Using Genetic Algorithms for Safe Swarm Trajectory Optimization

Rahul Rughani* and David A. Barnhart†
University of Southern California - Information Sciences Institute (ISI) and Space Engineering Research Center (SERC), Marina del Rey, CA, 90292

Satellite swarms are groups of spacecraft that operate cooperatively to perform a common task or goal in orbit. Examples of this are in-space assembly or manufacturing of structures, cooperative sensing platforms, on-orbit servicing, asteroid mining, etc. Swarms would be operating in relative-motion orbits, working around a common point in space, within a few kilometers of each other. Such swarms would share information between each member of the swarm, allowing them to have a shared situational awareness of the entire mission. This paper describes a novel approach to optimize relative-motion orbital trajectories for a spacecraft swarm of arbitrary size and function, using genetic algorithms. The use of genetic algorithms allows convergence to an optimal set or family of trajectories for each member of the swarm regardless of the size of the swarm or mission functions required, should a solution exist. These trajectories will be such that each spacecraft can perform their required individual actions while minimizing the fuel required for maneuvering and also avoiding conjunctions, to a prescribed probability of collision, for a given amount of time. Genetic algorithms have been used previously for optimization of low thrust orbit transfers, drone delivery networks, and control of self-driving cars, among other applications.

I. Nomenclature

N_{pop}	=	population size
N_{gen}	=	number of generations
N_{sat}	=	number of spacecraft
p_{cross}	=	probability of crossover
p_{mut}	=	probability of mutation
δr	=	relative position
δv	=	relative velocity

II. Introduction

With the emergence of the space servicing sector, along with the anticipated return of manned missions beyond low earth orbit, there is a need for quick, efficient, and most of all, safe Rendezvous and Proximity Operations (RPO). More than that, the next technologic step forward in the space domain will be building objects in space using robotic vehicles, which will involve large numbers of spacecraft cooperating in close proximity to each other, all subjected to the laws of orbital mechanics. Currently, there is a lack of knowledge about how to safely operate a swarm of spacecraft in close quarters in a dynamically changing environment (i.e., what is considered a "space construction site"), without creating a high risk of collision and potential debris creation. Methods for swarm RPO safety are being developed but have not yet been tested in space, primarily due to a lack of focused application-based funding on the theoretical side, and a lack of matured technology on the hardware side, given the high risk of operating many spacecraft in close proximity to each other in orbit.

Many people talk about a future in which spacecraft and structures are manufactured and built in space, and indeed there are a handful of companies focusing today on making that dream a reality. However, there is no framework as of yet to be able to setup or control a group, a *swarm*, of individual spacecraft working cooperatively in close proximity to

*PhD Candidate, University of Southern California - Astronautical Engineering Department. AIAA Student Member.
†Research Professor and Director, University of Southern California - Space Engineering Research Center. AIAA Associate Fellow.

each other. This is different from traditional formation flying [1, 2], where a small number of spacecraft are in a static or highly-controlled pre-planned configuration, typically to collect scientific measurements. Swarms, by contrast, are inherently dynamic in nature, shifting in size and scope as the mission progresses. In the case of the on-orbit assembly or construction site example, it is expected that the object under construction would grow in a non-linear fashion over time, and thus each independent construction or servicing spacecraft would be required to constantly modify its movement around the growing central object, continuously avoiding any conjunction.

Whereas there is currently a focus on either the design of a spacecraft that can perform on-orbit assembly [3–5], or large structures to be manufactured in space from a single spacecraft [6–8], there is a distinct lack of research into the processes of in-space construction and how to enable large-scale cooperative actions safely around a central object that is growing over time (i.e. an assembled large scale structure). This paper proposes a new framework and universal tool for spacecraft swarms that enables both building of a set of trajectories for a swarm with distributed tasks and mission goals, and orbit maintenance/control to compensate for gravitational perturbations and unexpected events, including a swarm architecture that is fluctuating in size and shape.

III. Swarms

Satellite swarms are groups of spacecraft that operate cooperatively to perform a common task or goal in orbit. Examples of this are in-space assembly of structures, construction, cooperative sensing platforms, on-orbit servicing, asteroid mining, etc. The swarm framework is designed using relative-motion orbits, working around a common point in space, usually within a few kilometers of each other. Such swarms can share information between each member of the swarm, allowing them to have a shared situational awareness of the entire mission. Although not all members of the swarm may have the same functionality, i.e. some may be communication nodes, others may have robotic arms, and yet others may have scanning cameras and LIDARs, it is expected that they will all have propulsive capabilities to allow them to maneuver in relative orbits around each other and/or a common target.

