A High Performance Conjunction Analysis Technique for Cluster and Multi-Core Computers

Eric R. George
The Aerospace Corporation, Colorado Springs, CO, 80916

Abstract

Satellite collision avoidance analysis has received increased emphasis since the Iridium-COSMOS collision of February 2009, including serious consideration of regular "all vs. all" collision avoidance analysis of the entire space catalog by the US Air Force. This paper describes a technique for the very fast screening of large conjunction analysis problems. This technique makes no assumptions about orbit shape or motion and is suitable for the analysis of powered or hyperbolic flight with no special consideration required. Implementations of this technique (CSieve) on both cluster computers and large multi-core computers are presented including discussion of analysis scaling. Results are presented for all vs. all analysis of up to 93K objects. Discussion of batch vs. continuous analysis is also presented

1. Nomenclature

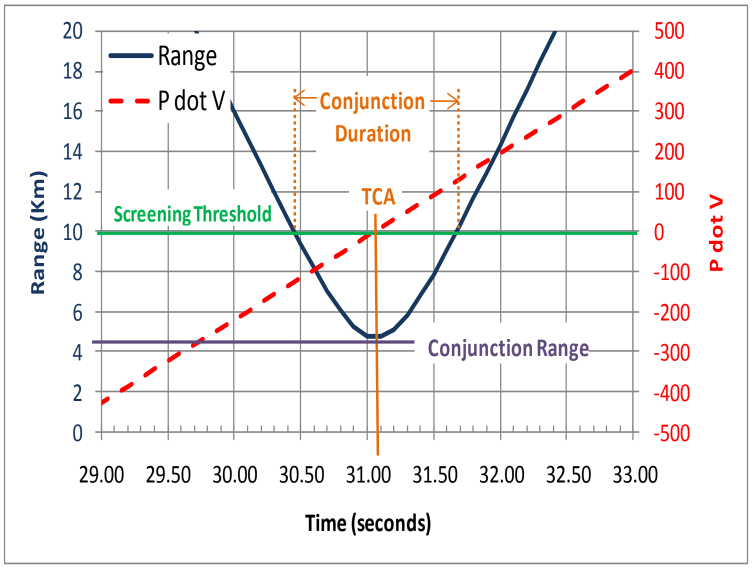
Δt	Analysis time step
X, Y, Z	Component directions of a Cartesian coordinate frame.
X_{low}/X_{high}	Minimum and Maximum values the X position of a satellite within the current time step
Y_{low}/Y_{high}	Minimum and Maximum values the Y position of a satellite within the current time step
Z_{low}/Z_{high}	Minimum and Maximum values the X position of a satellite within the current time step
$A.X_{low}$	The X_{low} value for object A
N	Number of satellites in the analysis catalog
PCA	Point of Closest Approach
TCA	Time of Closest Approach
R	Range Screening Threshold
C	The set of objects whose trajectory circumscribing volumes intersect the volume of a specified object.
C_x	The set of objects whose trajectory circumscribing volumes intersect the volume of a specified object in the X direction.
$\vec{P}_r$	Relative position vector
$\vec{V}_r$	Relative velocity vector

2. Introduction

Conjunction analysis (CA) screening is the process by which conjunctions between orbiting objects are identified. A conjunction is defined here as the Point of Closest Approach (*PCA*) between two space objects. More formally, it is the local minimum of the relative range vs. time curve of two space objects such that the minimum range is less then or equal to the screening threshold. This minimum corresponds to the root of the dot product of the relative position and velocity vectors. The time of the local minimum is known as the Time of Closest Approach (*TCA*) and the magnitude of the relative position vector at TCA is the Conjunction Range.

A conjunction may also be defined as the time interval during which the range between two objects is less than the screening threshold. This interval is easily determined given the TCA of an event.

