Application of Linear Consensus Protocol to Space Debris Deorbiting

Omar Elhayes
Aerospace Engineering Department
University of Illinois Urbana-Champaign
Urbana, IL
oelhay2@illinois.edu

Pedro G. Leite
Aerospace Engineering Department
University of Illinois Urbana-Champaign
Urbana, IL
pgleite2@illinois.edu

***Abstract*—The escalating accumulation of space debris in Medium and High Earth Orbits (MEO and HEO) poses a significant threat to space missions and hinders deep space observations. This paper proposes a novel approach utilizing Linear Consensus Protocol (LCP) to expedite the deorbiting process of randomly placed debris. In contrast to existing methods tailored for clustered debris, our approach involves agents dynamically performing orbital transfers to efficiently deorbit debris. The objective is for agents to rendezvous with space debris, prioritizing agents with the lowest required ΔV within a given time horizon. To achieve this, we implemented a modified LCP to allow for an agent to rendezvous with a debris in the debris' reference frame, as the debris is moving at constant speed along its trajectory, and demonstrated effectiveness through plots in a 3-dimensional Cartesian plane, as well as potential implementations in spherical coordinates.**

I. Motivation

Space debris has been a problem ever since humanity started launching the first spacecraft into orbit. Most of the man-made debris in orbit today consists of parts of old decommissioned spacecraft or residue from the collision between spacecraft and other pre-existing debris. The accumulation of debris on Earth's orbit has rapidly increased over the years, posing a substantial threat to both unmanned and manned missions, while also obstructing deep space observations from ground observatories. Despite recent efforts by private space companies to address their contributions to the problem, space debris is still an increasing concern, especially at higher Earth orbits where the atmosphere is thin enough to allow debris to stay for years or even decades before naturally decaying and burning up. Not to mention, space debris poses an ethical threat to future generations as it also obstructs stars which could potentially rob future generations from experiencing the night sky as us and those before did.

The motivation for this paper is to explore methods in multi-agent systems to accelerate the process of deorbiting debris from Medium and High Earth orbit, contributing to the sustainability of space activities and the safety of future space missions. In [1], the ant colony optimization method is utilized in order to identify clusters of debris, and guide agents to the areas. Their agents would stay in close proximity to the cluster until they are all deorbited. We aimed to implement a simpler yet effective solution to the problem, one that would focus on optimizing the assignment of each agent to a newly sensed debris rather than optimizing for the rendezvousing trajectories while allowing for randomly placed debris, not relying on finding clusters of orbital debris. Our approach assumes randomly placed debris so that the agents have to constantly perform orbital transfers in order to successfully deorbit the debris.

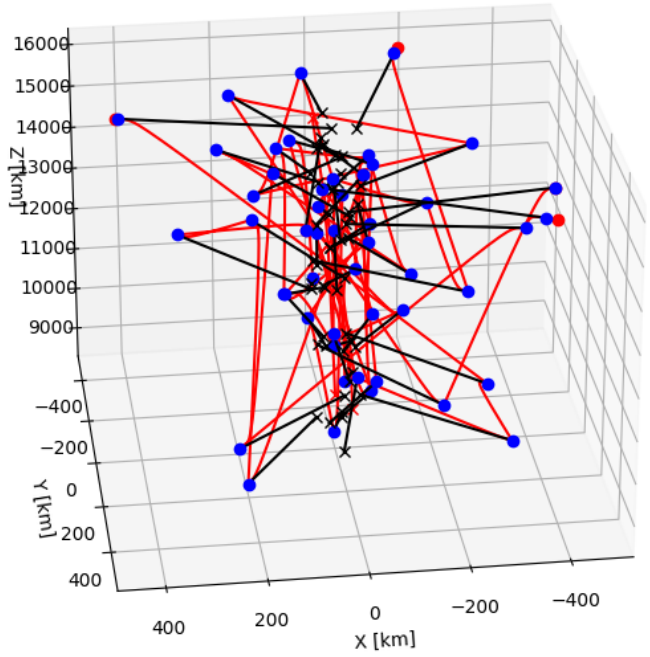

Fig. 1. Simulation of constrained LCP in cartesian coordinates without step velocity constraints.

II. Problem Formulation

The goal of our system is to have agents $a_i \in \mathcal{A}$ rendezvous with all space debris $d_j \in \mathcal{D}$, where $\mathcal{D}$ is the list of debris that have been sensed by agents in the system, and we denote s_i as the best option for a_i to pursue. We define "best option" as the object requiring the smallest ΔV for our agent to rendezvous with it within a set time horizon. We chose this time arbitrarily in order to add a time constraint to the problem. We will define

a relationship between the movements of the agents and ΔV later in the paper.

