JOURNAL OF SPACECRAFT AND ROCKETS
Vol. 40, No. 2, March–April 2003

# Satellite Close-Approach Filtering Using Genetic Algorithms

Anthony L. Faulds* and David B. Spencer†
*Pennsylvania State University, University Park, Pennsylvania 16802*

**When doing collision analysis between two objects in orbit, it is important to compare close approaches between all active orbiting satellites with all tracked objects. All possible combination of object pairs must be analyzed. An algorithm for finding object pairs that have a low chance of collision has been developed. To allow for quick computation so that all close approach possibilities do not need to be analyzed using a high-order propagator, an algorithm with a low-order propagator, incorporating parallel processing techniques and a genetic algorithm to find closest approach, is developed. The resulting algorithm is applied to a sample satellite catalog, and the simulation showed a significant decrease in execution time, as well as rapid convergence on minimum distances between two objects.**

## Nomenclature

$EP$ = parallel efficiency
$m$ = number of active satellites in catalog
$n$ = number of objects in catalog
$P$ = number of processors
$Q$ = smaller apogee of the two orbits, km
$q$ = larger perigee of the two orbits, km
$TP$ = execution time on $P$ processors, s
$T1$ = execution time on one processor, s

## Introduction

ONE important concern of Earth-orbiting vehicle operations is collision avoidance. Currently there are approximately 9000 cataloged objects orbiting the Earth, and 90% are inert. Various components go into the process of determining accurate close approaches. Among these are accuracy of computational models, accuracy of position and velocity vectors of space objects, and computational ability to conduct this analysis in a timely manner. In an ideal world the close-approach analysis process would benefit tremendously from using actual satellite ephemeris data (which are not always available), high-order propagators (which are not always practical), and advanced computer technology (which might not be in place). Compounding the problem is the fact that analyzing a very large number of object pairs for close-approach determination using a high-order propagator must be compatible with continuous operations.

This paper begins with a discussion of close-approach determination analysis methods. Specifically, a long-term, seven-day, close-approach prediction is the goal. The objects are analyzed on a one-on-one basis, which means that every object is assessed for a close-approach possibility with every other object. However, a close approach between a piece of debris and another piece of debris is only of academic interest: only a close approach involving an operational satellite is of operational interest. This assessment needs to be done approximately $m \times n$ times, so that $m$-on-$n$ satellite close-approach possibilities could be computed with accuracy and speed. The full close-approach analysis includes the gathering of satellite data, calculating mean elements, using a filter to remove improbable satellite close approaches, incorporating a high-order propagation analysis, and developing an algorithm to calculate the global minimum distance between two orbiting objects, as well as the locations of all minimum distances that are less than a specified "keep-out" distance. This paper addresses all parts of the algorithm.

## Objectives

The first questions addressed are why a filter for removing improbable close approaches is needed and what filter would best suit this issue. To do comprehensive and accurate close-approach analysis, a large number of objects will be dealt with. Ideally, calculations of all one-on-one close approach potentials using a computationally intensive high-order propagator are preferred. Fast algorithms are needed to eliminate satellites that obviously have no chance of coming close to each other. These filters should use low-order, faster running algorithms that calculate close approaches. The filter must also satisfy the following assumptions:

1) If a satellite is eliminated from concern using the filter, then using a high-order propagator would also indicate that the satellite should be eliminated from concern.

2) The computation time for a filter should be orders of magnitude less than that of the higher-order propagator.

These two assumptions might seem obvious but need to be stated to understand why approximate methods are adequate for the filter. The filter only removes satellites from the possible object pairs that have a low chance of coming close to each other; it does not determine whether satellites will approach each other. If the filter predicts a close approach, it does not necessarily mean there will be a close approach but that further analysis should be done. Also, this filter must run much faster than the high-order propagation; otherwise, applying a filter defeats the purpose of having a filter in the first place.