Figure 1 Conjunction Illustration



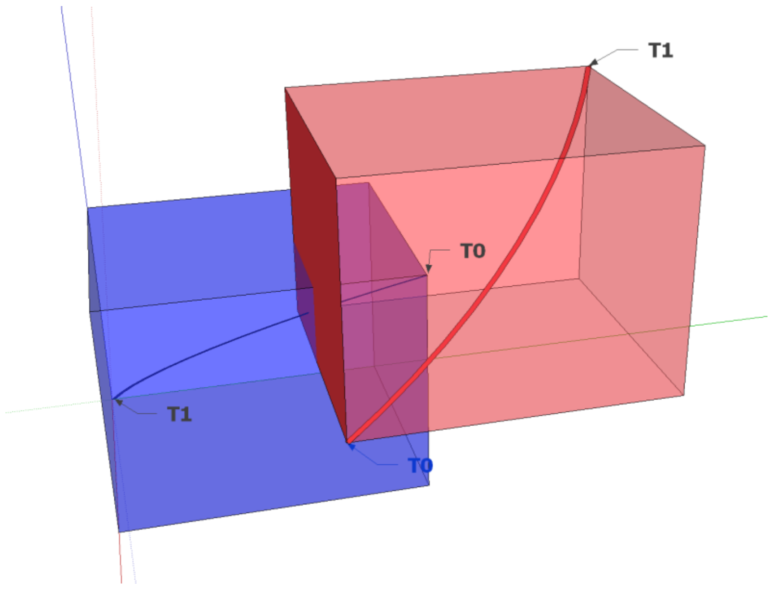
The CSieve software is designed to perform this screening very quickly for large, full catalog analysis problems (all vs all), making extensive use of high performance parallel computing resources.

1. Geometric Screening Algorithm

Many conjunction analysis algorithms perform their screening as essentially a series of 1vs 1 analysis, usually with extensive geometry or time based pre-filtering[1] to quickly eliminate pairs of satellites as unable to conjunct. Examples of these filters include apogee/perigee filters and orbit path filters. While simple in concept, these filters must be carefully configured and applied to avoid missing events.[2] These techniques are often not directly amenable to analysis of maneuvering/powered vehicles and non-earth-centered trajectories. While these 1 vs 1 comparisons are independent, and therefore very simple to parallelize, there is significant redundancy of data access and propagation in these schemes.

The core of the CSieve software is the geometric screening algorithm, inspired by Healy[3]. In its simplest terms, the algorithm defines a rectangular volume circumscribing the path of a space object over some small time interval Δt. Overlapping of these rectangular volumes (Figure 2) indicates that the two objects within the volumes were close to each other during the current Δt window.

Figure 2 Intersection of Rectangular Circumscribing Volumes



Examining the value of the dot product of the relative position and velocity vectors at the beginning and end of the Δt window reveals the presence of a local minima in the current window (Equations 1, 2).

(1) $$\vec{P}_r(t) \cdot \vec{V}_r(t) < 0$$

(2) $$\vec{P}_r(t + \Delta t) \cdot \vec{V}_r(t + \Delta t) > 0$$

At this point a simple iteration is performed to find the root of the dot product (Equation 3), which corresponds to the minima of the relative position vector, as shown in Figure 1.