As swarms are composed of many spacecraft operating in relatively close proximity to each other, there is an inherent risk of possible collision, requiring complex planning to ensure safe operations in orbit. A large part of safety to-date for satellite swarms is passive, where free orbital trajectories will not cause a conjunction within a given amount of time, preventing unnecessary maneuvering and risk. In order to ensure passive safety of orbital trajectories of satellite swarms operating in close proximity, such that one vehicle's failure does not impact any other members of the swarm for a prescribed amount of time, a new set of orbital optimization algorithms are required to obtain these trajectories. This process will consider low collision risk and low delta-v, but also allow each member of the swarm to perform their specific set of tasks assigned to them.

The method of swarm movement optimization that will be investigated in this paper is the use of genetic algorithms to converge on an optimal set or family of trajectories for each member of the swarm. These trajectories will be such that each member can perform their required individual actions, while minimizing the fuel required for maneuvering and also avoiding conjunctions, to a prescribed probability of collision, for a given amount of time. While near term it is expected that ground command uplink intervention would be needed to determine and execute remedial actions in the case of failure on orbit of a single element in the swarm, the long term goal is to develop autonomous algorithms to enable a swarm to accept and remediate failures, in real time. Genetic algorithms have been used previously for optimization of low thrust orbit transfers [9], drone delivery networks [10], and control of self-driving cars [11], among other applications.

The expected result from this analysis will be a methodology to use genetic algorithms for optimization of orbital trajectories for satellites swarms. A genetic algorithm is a very powerful optimization tool, but it requires a great deal of fine tuning to work properly and efficiently. An important result of this analysis will be the identification of a cost function that will govern the optimization of the swarm, while allowing each member of the swarm to complete their specific assigned mission, without endangering the swarm as a whole.

IV. Relative Motion

The equations of motion that are used to model the gravitational forces throughout the optimization process are initially the Clohessy-Wiltshire (C-W) equations. These are linearized solutions of the relative orbital motion problem, in which two spacecraft are both orbiting the same central body in similar orbits. The solution space for this problem is in the LVLH (Local-Vertical-Local-Horizontal) reference frame, centered on the target spacecraft, making it a rotating reference frame and thus a non-inertial space, which is why linearized solutions are much easier to solve. After creating

a working algorithm and cost function for the linearized solution, a non-linear solution of the relative motion problem will be applied to the genetic algorithm to test for variations between the nonlinear model and the linearized model [12].

For the purposes of this analysis, the definition of a swarm is: *a group of two or more spacecraft cooperating towards a common task or goal* [12]. The analysis is performed in the relative motion non-inertial coordinate system defined by the C-W equations [13].