In order to narrow down the scope of our project, we chose to make some simplifying assumptions. We narrow the range of the initial states of both agents and debris to be only within Medium and high Earth Orbits (MEO and HEO) as debris in low Earth orbit decay naturally and relatively fast due to atmospheric drag. Similarly, we assume that all agents and debris in orbit do not experience atmospheric or solar drag, which simplifies calculations while not significantly impacting our results over such short time scales. For similar reasons, we also assume the Earth is a perfect sphere to avoid gravity gradients. We assume our agents are not limited by fuel and are allowed to perform a limitless amount of orbital maneuvers, and agents can share information with one another anywhere in their orbits through ground station communication. From this, we can also assume that all agents have access to a debris' orbital parameters as soon as it has come within an agent's sensing range once, which can be said because of the deterministic nature of the two-body problem.

Our problem is initialized by setting n agents and m debris at random azimuth θ and zenith ϕ angles, and at a random altitude r between the lower boundary of MEO and the upper boundary of HEO. The ajacency matrix is initially $A= (\mathcal{V}, \mathcal{E}) =0_{n\times m}$, and is then updated to include debris as they come within the sensing range of agents. When a new debris is detected, the agent that detects it will share the information with every other agent by relaying through ground stations. At that point, each agent will calculate the ΔV required to rendezvous with that debris. This information will be shared between all agents, and the agent that calculated the smallest rendezvous ΔV will move on towards it.

$$s_i = \min_{d_j} \|a_i - d_j\|, \forall a_i, d_j \in \mathcal{D}$$

$$\mathcal{A}[a_i, d_j] = 1$$

Once consensus happens between a_i and d_j, the j^{th} entry of the deorbiting array $\mathcal{O}$ will be updated. This will be done with every $d_j \in \mathcal{D}$ until $\mathcal{O} = [1]_m^T$.

A. Adjacency Matrix Algorithm

In order to simply and efficiently optimize the connection assignment between agents and debris, decided to write an algorithm to be ran at every step. The algorithm can be broken up into three parts. The first checks at every step if any debris d_j has come within sensing range of any agent a_i.

Algorithm 1 Detect debris

Input: x
Output: Connections array
  for i in ranger(n) **do**
    for j in range(m) **do**
      if debris deorbited$[j] = 0$ and $A^T[j] = [0]_{1\times n}$ **then**
        distance = $\sqrt{a_i \cdot d_j}$
        if distance $<$ R **then**
          current connections$[j] = 1$
        end if
      end if
    end for
  end for
  return current connections

The second algorithm executes when there is a change in the connections boolean array from Algorithm 1.

Algorithm 2 Creates connection between new debris and optimal agent

Input: x, connections array
Output: A
  A matrix is scanned to see how many agents are available.
  if current connections changed **then**
    if # available agents $<$ # newly sensed debris **then**
      write an array holding indexes of leftover debris such that they're not ignored in the next time step
    end if
    for i in range(n) **do**
      if deorbited debris[i] = 0 and new connections[i] = 1 **then**
        for every available agent **do**
          The orbit of the agent is compared with the orbit of the debris, and the ΔV required for rendezvous is recorded.
        end for
        The smallest ΔV for rendezvous between agents and debris determines which agent will be assigned to which debris, and the A matrix is updated.
      end if
    end for
  end if

Lastly, at every time step, Algorithm 3 is ran. It checks to see if debris d_j is within deorbiting range of its seeking agent a_i, and updates the adjacency matrix to reflect deorbiting.

Algorithm 3 Stops deorbited debris propagation and marks seeking agent a_i as available

Input: x
Output: A
  for i in range(n) **do**
    for j in range(m) **do**
      if $A[i][j] = 1$ **then**
        distance = $\sqrt{a_i \cdot d_j}$
        if distance $<$ deorbiting distance **then**
          $\mathcal{O}[j] = 1$
          $A[i][j] = 0$
        end if
      end if
    end for
  end for

In order to achieve this, we first perfected an analog simulation in a simpler environment, commanding the rendezvous and movements in 3-dimensional Cartesian plane, while holding all previously mentioned constraints. The linear dynamics for our system are as follows:

$$\dot{a}_i = d_j - a_i + \dot{d}_j$$

III. Challenges

Many of the challenges we faced when trying to implement our algorithm were related to trying to have short computation times while at the same properly constraining our system to behave as intended. The first of those challenges comes when the debris is sensed by the agents. When creating connections, multiple agents would be assigned to the same debris, thus causing inefficiencies in our script performance as the completion times increased, and the work done by each agent increased. This was solved by writing a connection boolean array that is checked for every debris to check whether it is already being tracked by an agent. A similar problem appeared with a single agent being assigned to more than one debris. This is even larger of a problem as convergence would never happen, but instead the agent would converge to the point in space between the two debris it was seeking. This was solved by always checking the agent's corresponding row in the A matrix to make sure that it is not assigned.