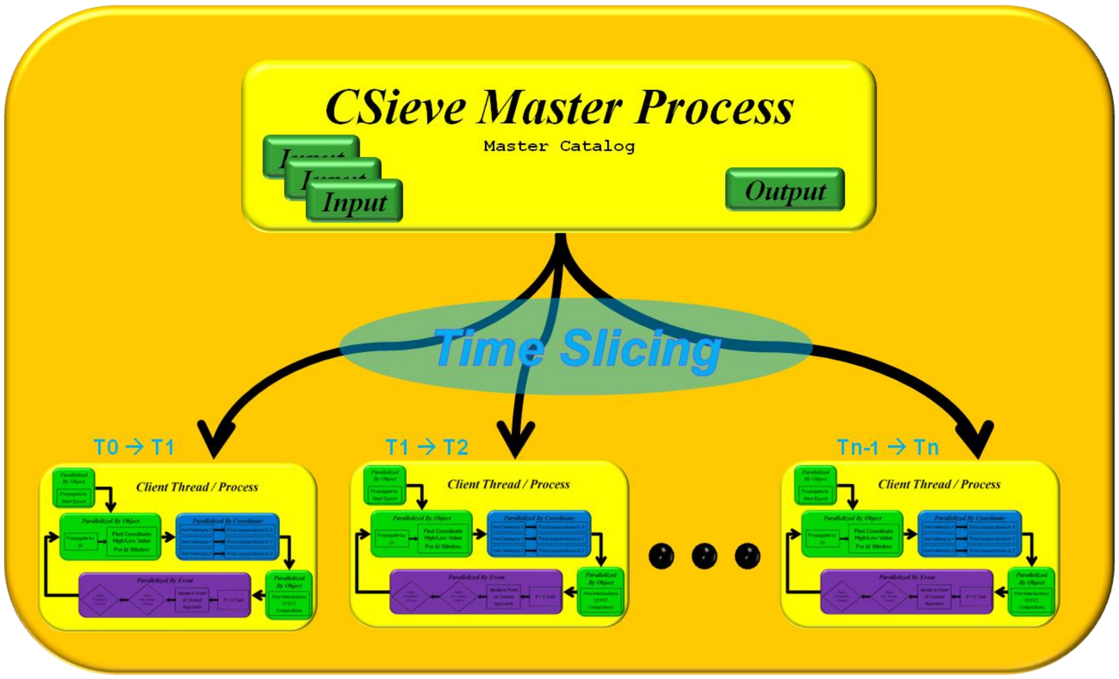
(3) $$\vec{P}_r(t) \cdot \vec{V}_r(t) = 0$$

This technique can be very simply parallelized in the time domain by using multiple threads or processes to analyze subsections of the overall analysis time frame.

2. Algorithm Details

While the description above is useful for visualizing the fundamental operation of the algorithm, the implementation of this technique for a large quantity of objects does not proceed as a series of pair-wise comparisons of bounding volumes. Figure 3 provides conceptual diagram of the overall program structure, while Figure 4 outlines the CSieve algorithm.

Figure 3 Overall Program Structure



Time Slicing

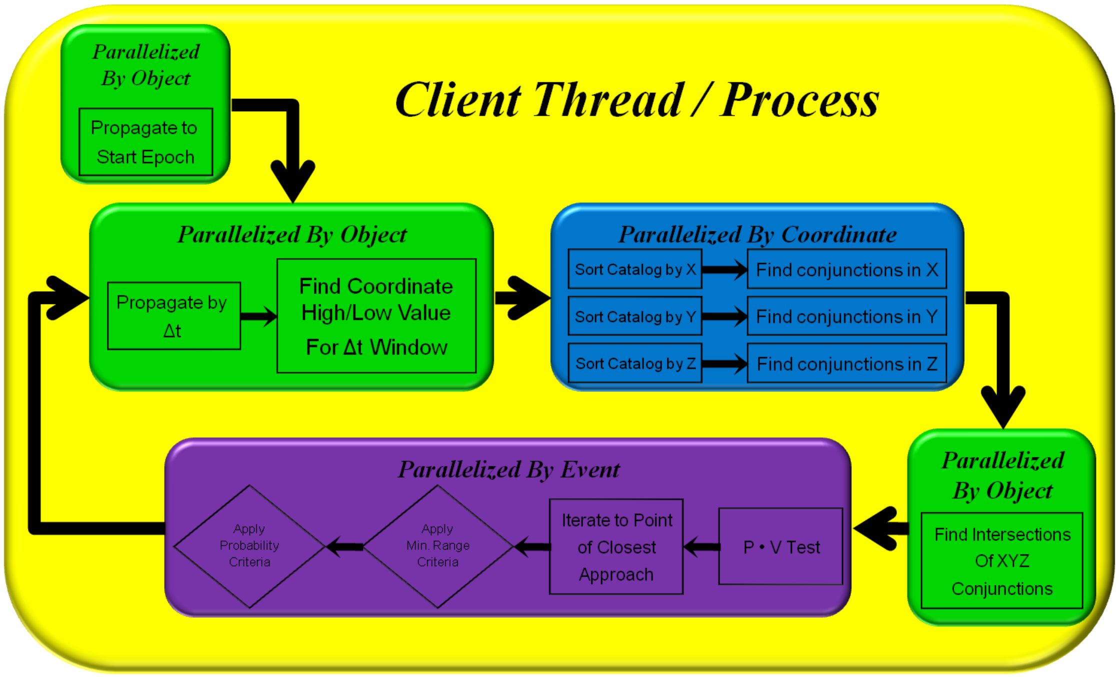
The first division of the problem is in the time domain. Here the overall time span of the analysis is divided into sections. Each section or "slice" is then analyzed by a client implementing the CSieve algorithm. The number of slices the problem is broken into is a function of the computational resources available.