There are two parts of the filter that will be implemented in the algorithm in this paper. The first is called an apogee–perigee (AP) filter, one of several prefilters in Ref. 1 that has been implemented previously for many close-approach algorithms. The second part of the filter uses a propagator to estimate the close approaches better than an apogee–perigee filter. An orbit propagator that is faster than a high-order propagator is used. The Simplified General Perturbation (SGP4) algorithm has been chosen as the orbit propagator for the filter. (Data are available online at http://www.znark.com/sat/files/spacetrk.pdf.) This propagator has many features that are nice for this type of problem, including the evaluation of the dominant geopotential perturbations and atmospheric drag. The greatest advantage of SGP4 is that the algorithm is not an ordinary differential equation (ODE) that needs to be numerically integrated. Because the resulting calculations from the algorithm are an estimated solution to the ODE of Earth orbits, the algorithm takes the same amount of time to calculate 2 s into the future as two days into the future. The equations are simply a function of mean elements and are readily available as two-line element sets (TLEs). TLEs are derived from averages using the SGP4

Received 6 May 2002; revision received 25 November 2002; accepted for publication 5 December 2002. 

*Graduate Student, Department of Aerospace Engineering; currently Software Engineer, RAM Laboratories, Inc., 6540 Lusk Boulevard, Suite C200, San Diego, CA 92121. Student Member AIAA.

†Assistant Professor, Department of Aerospace Engineering. Associate Fellow AIAA.

algorithm. With these TLEs the AP analysis can be done, and the mean position at a given time can be found. With SGP4 continuous functions where the domain is the time relative to epoch and the range is radius and velocity vectors of the satellite in Cartesian coordinates can be obtained. Using the radius vectors of the two satellites, a distance between the satellites can be calculated. The goals are to find either the minimum distance between two objects, or the location where close approaches occur below some threshold, as the greatest opportunity for collision occurs at this/these point(s).

The results from computing the distances between two satellites have many local minima and maxima. Most optimization algorithms such as Newton's method and method of steepest descent can find minima given the right condition, but they only find local minimum. Any of these methods might converge on a local, not global, minimum. One optimization method that has been developed recently is called genetic algorithms and are currently being used to solve nondeterministic polynomial time problems. This means the time, or number of operations, cannot be represented by a polynomial. These new methods can also solve more complex problems, where it is very difficult to calculate or estimate gradients.

## Analysis

Various geometric prefilters are used to reduce the number of computations needed to determine whether a close approach falls within some predetermined keep-out range. One of these geometric filters that is still used today is the AP filter. This requires almost no computation, just a comparison of the apogee of one orbit to the perigee of the other. Introducing a positive value called critical distance (CD) leads to the assumption that if the distance between two satellites is greater that CD, the object pair will be filtered out. The satellite will be filtered out if the inequality in Eq. (1) is satisfied[1]:

$$q - Q > \mathrm{CD} \tag{1}$$

Otherwise, the object pair stays in the system for further analysis. This prefilter is a simple calculation that is very quick and accurate. Any satellite pair that passes this criterion will not collide, even with drag included, because CD can be set large enough to compensate for drag effects. The main advantage is that the analysis is not wasting time, for example, calculating the chance of collision between a low-Earth orbit vehicle and a geosynchronous Earth orbit vehicle. The AP filter takes on the order of 10 floating point operations to filter a satellite pair.

To propagate the orbits, a fast, low-order propagator is desired. The SGP4 algorithm, chosen for this study, uses data from satellite orbits and averages these data to develop mean elements. These mean elements can then be inputted into SGP4 to estimate future positions and speeds of a satellite. This is a nice algorithm because it is a fully analytical model; thus, no numerical integration is required. SGP4 includes estimated drag effects, short-period periodic motions, long-period periodic motion, secular, and Earth oblateness effects. The algorithm in its entirety can be found online at http://www.znark.com/sat/files/spacetrk.pdf.

Two-line elements used for SGP4 are mean elements stored in two lines of data. The information is in fixed width format. Each mean element is stored in a fixed number of spaces whether it is long enough to fill it or too long to fit. This is one of the drawbacks in rounding that occurs when operations uses this format. These data and a time from epoch are entered into SGP4 and results in a radius and velocity vector in the inertial fixed system.