If at some point there are more debris detected than agents available to deorbit them, the system would naturally overwrite the leftover debris that could not be chased at the current time step. This was a problem as these scenarios would result in some debris never being deorbited, as they were never tracked by agents even after those agents were done deorbiting other debris. We corrected this by implementing a method to separate the remainder debris from the ones to be tracked by the available agents. Those were stored in an array and are only called once when new agents are available to track them.

Another challenge we had to solve which was specific to our Cartesian plots was finding a good method of creating optimal connection assignments. Instead of calculating ΔV for every possible connection and using the smallest one, we decided to propagate the linear paths of the debris to be deorbited those of the available agents, and compare which trajectories would be closest within a specific amount of future steps. We think this is a good analog to comparing required ΔV for rendezvous as it similarly compares how close the trajectories currently are, and, in theory, more similar trajectories would require less effort to rendezvous.

In order to imitate orbital behaviors in a linear field, we also wanted to limit the allowed step velocities of the agents. In order to achieve this, we bounded the possible step sizes for the agents to have a magnitude ranging within the orbital velocities at MEO and HEO. For the sake of accuracy and limiting ΔV, we also constrained the agents' movements such that it would move linearly in the same direction as the last debris it deorbited until a new connection was formed.

We noticed that, for a large number of debris with a small nonzero number of agents, the system would take large amounts of time to terminate, and it scaled poorly with every additional debris. This is mostly caused by the fact that trajectories in our 3D Cartesian simulation are unbounded and our velocities are limited. As time goes, the debris not being tracked would constantly reach farther and farther distances, while our agents would have to essentially "catch up" with those debris. If the debris were in lower altitudes, they would have velocities close to the maximum velocities the agents can command, and the simulation would take very long to complete. This would likely not be a problem with the spherical coordinates scenario, as orbital trajectories are bounded and agents would always be able to reach consensus with agents within a finite amount of time.

The biggest challenge we faced overall was to transfer our approach from Cartesian coordinates to spherical coordinates. We believe there could be a way of applying LCP to this scenario by changing some variables and the way they are updated, due to the way $(\dot{x}, \dot{y}, \dot{z})$ are constant in the Cartesian plane, and $(\dot{\theta}, \dot{\phi}, \dot{r})$ would also be constant in the spherical coordinate system.

IV. Findings

Our findings boil down to the implementation of three dimensional Linear Consensus Protocol in Cartesian coordinates and spherical coordinates. Both of which are highlighted below:

A. Cartesian Coordinates

We successfully implemented LCP in Cartesian coordinates. The program works in such a way that n agents rendezvous and "deorbit"/reach consensus with m debris. Since the adjacency matrix is updated at every time step, it does not matter the number of robots compared to the number of debris as the closest robot will rendezvous, "deorbit" a debris and then move on to the next closest debris moving away at a constant speed until all debris has been "deorbited". This was done in such a way that optimized the trajectories of the agent(s). This means that the agent(s) always reached consensus with

the debris that is the shortest distance away. This can be seen in Figure 2 below:

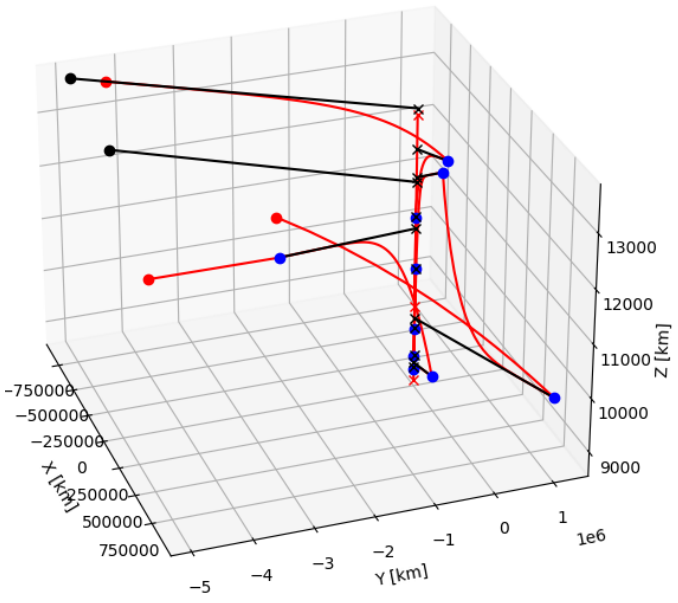

Fig. 2. Consensus of Agents to Debris in Cartesian Coordinates

However, this algorithm does not scale well with the number of debris. This is because, with the increase of the number of debris, the fixed number of agents take longer to rendezvous with all the debris as it is moving away at a constant speed. So the debris travels further as the agents try to rendezvous and "deorbit" all debris. Essentially, the agents struggle to catch up to the debris as it takes longer for to rendezvous with them. This would potentially be fixed in spherical coordinates as orbits are bounded and thus the trajectory of the debris always comes back around and does not keep moving away in a straight line.

B. Spherical Coordinates

We successfully propagated random orbits around the earth in MEO and HEO. This was done by initializing a random position vector with a magnitude between the altitudes of MEO and HEO. Then, the initial velocity vector is calculated based on the position vector using the Vis-Viva Equation (1) below.

$$v = \sqrt{\mu\left(\frac{2}{r} - \frac{1}{a}\right)} \tag{1}$$

In this equation, μ is the gravitational parameter of the Earth, r is the magnitude of the position vector and a is the semi-major axis of the orbit. In circular orbits, the semi-major axis is equal to the magnitude of the position vector.

In our case, the orbits are circular: thus ensuring that the orbit of the space object is circular and within the intended altitudes. This can be seen for six space objects in Figure 3 below:

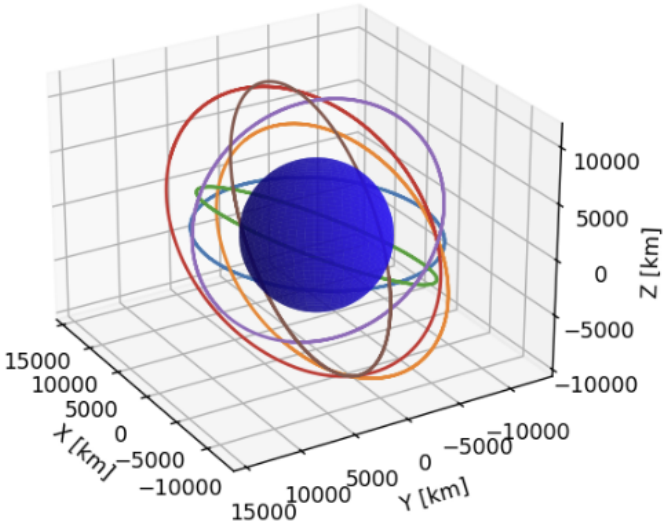

Fig. 3. Propagated Orbits in MEO

With the initialization, it became clear that for an agent to reach consensus with a debris in orbit it must perform a plane change and a phasing maneuver. However, Linear Consensus Protocol cannot be constrained to perform specific orbital maneuvers. LCP is not a path finding algorithm; it can only be made to reach consensus but not to take a specific path. Not to mention, LCP cannot be applied to nonlinear trajectories like those found in orbit. We believe that LCP could be achievable in spherical coordinates for circular orbits due to their constant velocity. Similar to the cartesian coordinates implementation of LCP, the accelerations in spherical coordinates for circular orbits would be zero. Attempts at such an implementation were made and can be seen below in Figures 4 and 5. Overall however, a nonlinear method is necessary for the consensus of agents to nonlinear trajectories.

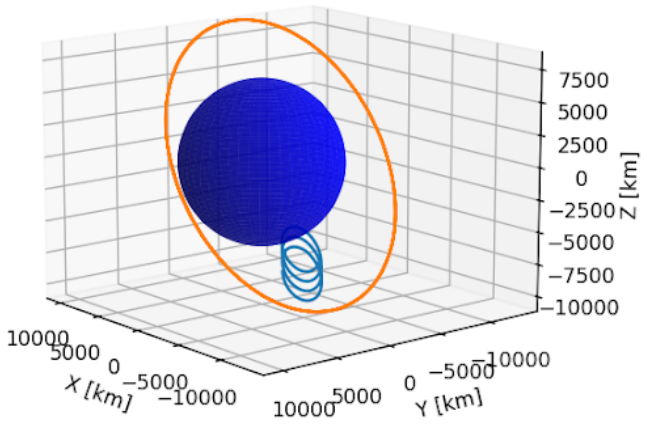

Fig. 4. Propagated Orbits in MEO

Figure 4 depicts the debris orbit in orange and the agent

trajectory in blue. In this case, the agent did not reach consensus and spiraled out of control.