There are two versions of the CSieve program at this time. The original version was developed for cluster computers. In this case, the client is actually a separate computational process operating on an independent processing node of the cluster. Each client has its own copy of the catalog and communicates with the master process via a message passing library.

With the advent of large multi-core processors, a second version of the code was developed to take advantage of large "Single System Image" (SSI) high performance computers. In the SSI version of the code each client is a thread and there is a single copy of the catalog maintained by the master thread. The clients share access to this common catalog using MUTEX (Mutual Exclusion) locking mechanisms to control access to individual satellite objects.

The SSI version of the code is the most actively developed and maintained at this time and is the version referred to for the remainder of this paper.

Figure 4 The CSieve Algorithm



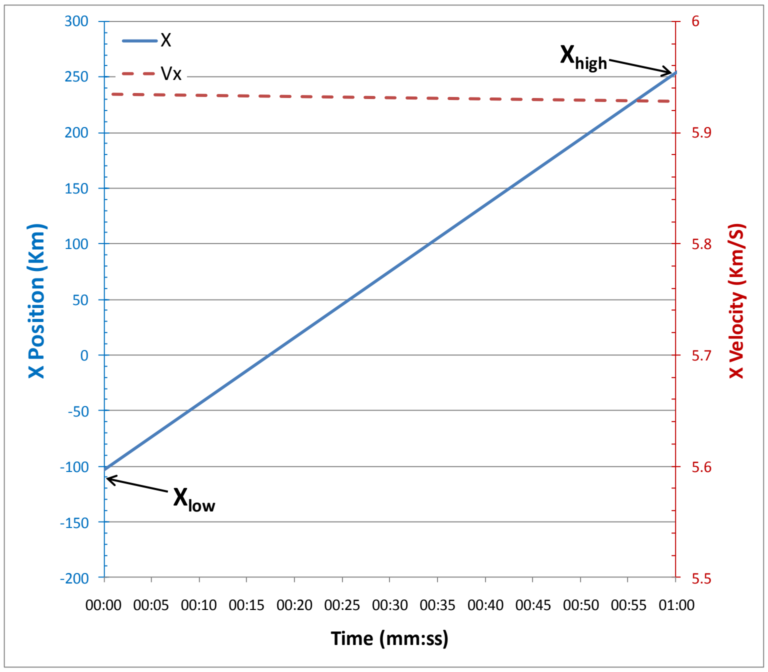
The CSieve Algorithm

Figure 4 outlines the CSieve screening methodology, which will be described below in more detail.