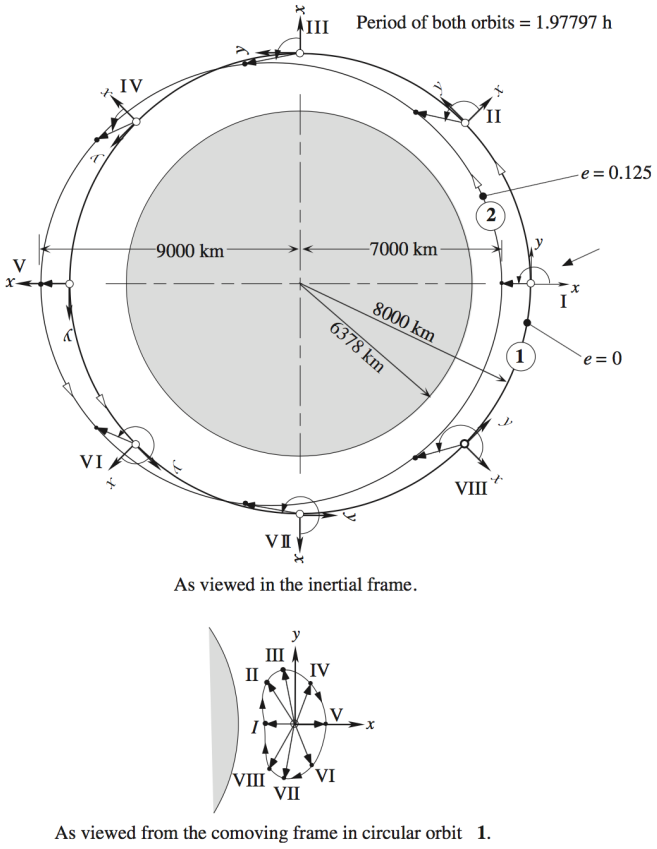

Fig. 1 Slightly eccentric orbit allows relative motion

As seen in Fig. 1, the spacecraft depicted (1 and 2) are in slightly different orbits from each other (upper subfigure), such that in the relative motion space (lower subfigure) spacecraft 2 appears to be "orbiting" around spacecraft 1. The mechanics of the free-trajectory motion following these relative motion orbital tracks are well known and understood, having been used for more than fifty years, prior to the Apollo missions [14].

V. Genetic Algorithms

Genetic Algorithms (GAs) are a method of optimization, applicable to a wide variety of problems, that use a process similar to Darwinian evolution to *evolve* a set of random (or pseudo-random) initial conditions to find an acceptable solution, or even a globally optimal solution, to a problem [15]. These initial conditions form the initial population, of size N_{pop}. This initial population is then propagated, in this case using the C-W equations, to the final state at time t_f. Fig. 2 shows a depiction of the GA process, explained in detail in the following section.