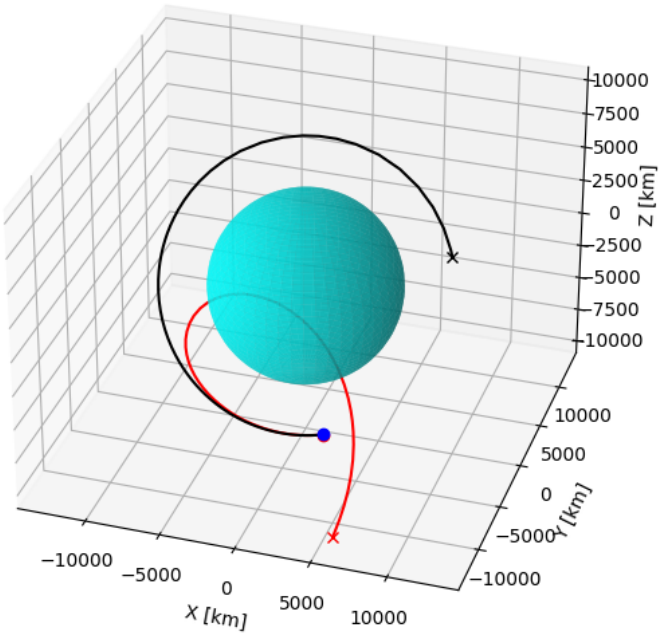

Fig. 5. Propagated Orbits in MEO

Figure 5 depicts the debris orbit in black and the agent trajectory in red. In this case, the agent was able to reach consensus but travelled through the Earth in the process. This would entail that some type of collision avoidance would have to be at play to ensure that the path of the agent is not overlapping with the location of the Earth.

V. Conclusion

Our algorithm works as intended in cartesian coordinates. Although we were able to formulate the problem in orbit, due to the nonlinear trajectories of debris in orbit, Linear Consensus Protocol cannot be applied when dealing with orbital motion. In order to apply our algorithm to the two-body problem scenario, we will need to substitute our LCP implementation for a nonlinear protocol such as Consensus-ADMM.

In a distributed system, Consensus-ADMM involves several nodes, each having its own local objective and data. These nodes collaborate to find a consensus solution to a global problem while exchanging information with their neighboring nodes. The algorithm makes use of iterative steps where each node solves its own local problem while also communicating and updating information with neighboring nodes to reach a consensus/global solution for a global problem. In our case, this would be a perfect fit for agents in orbit around the earth attempting to deorbit space debris in similar orbits to themselves. Each agent can work to deorbit debris while sharing information with neighboring agents to updated each agents data about debris in its vicinity. This would allow agents to share data concerning planned orbital maneuvers, debris characteristics, and debris tracking data. The shared data can be used to optimize maneuvering costs, apply collision avoidance algorithms, and assign debris for deorbiting. Globally, the agents would be working to maximize the amount of debris deorbited while minimizing orbital maneuvering costs, fuel costs, and deorbiting time for debris.

This project could be improved in many ways. In order to achieve our goal of implementing a simple and effective method for clearing our medium and high Earth orbits, we would have to apply a nonlinear consensus method. Future work would involve exploring the implementation of C-ADMM or a similar method that would allow us both to optimize trajectories as well as allow for consensus through orbital maneuvers. Another potential topic to explore in relation to this topic would be the application of a Traveling Salesman Problem to our agent-debris assignment. In a situation where there are large amounts of debris known at any point, we would like to not only assign the optimal connection and optimize the trajectories for rendezvous, but also optimize the entire trajectory to be taken between all deorbiting maneuvers.

References

[1] J. Stuart, K. Howell, and R. Wilson, “Application of Multi-Agent Coordination Methods to the Design of Space Debris Mitigation Tours,” Purdue University, 2016.

TABLE I
Contribution Table

Person	Task
Omar	Propagated orbits of space objects.
Omar	Attempted to implement versions of LCP in spherical coordinates.
Omar	Wrote the ”challenges”, ”findings” and ”conclusion” sections in the report.
Omar	Created the ”spaceObjects” class to store trajectory data and propogate orbits.
Omar	Generated random orbits in MEO.
Pedro	Wrote the ”Abstract”, ”Motivation”, ”Problem Formulation”, ”Challenges”, and part of ”Conclusion”.
Pedro	Wrote script to generate and plot the deorbiting maneuvers in Cartesian coordinates.