Propagation

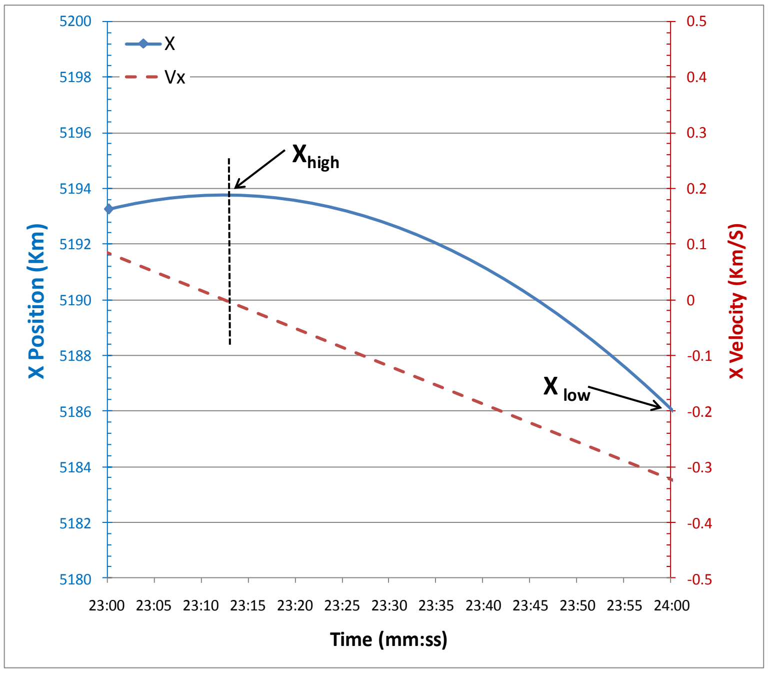
The entire catalog is initially propagated to the starting epoch of the client. The state vector for each object is saved, and the catalog is then propagated again by the analysis time step Δt. This state is also saved. The two saved states are then examined to determine the high and low values of each component of the position vector within the current Δt window. The high and low values will typically be the already computed states at the ends of this interval, as show in Figure 5

Figure 5 Typical Position Component High/Low Values



However, twice per orbit revolution each position component reaches an extreme. This extreme can be easily identified by the change in the sign of the associated velocity component across the analysis window, as shown in Figure 6. In this case an iterative process is used to find the root of the velocity component, and corresponding position extreme.

Figure 6 Position Component High/Low Values at Component Extreme



The extremes of the position components define the bounds of the circumscribing volumes described in Figure 2. This portion of the algorithm is very simply parallelized on a per-object basis.

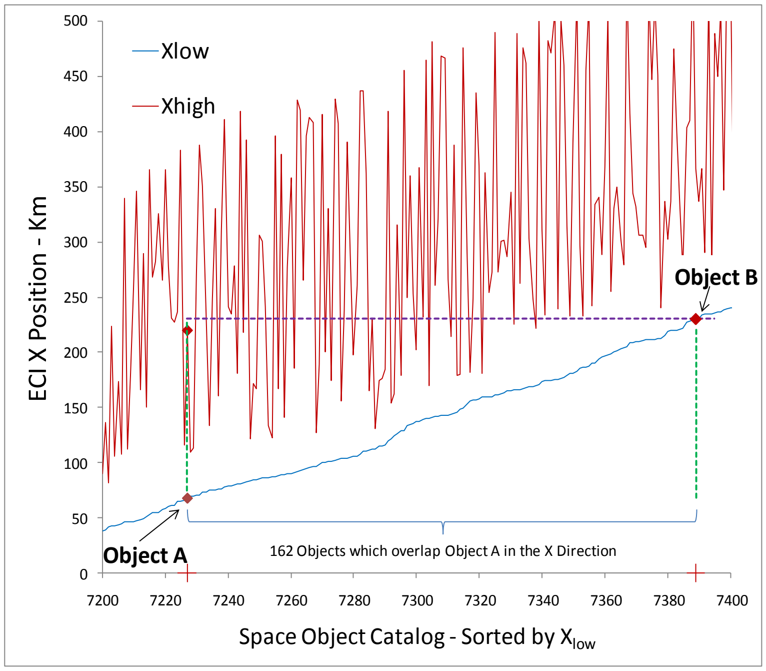
The computational complexity of the propagation operation is *O(N•S)* propagation operations where S is the number of analysis time steps.

Per-Coordinate Conjunctions

The next task is to identify which objects have intersecting circumscribing volumes. In order for the rectangular volumes to intersect, they must overlap in each of the component directions. In fact, the volumes don't need to actually intersect, but they must be within the specified screening distance of each other.

A vector representing the catalog is now sorted according to the X_{low} value for each object. Objects which are close to each other in the *X* direction in space are now close to each other in the vector sorted by X_{low}. In order to find the objects which overlap in the *X* direction, the vector is traversed from low to high. For each object (A) the vector is searched in the forward direction for the first object (B) for which $B.X_{low} > (A.X_{high} + R)$. The objects between A and B are all within the screening threshold *R* of object A in the *X* direction. This is illustrated in Figure 7 below.