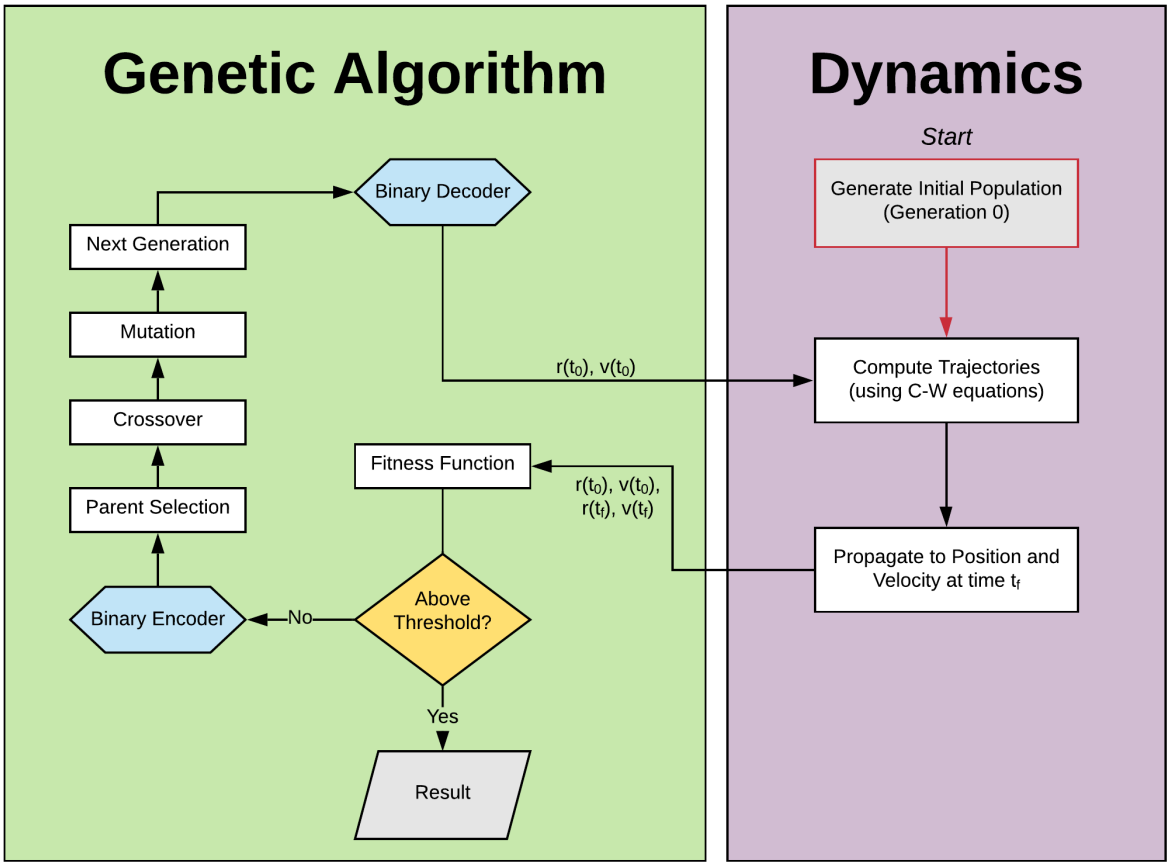

Fig. 2 GA Example Flowchart

Once the initial population is created and propagated, the solutions are ranked based on how close they come to the desired solution, using a fitness function. For simplicity, we created our fitness function such that it ranges from 0 to 1, where a value of 0 has no attributes of a desired solution, and a value of 1 is the desired solution. For the initial problem of finding closed and repeating trajectories in the LVLH frame, this was defined as:

$$F = \left(1 + C_r \left\|\vec{r}(t_f) - \vec{r}(t_0)\right\| + C_v \left\|\vec{v}(t_f) - \vec{v}(t_0)\right\|\right)^{-1} \tag{1}$$

where

$$\begin{aligned} C_r &: \text{coefficient of position} \\ C_v &: \text{coefficient of velocity} \end{aligned}$$

Given a start time t_0 and end time t_f, Eq. (1) defines a fitness function that prefers solutions that are closed trajectories. The closer the final conditions, $\vec{r}(t_0)$ and $\vec{v}(t_0)$ are to the initial conditions, $\vec{r}(t_f)$ and $\vec{v}(t_f)$, the higher the fitness function's value will be, since an desired solution is one where the final conditions and initial conditions are the same. Once the population members are ranked based on their fitness, the bottom half is culled as they are not desirable solutions. However, we need to rebuild the population back to size N_{pop} for the next generation ($N_{pop} = 200$ in our case), so this is where genetic crossover is implemented. To perform crossover, each member of the population (chromosome) should be represented in binary notation in order to represent the data with the most number of genes (string elements), since binary is lowest-order possible data-encoding scheme, with a radix of 2. In the case of swarm trajectories, where our population is composed of 3 position and 3 velocity variables, each of these are represented in binary as 16 bit floats and appended to form a 96 bit string, called a *chromosome*, seen in Fig. 3.

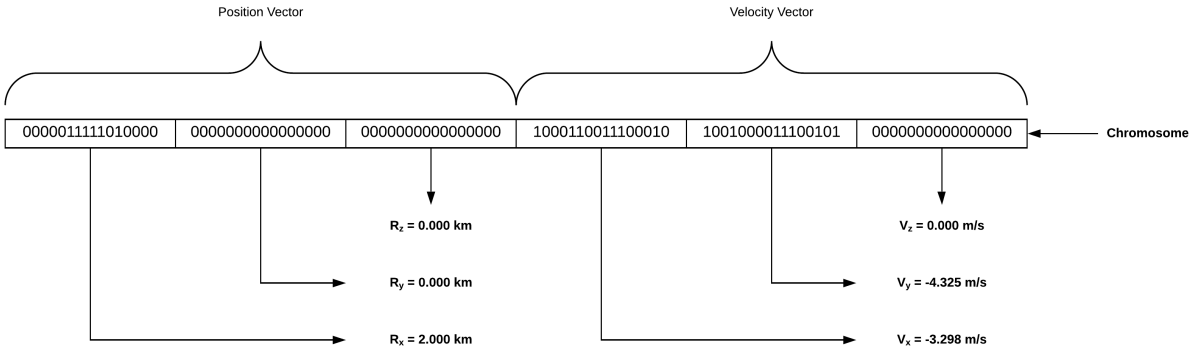

Fig. 3 GA Binary Representation

Crossover is then performed by choosing two of the remaining solutions as *parents*, and taking portions of their *chromosomes* (in this case bits) to form members of the next generation. There are many methods of genetic crossover that can be used in GAs, the simplest of which is random pairing [16]. As random pairing is inefficient at reaching a solution, our method uses roulette selection, which assigns a weighting factor to each parent based on their fitness values. Then pairs are selected to be mated using the weighting factors, such that the chance of selecting a parent with a fitness value of 0.5 is five times higher than selecting one with a fitness value of 0.1. When mating pairs for the crossover, a random number between 1 and 95 is selected for each crossover event, to determine at which point in the *chromosome* to cut and swap, as depicted in Fig. 4. Then, the *chromosomes* of each of the two parents are cut at this crossover point and swapped to make two new *offspring*. This is done until the population size has been rebuilt to N_{pop} for the next generation's computations.