Various algorithms can accomplish computing the minimum distance between two objects. One new algorithm that can be used is known as a genetic algorithm (GA). Holland[2] first introduced the concept of genetic algorithms (GAs). The idea was not practically applied until Koza[3] used GAs to have software programs evolve by showing that a program could learn to do sequence induction, symbolic regression, and even solving equations like differential equations, integral equations, and functional equations. GAs are designed to mimic the natural processes of life and operate on Darwin's theory of survival of the fittest. Using this method results in an algorithm that produces a best result or an optimizing algorithm (such as a minimum or maximum value of a function).

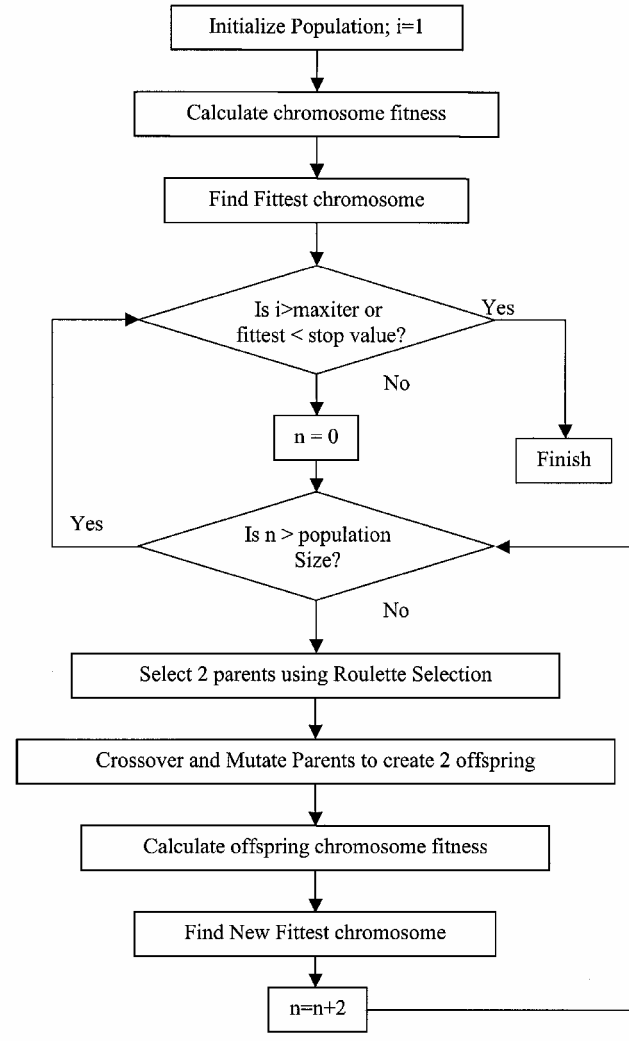



**Fig. 1 GA flowchart.**

In nature, the population of a species bears new generations. Each generation gains desirable traits from the previous generation. GAs work the same way. For an optimization problem a search space is defined as the domain of the problem. A population is made up of random points in the search space. The desirable trait desired is a minimum or maximum solution to the optimization problem. To create a new generation, crossovers and mutations occur. The combination of crossover, which narrows the population to those with good traits, and mutation, keeping the population diverse, allows the algorithm to converge to absolute maximum or minimum.

The initial population is usually generated randomly but can be created if there is knowledge about the problem and it is desired to speed up convergence. Each individual in the population is the input to the problem. The fitness of an individual is found using a fitness function. The fitness is a single number found by evaluating an individual with the function being minimized. In this problem the fitness function is the distance between two space objects. The fitness is usually the most computationally intensive part of the GA. The resulting algorithm is shown in Fig. 1. This shows each step along with where control can be applied to change how the GA converges on solutions. More details on the specifics of the algorithm can be found in Faulds.[4]

These algorithms are combined with a powerful tool: parallel processing. Parallel processing has been a topic of research since the first computers and has recently become a hot topic because of the ability to network personal computers to create an inexpensive supercomputer. Most algorithms have a parallel and serial part to them. There is a special class of parallel algorithms called embarrassingly parallel (EP). EP algorithms are ones in which the calculations are completely independent. Instead of parallelizing parts of the algorithm, the complete algorithm is run on each processor separately.