Figure 7 Overlapping Volumes in X



The objects which are close to Object A, but to the left of A in Figure 7, are accounted for by the previous examination of the objects to the left of A. This analysis is repeated for the *Y* and *Z* directions so that every object in the catalog has associated lists of objects which are close to it in the *X*, *Y* and *Z* directions.

This algorithm is parallelized by coordinate with a separate thread used to sort and search on each of X, Y and Z.

The computational complexity for the sort portion of this stage of the algorithm is O(S•*N•log(N))*. The search portion is a sequence of binary searches, each of complexity O(log(N)) on a vector where N decreases by one for every iteration.

(4) $$\sum_{i=1}^{N-1} \log(N-i) = \log\left(\prod_{i=1}^{N-1}(N-i)\right) \cong log(N!)$$

So the overall complexity of the search portion is *O(S•log(N!))*. This is actually an upper bound for the complexity of the search portion, because only a fraction of the right side of the list in Figure 7 will need to be searched for every object after the first object. For example, denote the object from Figure 7 which immediately follows object *A* as object $\acute{A}$. If $\acute{A}_{high} < A_{high}$ then only the portion of the vector between object $\acute{A}$ and object *B* must be searched. Similarly, if $\acute{A}_{high} > A_{high}$ then only the portion of the vector between object *B* and the end of the vector must be searched.

Intersection

Each object *A* now has three lists of objects associated with it. Objects which are close to object *A* in the *X* direction are denoted C_x, objects which are close in *Y* are denoted C_y and those close in *Z*, denoted C_z. The set of objects with circumscribing volumes which overlap the circumscribing volume of object *A* is the intersection of these sets.

(5) $$C = C_x \bigcap C_y \bigcap C_z$$

Since the volume of object *A* overlapping the volume of object *B* is identical to the volume of *B* intersecting the volume of *A*, the set of intersecting objects *C* for any object *A* is arbitrarily restricted to contain only objects with catalog numbers higher than that of *A*.

The complexity of the intersection operation is *O(S•N^2)* of integer comparisons.

Refinement

Intersection of circumscribing volumes is a necessary, but not sufficient condition for a conjunction meeting the range threshold. The first step in the refinement process is to determine if there is a minima of $|\vec{P}_r|$ within the current analysis time step. The presence of a minima within the analysis time step is indicated by the signs of the dot products of the relative position and velocity vectors at the beginning and end of the time step (Equations 1, 2).

Care must be taken in the selection of Δt such that there cannot be more than one minima of $|\vec{P}_r|$ within any time step.

If a minima is detected, it can then be found as the root of the dot product of $\vec{P}_r$ and $\vec{V}_r$ (Equation 3) as described in Figure 1. The range of this minima is then compared to the range screening threshold.

At this point a risk assessment filter, such as probability of collision[4] (P_c) can be applied. While presented as an example of a risk assessment filter, a discussion of satellite probability of collision is beyond the scope of this paper.