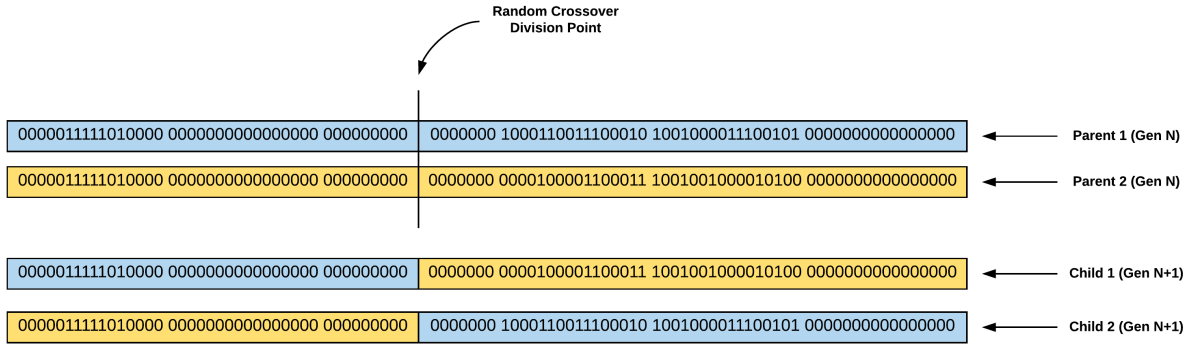

Fig. 4 GA Crossover Example

After crossover is completed, the final step of the GA sequence is to perform a mutation on the chromosomes. The crossover process spreads genetic diversity throughout the population, but does not introduce any new possibilities to the population. This is where mutation comes in; mutation allows new structures or solutions to appear by randomly flipping bits throughout all the chromosome. A variable, p_{mut}, is used to control this probability, and thus a small subset of all bits in all chromosomes are flipped, introducing new and random solution possibilities (see Fig. 5). Good mutations will survive to the next generation and undesirable mutations will not, by means of the fitness function. Although mutation is an important part of the GA process, it must be used sparingly to avoid conflicting with the crossover process. In this case, we use a probability of mutation of 0.2% ($p_{mut} = 0.002$).

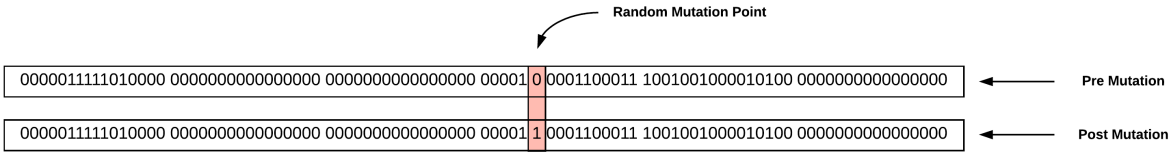

Fig. 5 GA Mutation Example

Once the mutation is completed, the binary data is then decoded back into their separate variables, and the process begins again for the next generation. This process continues until a fitness value of one is achieved for a member of the generation, or the maximum number of generations has been reached ($N_{gen} = 100$). In practice, however, a threshold must be specified, since it is impossible to converge to an exact solution [16]. For an accurate solution, a threshold of 0.001 is used; however, in practice it is more computationally efficient to use a threshold of 0.01 to get near the solution and use another targeted optimization technique to further refine the solution. This is due to the fact that the Genetic Algorithm (GA) method is designed to search across the entire solution space and find a solution among many possible solution spaces, and thus is very good at identifying the location of an optimal solution, but lacks efficiency in arriving at the exact solution itself [15].

VI. Solving for Spacecraft Swarms

A. Initial Trajectory Generation

In order to solve for a set of trajectories for a swarm of spacecraft, multiple genetic algorithms are used, one for each spacecraft, all nested within a larger GA to de-conflict for collisions.