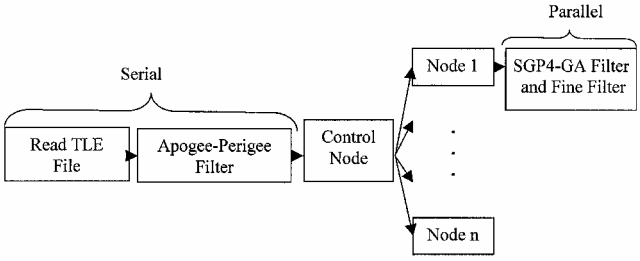



**Fig. 2 Collision avoidance flowchart.**

To measure the efficiency of the algorithm, the run time is calculated for the algorithm running on one processor and $n$ processors. These values can then put used to calculate algorithmic speed up (SPA), as seen in Eq. (2):

$$SPA = T1/TP \tag{2}$$

This value is then used to calculate parallel efficiency $EP$ in Eq. (3):

$$EP = (SPA/P) \times 100\% \tag{3}$$

This value represents how much of the program is parallel. In the real world no algorithm is 100% parallel because communications are needed between the control node and the processing nodes.

To parallelize the algorithm, the control node must take each satellite pair and distribute it out to a processor to determine closest approach. The values of the TLE set are sent to each node. When creating a parallel code, it is very important to limit how much information is being passed back and forth between nodes. Figure 2 shows the AP filter in the serial part of the code with reading the elements while the SGP4-GA filter in parallel. The AP filter can be parallelized, but in hindsight passing the two TLE sets takes relatively the same amount of time to send to the each node as applying the AP filter to them. This parallel design attempts to optimize the parallel efficiency.

## Results

An example assessment of the close-approach determination is performed, and the following results were obtain using Celestrak data available online at http://www.celestrak.com from December 2001. This catalog contains $n = 804$ unclassified satellites. The first part of the algorithm is reading data from a file. The satellites are then split into conjunction pairs. The results are $(n^2 - n)/2 = 322{,}806$ pairs of satellites. The program created to analyze this data is written in Java on Pentium 3-PCs with 256 Mb RAM and 10/100 Mb network connections. The method used to create a parallel algorithm incorporates a commercial software package called JGravity™ available online at http://www.jgravity.com. The speeds and accuracies are correlated to these pieces of hardware and software.

The first computational part of this algorithm is the AP filter. This part is the easiest to implement and removes a large amount of satellite pairs. Figure 3 is a visual representation of the International Space Station and the Combined Release and Radiation Effects Satellite orbits. This pair of satellites fails to pass the AP filter. This figure shows that where the orbits cross a more complicated analysis needs to be carried out.

For a CD of 100 km, the AP filter removes 229,598 of 322,610 satellite pairs. This computation is accomplished in less than a second on average (~0.9 s). In comparison, it takes approximately 4 s to read the 804 satellites in the catalog. There are 93,012 satellites that come within the CD used in the AP filter; thus, approximately 72% of the satellite pairs have been removed.

To validate the SGP4-GA, it is compared to proven optimization techniques, specifically bisection method. Two amateur radio satellites that are not filtered out by the AP filter are used to show the ability of SGP4-GA algorithm. The TLEs for these satellites are given in Table 1.

These orbits are evaluated every 500 s epoch time for visualizing the function to be minimized. Figure 4 shows the two orbits (Fig. 4a) and the search space (Fig. 4b) the GA will use to find the closest

**Table 1 Two-line elements for two amateur satellites**

| Descriptor | AO-10 | AO-40 |
|---|---|---|
| Satellite number | 14129U | 26609U |
| International designator | 83058B | 00072B |
| Epoch YYDDD.DDDDDDDD | 01133.71552800 | 01134.31089787 |
| $\frac{1}{2}\frac{d}{dt}$(mean motion) (rev/day$^2$) | −0.00000146 | 0.00000009 |
| $\frac{1}{6}\frac{d^2}{dt^2}$(mean motion) (rev/day$^3$) | 0 | 0 |
| B* (nondimensional) | 0.0001 | 0 |
| Element number | 735 | 7924 |
| Inclination, deg | 26.5901 | 5.1930 |
| Right ascension of ascending node, deg | 272.6815 | 193.9752 |
| Eccentricity | 0.6001393 | 0.8146456 |
| Argument of perigee, deg | 149.3922 | 267.6333 |
| Mean anomaly, deg | 266.4049 | 8.8683 |
| Mean motion, rev/day | 2.05865171106761 | 1.27005342 |
| Epoch revolution | 6761 | 2473 |