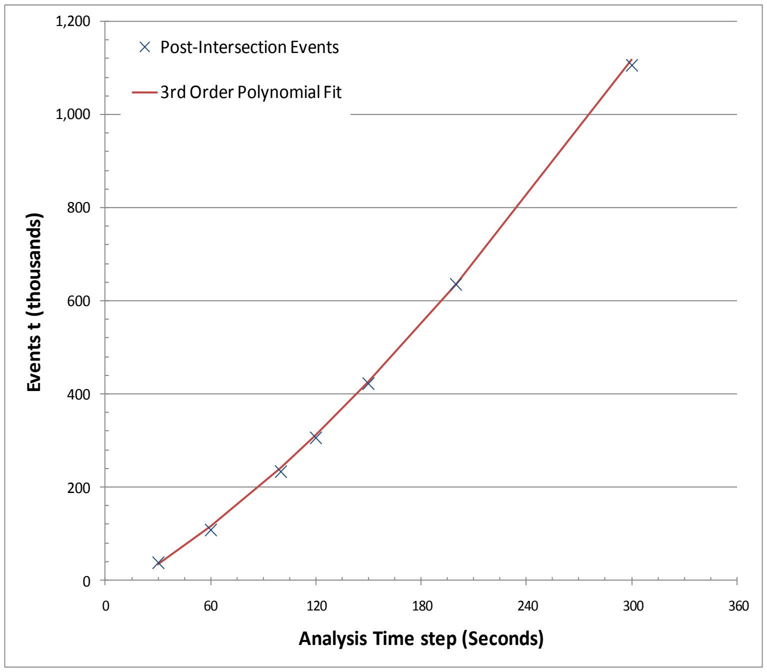
Table 1 below shows the number of events at various stages of the algorithm. The results are from an analysis of 14,476 objects with a range threshold of 10 Km, an analysis time span of 10 minutes and *Δt* of 1 minute. The P_c estimate was obtained with estimated general perturbations (GP) covariance[5] and object sizes derived from public sources or radar cross section (RCS) data.

Table 1 Events by Processing Stage

	Total Events	Avg. per Object, Δt	Stage
C_x (Conjunctions in X)	22,239,291	153.62870	***Intersection***
C_y (Conjunctions in Y)	22,333,387	154.27872	
C_z (Conjunctions in Z)	29,606,712	204.52274	
C $(C_x \cap C_y \cap C_z)$	106,786	0.73768	
$\vec{P}_r \cdot \vec{V}_r$ Sign Test	75,685	0.52283	***Refinement***
Range <= 10 Km	288	0.00199	
Pc >= $1\text{x}10^{-6}$	6	0.00004	

The number of events which enter the Refinement stage (i.e. the output of the Intersection stage) scales with the size of the circumscribing volumes. This volume increases as the cube of the time step Δt. Figure 8 shows the quantity of post-intersection events as a function of Δt for the analysis described above.

Figure 8 Post Intersection Events as a Function of Δt



3. Software Implementation

Language and Platform

The CSieve algorithm described here is implemented in C++ for the Linux operating system. Significant external libraries include the MySQL[*] client library for database access, the Boost[†] C++ libraries for compressed file I/O. The integrated SGP4 propagator is from Vallado[6].

[*] http://www.mysql.com

Program Features

Data and Configuration Input and Output

The CSieve program is tightly integrated with the MySQL relational database system. Most configuration data is stored as records in a database table. Command line input is limited to the database server and login information, run title, number of clients to be executed and the configuration ID number.

CSieve can use TLE (Two Line Element) element set data or ephemeris data for analysis. Input of TLE and ephemeris data is multi-threaded to improve start-up performance. TLE data is retrieved from a database populated with the full public historical TLE record, which is regularly updated from the Air Force Space-Track[‡] website. Analysis can be requested for the current catalog, or for any historical time. For historical analysis the program can load multiple TLEs for the requested analysis period. During the analysis it will select the TLE with the epoch closest to, but prior to the beginning of the current analysis window.

Ephemeris data can be provided in an in-house format or the STK (Satellite Tool Kit) ephemeris format. Ephemeris files compressed with ZIP or GZIP can be read directly. While the ephemeris data itself is not stored in the database, metadata to manage access to the data is. TLE and ephemeris data can be combined in the same analysis.

All output data is written directly to the database. The database design emphasis the traceability of the analysis results. Any result can be simply associated with a particular analysis run, run configuration, computing platform, code version, and input data.