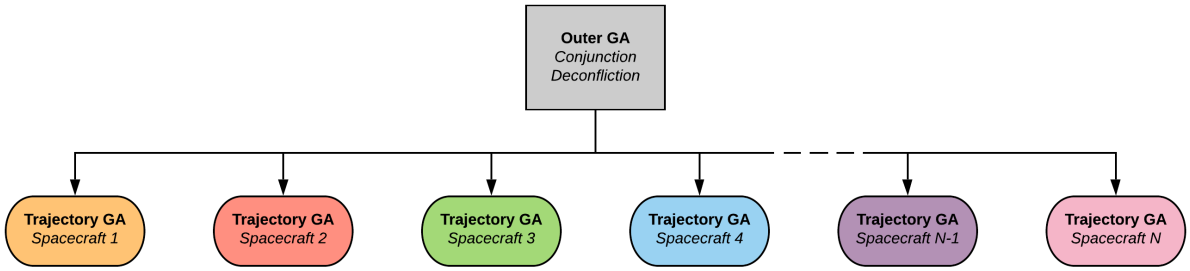

Fig. 6 Hierarchy of Genetic Solvers

Each spacecraft is assigned its own fitness function, and this is defined by the mission requirements for a spacecraft. For example, a spacecraft that has a requirement to be within d_{max} but no closer than d_{min} from a Client spacecraft will have a fitness function as defined in Eq. (2). Note that for the three coefficients, they can be used to tweak which parameters are desired to be solved to a higher accuracy. By default they are all set to 1, but if velocity knowledge is valued at higher precision over position knowledge, then C_v can be set lower (e.g., $C_r = 1$ and $C_v = 0.5$ will result in a twofold increase in precision for velocity)

$$F = \left(1 + C_r \left\|\vec{r}(t_f) - \vec{r}(t_0)\right\| + C_v \left\|\vec{v}(t_f) - \vec{v}(t_0)\right\| + C_d \delta_{dist}\right)^{-1} \tag{2}$$

where

$$\delta_{dist} = \begin{cases} d_{min} - r_{min} & \text{if } r_{min} < d_{min} \\ r_{max} - d_{max} & \text{if } r_{max} > d_{max} \\ 0 & \text{otherwise} \end{cases}$$

C_r : coefficient of position
C_v : coefficient of velocity
C_d : coefficient of distance
r_{min} : closest range to Client spacecraft [km]
r_{max} : farthest range to Client spacecraft [km]
d_{min} : closest permissible distance to Client spacecraft [km]
d_{max} : farthest permissible distance to Client spacecraft [km]

These separate fitness functions allow the optimizer to solve for each spacecraft using its own GA, which can be run in parallel to save on computation time. Once a trajectory is generated for each spacecraft (identified by its initial position and velocity vectors), the outer GA checks for collisions. This is done by propagating each of the trajectories over the course of an orbit, sampling at a fixed timestep (60 s in our case). These position vs. time values are compared for all the spacecraft to determine if there is a chance of collision. For simplicity, a collision is determined to be possible if the two spacecraft are predicted to be within 1 km, however future iterations will involve more rigorous collision detection schemes [17].

If there is a collision predicted, then the outer GA will isolate the two spacecraft that are involved in the collision and determine how to most efficiently mitigate it, as well as which spacecraft has the least restrictions on it to modify its trajectory. The simplest solution is not to change the trajectory at all, but instead adjust the insertion time of one into its trajectory so as to adjust its phase, thereby avoiding a collision. If this is not possible, or if this results in further collisions, then the solver will try slight variations of the trajectories until one is found that does not result in any conjunction.

Running this for a set of 10 spacecraft, with a requirement to be between 0.5 km and 10 km of the Client, and to avoid conjunctions within a 1 km buffer corridor of each spacecraft, Fig. 7 shows a set of closed and repeating relative motion trajectories that satisfy this criteria

