM J. Control Optim.*, vol. 34, no. 5, pp. 1496–1511, Sep. 1996, doi: 10.1137/S0363012994275932.

[11] "Sabertooth - PCIe/104 Embedded Computer," VersaLogic. Accessed: Oct. 03, 2024. [Online]. Available: https://www.versalogic.com/product/sabertooth/

[12] M. M. Rahman, D. Shankar, and S. Santra, "Analysis of Radiation Environment and its Effect on Spacecraft in Different Orbits," Sep. 2017.

[13] "8.0 Small Spacecraft Avionics - NASA." Accessed: Oct. 04, 2024. [Online]. Available: https://www.nasa.gov/smallsat-institute/sst-soa/small-spacecraft-avionics/#8.3.2

[14] "Space Product Literature | BAE Systems." Accessed: Oct. 03, 2024. [Online]. Available: https://www.baesystems.com/en-us/our-company/inc-businesses/electronic-systems/product-sites/space-products-and-processing/radiation-hardened-electronics

[15] "Consensus-Based Bundle Algorithm (CBBA) - Aerospace Controls Laboratory." Accessed: Sep. 25, 2024. [Online]. Available: https://acl.mit.edu/projects/consensus-based-bundle-algorithm

Biography

Evan Finnigan *is an artificial intelligence software engineer at Stottler Henke Associates, Inc. He has built case-based reasoning, fault management, training, scheduling and planning software for a wide variety of complex real-world problems. Before working at Stottler Henke, Evan was a student researcher at UC Berkeley designing and building assistive robotics.*

Brandon Liu *is an engineer at Stottler Henke Associates, Inc. He has built and designed algorithms for a variety of space and terrestrial applications, ranging from networking optimization to trajectory optimization.*

Richard Stottler *co-founded Stottler Henke in 1988 as a software company dedicated to providing practical solutions to difficult problems by skillfully drawing upon a large repertoire of artificial intelligence technologies. Under his leadership, Stottler Henke has grown steadily and profitably into a 40-person research and software development company with distinctive expertise in intelligent tutoring systems, intelligent simulation, automated planning and scheduling, and intelligent knowledge management. Dick provides technical leadership in the design and development of intelligent tutoring systems, intelligent planning and scheduling systems, and automated design systems. He combines a strong applied research record in artificial intelligence with practical experience in rapid and efficient knowledge engineering. He also led the development of intelligent planning systems for NASA space shuttle missions and aircraft assembly and automated scheduling for the International Space Station. Dick has written or presented dozens of papers and articles for publications such as the proceedings of the International Joint Conference on Artificial Intelligence (IJCAI). He received his BS in engineering from Cornell University and his MS in computer science (artificial intelligence) from Stanford.*