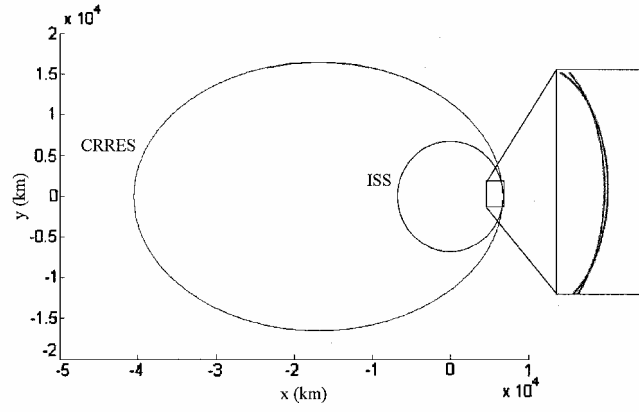



**Fig. 3 Close-up of AP filter.**

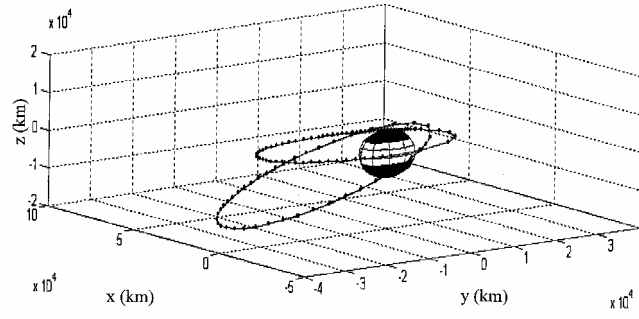



**Fig. 4a Three-dimensional orbits.**

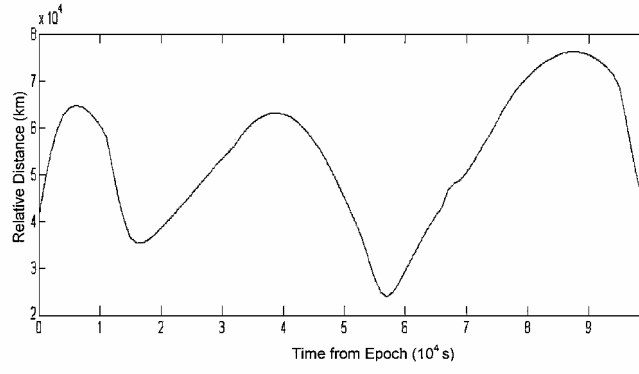



**Fig. 4b Relative distance vs time.**

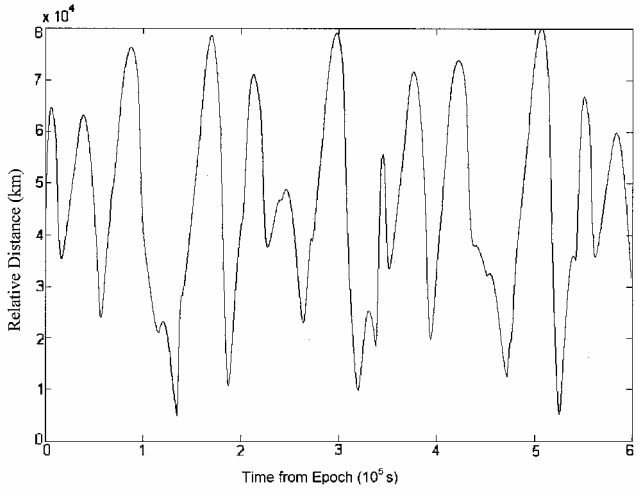
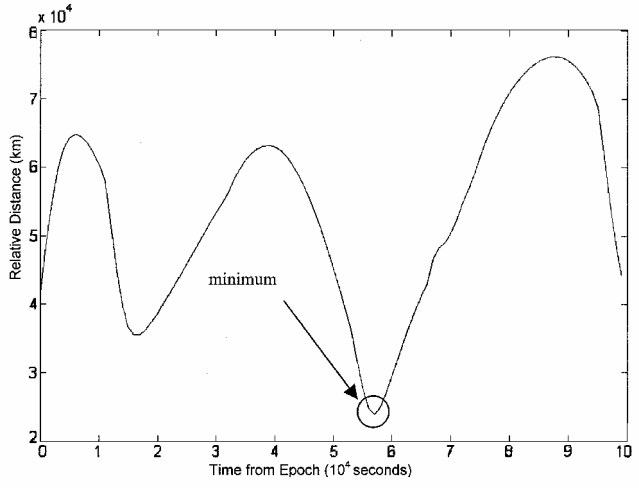
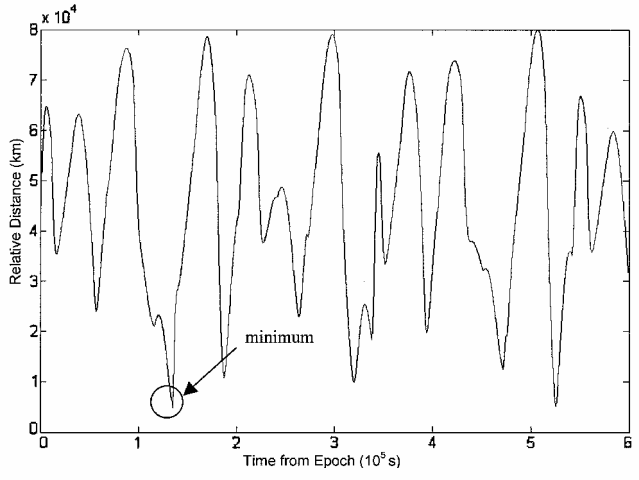