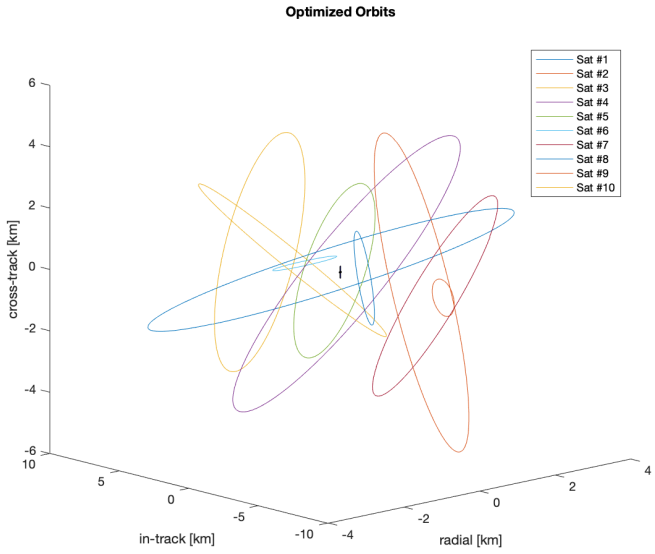

Fig. 7 Swarm Solution for 10 Spacecraft

It should be noted, however, that this is not a unique solution. There is a family of an infinite number of solutions that satisfy this criteria, we are only interested in one of infinite possible sets that satisfies our constraints.

B. Trajectory Modification for New Spacecraft Insertion

Now that a set of trajectories are generated for the swarm, the next problem to tackle is the dynamic nature of the swarm: what to do when the number of spacecraft or their requirements changes?

The problem of adding or removing a spacecraft from a set of swarm trajectories that have already been generated is fundamentally different from the problem above, since we cannot simply regenerate the trajectories for all spacecraft, we already have a set of spacecraft in their respective trajectories and want to modify this as little as possible. When adding a spacecraft to the swarm, its understood that some or all of the other spacecraft in the swarm may have to modify their trajectories, thereby using some of their fuel reserves, to enable a set of safe, mission-specific trajectories for the new swarm. However, it is desirable to do this in such a way that the delta-v used by the swarm as a whole is minimized, as well as the delta-v used by individual members of the swarm so as not to excessively deplete the reserves of a single spacecraft.

This is performed once again by using genetic algorithms, using the same nested GA scheme (see Fig. 6), but with a modification to the outer de-confliction GA to take into account the delta-v cost to attain a given trajectory from an existing one, and a modification to the spacecraft-level GA fitness function that uses the existing trajectory at the starting point for a solution rather than a random seed (see Eq. (3)).

$$F_{insert} = \left(1 + C_r \left\|\vec{r}(t_f) - \vec{r}(t_0)\right\| + C_v \left\|\vec{v}(t_f) - \vec{v}(t_0)\right\| + C_d\delta_{dist} + \mathbf{\Delta v}\right)^{-1} \tag{3}$$

An example of this can be seen in Fig. 8, which depicts a modification of the solution shown in Fig. 7 with an 11^{th} spacecraft added into the swarm. The trajectories of the original 10 spacecraft have been modified slightly to allow for the addition of the 11^{th}, conserving delta-v.

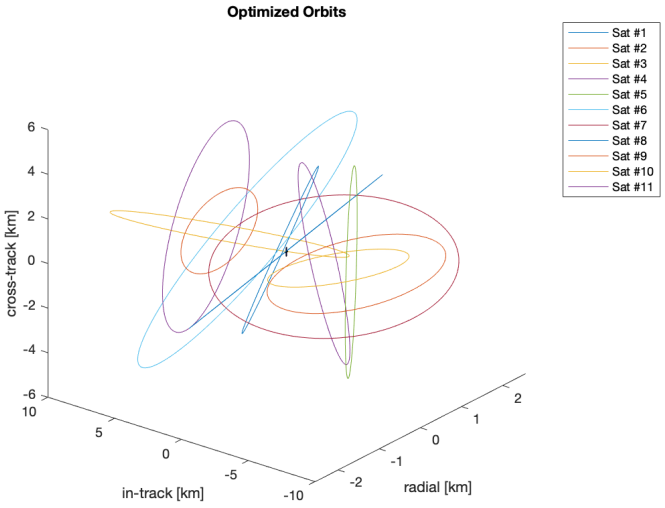

Fig. 8 Modified swarm solution for the addition of an 11th spacecraft

C. Considerations for Construction and Aggregation

When applying this methodology to in-space construction or aggregation of swarm members, consideration needs to be taken not only for the addition and removal of members from the swarm, but also for the dynamically changing dimensions, mass, and moment of inertia of the Client being constructed. As the structure grows, so will the keep-out zone specified for all spacecraft, especially if it is spinning. Although these simulations have not yet been performed, this is the next stage of research that is currently underway at the USC Space Engineering Research Center (SERC).