**Fig. 5 Visualization of the search space.**




**a)**

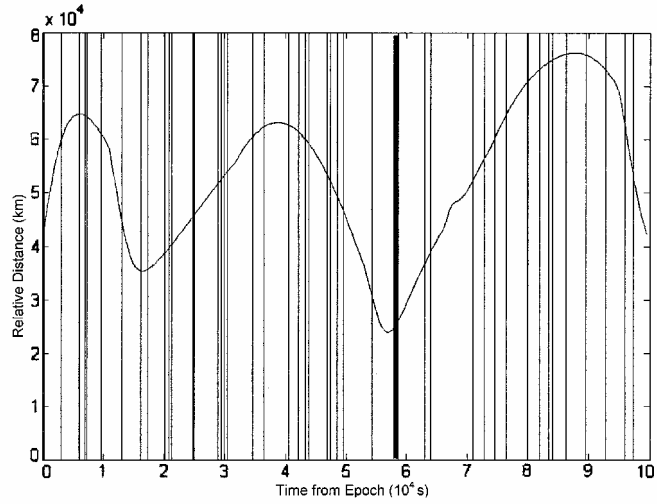



**b)**

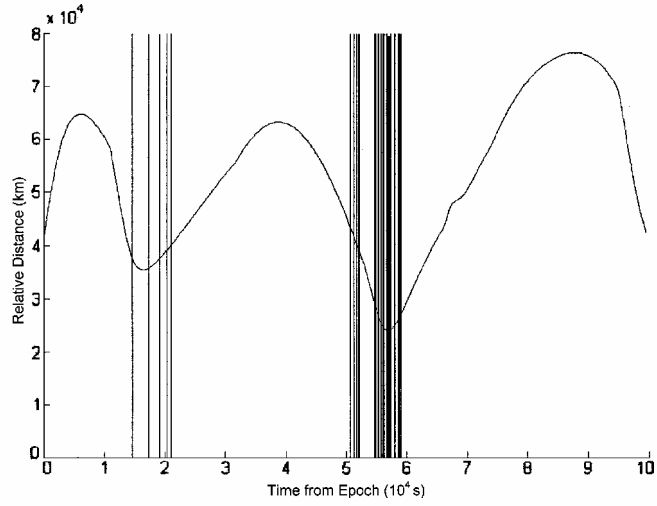
**Fig. 6 Minimum distances for a) 0–100,000 s and b) 0–600,000 s.**

approach. An important feature to notice about the search space is that it has many local minimums. Figure 5 shows the same search space but over the goal time span: seven days. This search space has many more local minimums.

The following results have been obtained using a population size of 50, a probability of crossover of 80%, and a probability of mutation of 10%. Using trial-and-error, these values were found as the fastest and most robust for this analysis. Figure 6 shows each search space and the minima.




**a) Generation 1**

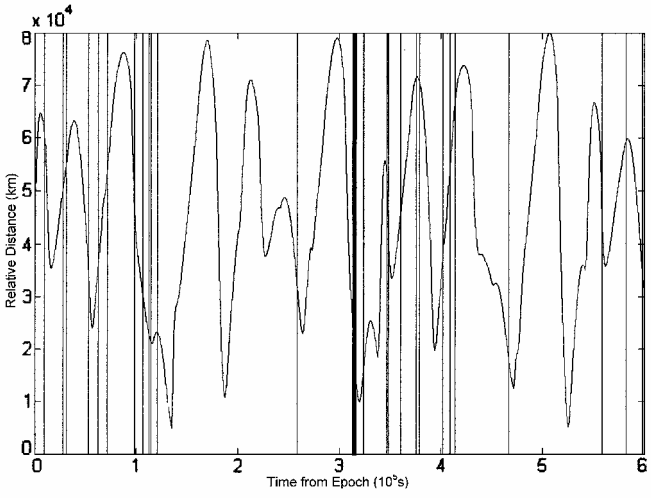



**b) Generation 2**

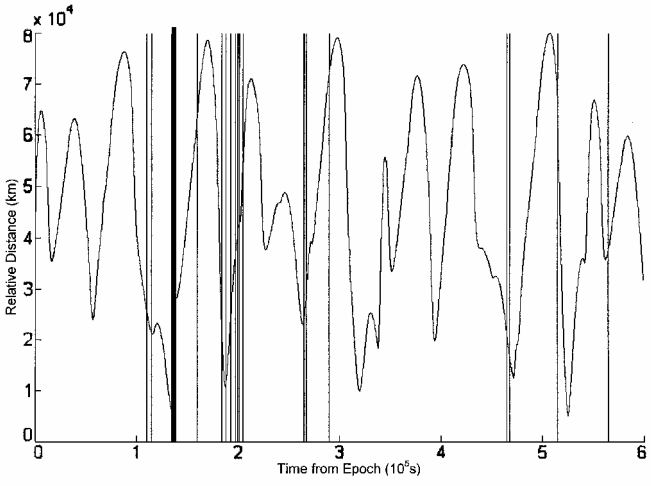
**Fig. 7 Selected populations of GA-SGP4 for one-day prediction.**

Figures 7 and 8 are used to help visualize the GA searching for a minimum. Both time spans are shown along with the initial population and the tenth population. The curved line in Figs. 7 and 8 represents the function in the search space, and the vertical lines represent chromosomes of the population. The solid vertical line represents the fittest chromosome for that generation. One important characteristic to notice is because the initial generation is chosen at random it is a sparse search of the search space. When the population is propagating to the tenth generation, it congregates around local minimum. Even for seven-day prediction (Fig. 8) the GA converges quickly to the global minimum. For a time period of about 1.2 days or 100,000 s, the SGP4-GA found the closest approach at 56,855 s beyond epoch time at a distance of 2.3925 km. The GA found this solution in five generations on average. Using bisection method and picking a range so that only one minimum is found within that range resulted in a minimum at 56,923 s at a distance of 2.3921 km. These are very close results. The same satellite pair is then used to predict closest approach for 0–600,000 s or seven days. On average after seven generations, the fittest chromosome in the GA was 134,768 s with a closest approach of 0.4216 km. From Fig. 8 it is difficult to visually discern what the global minimum is. Two local minima occur about one day from epoch and about six days from epoch, respectively. Using bisection method on both yields the former as the global minimum at 134,747 s at a relative distance of 0.4211 km. These results are typical of the complete catalog comparison.