VII. Conclusion

As the use of on-orbit servicing grows in Earth orbit, so will the use of satellite swarms to perform more complex maneuvers. The methodology outlined above, using genetic algorithms to narrow down a set of acceptable, efficient, and stable trajectories for spacecraft comprising a swarm, is one of many possible methods, but is adaptable to various scales and mission restrictions. The implementation of on-orbit construction is drawing closer and closer day by day, become less like science fiction and more like reality. Great strides have been made to design spacecraft that can

perform on-orbit assembly, and to develop structures that can be assembled in space; it's now time to do the same for the methodology of how to perform these cooperative actions in a non-intuitive micro-gravity environment in as safe and autonomous a manner as possible. The next steps, with research currently underway at SERC, are to combine this trajectory generation method with an autonomous GNC system to incorporate distributed computing and take advantage of the nodal architecture of the swarm. This would enable active maintenance of the swarm trajectories, compensating for gravitational and environmental perturbations, as well as unforeseen circumstances, such as the loss of control of one spacecraft in the swarm.

Acknowledgments

Multi-spacecraft simulations were performed with resources provided by the Center for High-Performance Computing at the University of Southern California (https://hpcc.usc.edu).

References

[1] Alfriend, K., Vadali, S. R., Gurfil, P., How, J., and Breger, L., *Spacecraft formation flying: Dynamics, control and navigation*, Vol. 2, Elsevier, 2009.

[2] Izzo, D., and Pettazzi, L., "Autonomous and distributed motion planning for satellite swarm," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 2, 2007, pp. 449–459.

[3] Hadaegh, F. Y., Chung, S.-J., and Manohara, H. M., "On development of 100-gram-class spacecraft for swarm applications," *IEEE Systems Journal*, Vol. 10, No. 2, 2014, pp. 673–684.

[4] Reed, B. B., Smith, R. C., Naasz, B. J., Pellegrino, J. F., and Bacon, C. E., "The Restore-L servicing mission," *AIAA SPACE 2016*, 2016, p. 5478.

[5] Barnhart, D., Will, P., Sullivan, B., Hunter, R., and Hill, L., "Creating a Sustainable Assembly Architecture for Next-Gen Space: The Phoenix Effect," *30th Space Symposium*, 2014.

[6] Katz, J., Mohan, S., and Miller, D., "On-orbit assembly of flexible space structures with SWARM," *AIAA Infotech@ Aerospace 2010*, 2010, p. 3524.

[7] Skomorohov, R., Welch, C., and Hein, A., "In-orbit Spacecraft Manufacturing: Near-Term Business Cases," *International Space University*, 2016.

[8] Piskorz, D., and Jones, K. L., "On-orbit assembly of space assets: A path to affordable and adaptable space infrastructure," *The Aerospace Corporation*, 2018.

[9] Rauwolf, G. A., and Coverstone-Carroll, V. L., "Near-optimal low-thrust orbit transfers generated by a genetic algorithm," *Journal of Spacecraft and Rockets*, Vol. 33, No. 6, 1996, pp. 859–862.

[10] Ferrandez, S. M., Harbison, T., Weber, T., Sturges, R., and Rich, R., "Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm," *Journal of Industrial Engineering and Management (JIEM)*, Vol. 9, No. 2, 2016, pp. 374–388.

[11] Saez, Y., Perez, D., Sanjuan, O., and Isasi, P., "Driving cars by means of genetic algorithms," *International Conference on Parallel Problem Solving from Nature*, Springer, 2008, pp. 1101–1110.

[12] Rughani, R., Villafaña, L., and Barnhart, D. A., "Swarm RPO and Docking Simulation on a 3DOF Air Bearing Platform," *70th International Astronautical Congress (IAC), Washington D.C., United States*, 21-25 October 2019.

[13] Curtis, H. D., *Orbital Mechanics for Engineering Students*, (pp.383-387) Elsevier Aerospace Engineering Series, Elsevier, 2014.

[14] Mayer, J., and Parten, R., "Development of the Gemini operational rendezvous plan." *Journal of spacecraft and rockets*, Vol. 5, No. 9, 1968, pp. 1023–1028.

[15] Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed., Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[16] Haupt, R., and Haupt, S., *Practical Genetic Algorithms*, Wiley InterScience electronic collection, Wiley, 2004.

[17] Chan, F. K., et al., *Spacecraft collision probability*, Aerospace Press El Segundo, CA, 2008.