Because of the inabilities of traditional optimizing techniques, older algorithms do a sparse search to find minimum. For example, Ref. 1 develops candidate times and does not rely on a step method




a) Generation 1




b) Generation 2

**Fig. 8 Selected populations of GA-SGP4 for seven-day prediction.**

except for certain classes of orbits. A step size of 10 minutes is small enough to catch close approaches of satellites. To predict seven days into the future, the method used in Ref. 1 requires 1000 SGP4 calculations or 1,000,000 FLOPS. In comparison, the SGP4-GA found the optimal solution is seven generations. With a population size of 50, this results in 350 SGP4 calculations or 350,000 estimated FLOPS. The work load has been significantly reduced.

Finally, the results of the whole filter in parallel are presented. When computing close approaches for a single satellite against 804 objects for a seven-day period, the algorithm had a total execution time of 14.46 s. Table 2 shows the resulting times for the entire filter.

The read time and AP filter take the same time no matter how many processors are used because they are the serial part of the code

**Table 2 Execution time for whole algorithm ($SPA = 14.37/2.82 = 5.10$, $EP = 100 * SPA/9 = 56.6\%$)**

| Simulation characteristics | Number of processors | Time required to read input data, s | Time required to execute AP filter, s | Time required to execute SGP4-GA algorithm, s |
|---|---|---|---|---|
| 1 against 804 | 1 | 0.9 | 0.01 | 13.46 |
| 1 against 804 | 9 | 0.9 | 0.01 | 1.91 |
| 804 against 804 | 1 | 0.9 | 4 | $\sim 10^5$ |
| 804 against 804 | 9 | 0.9 | 4 | 900 |

that is only run on one processor. They take up little of the whole algorithm speed. Most of the time is spent with the SGP4-GA filter. For the close approach computation between one satellite against 804 objects, the SPA is 5.10, and the resulting EP is 56.6%. Communication overhead prevents this efficiency from reaching 100%. JGravity™ passes byte codes of the algorithm to be run on each processor, which lowers EP, and all satellite pairs' TLEs that pass the AP filter are sent to nodes, which also lowers EP. Determining the close approach distance between 804 objects against 804 objects, the algorithm on one processor took on the order of four to five hours, whereas the parallel algorithm took about 15 min to run.

## Conclusions

The resulting algorithm fits the goals set out and takes an unmanageable problem and makes it realistic to solve in a short amount of time. The AP filter removes more than 70% of the catalog in 0.9 s. The SGP4-GA filter requires fewer calculations than traditional sparse search methods and always found a local minimum in the numerical research for this problem. Parallel processing then takes this fast algorithm and gives the ability to calculate closest approach of whole satellite catalog in minutes, in other words solving the *m*-on-*n* satellite problem.

Future work for this project includes using a filter between the AP filter and the SGP4-GA filter. Because 95% of the execution time is spent in SGP4-GA, the execution time can be reduced by using a filter that has on the order of 10∼100 FLOPS.

## References

[1] Hoots, F. R., Crawford, L. L., and Roehrich, R. L., "An Analytical Method to Determine Future Close Approaches Between Satellites," *Advances in the Astronautical Sciences*, Vol. 54, Pt. 1, 1984, pp. 281–298.

[2] Holland, J. H., *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, 1st ed., MIT Press, Cambridge, MA, 1992, pp. 89–120.

[3] Koza, J. R., *Genetic Programming III: Darwinian Invention and Problem Solving*, 1st ed., Morgan Kaufmann, San Mateo, CA, 1999, pp. 19–66.

[4] Faulds, A. L., "Satellite Collision Analysis Using Genetic Algorithms, Parallel Processing and Stochastic Methods," M.S. Thesis, Dept. of Aerospace Engineering, Pennsylvania State Univ., University Park, May 2002.

C. A. Kluever
*Associate Editor*