Primary Object Modes

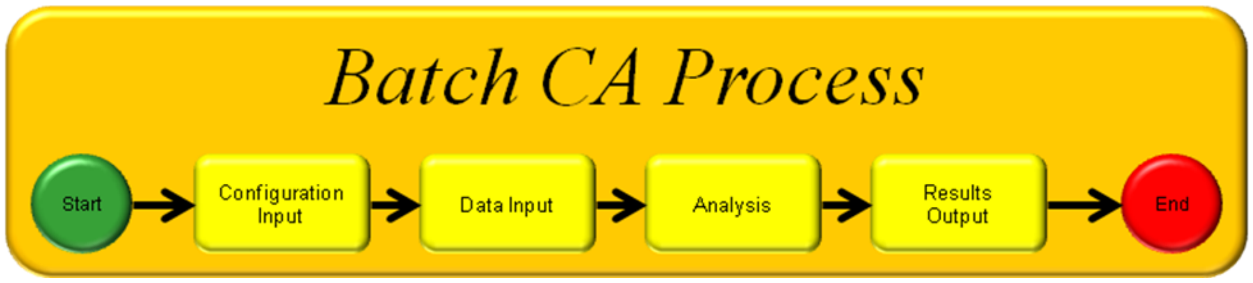
Space objects in a conjunction analysis are generally denoted as “primary” and “secondary” objects. Conjunctions are reported between primary and secondary objects, and reporting of primary vs. primary objects is a configuration option.

The default mode CSieve is to perform an All vs. All analysis (all objects are primary). However N vs. All and N vs. M analysis can be easily configured.

Batch and Continuous Analysis

Conjunction analysis is typically done as a batch process, as described in Figure 9. This model may incur very high startup costs, particularly for ephemeris based analysis. Significant redundancy of analysis may also occur without careful tracking of data updates.

Figure 9 Batch CA Process

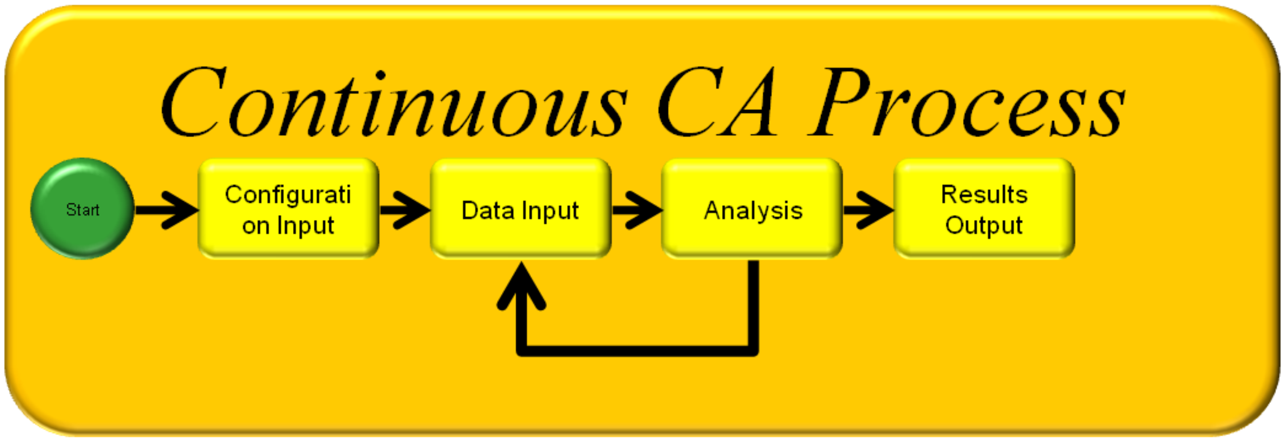


A continuous processing model has been developed for CSieve. In this model the CA process is started once and proceeds through the configuration, data input and analysis phases as described in Figure 9. However, in the

[†] http://www.boost.org
[‡] http://www.space-track.org

continuous processing model, the analysis doesn't exit at this point. A separate thread continues to receive new data as it becomes available and this new data is used to update the internal catalog for immediate analysis. This technique reduces the large startup cost of a full catalog analysis to a one-time event and can return analysis of updated data very quickly. This will be covered in more detail in future publications.

Figure 10 Continuous CA Process



Performance

Completeness

Reference 2 provides a detailed comparison of the accuracy and completeness of six CA tools, including CSieve. In this 7 day, 11K object study, CSieve was one of two tools which found all of the identified conjunctions (one additional tool found all events after the default filter configuration was found to be in error and corrected). The other tool which found all of the events, CAOS-D[7], is based on the CSieve algorithm.

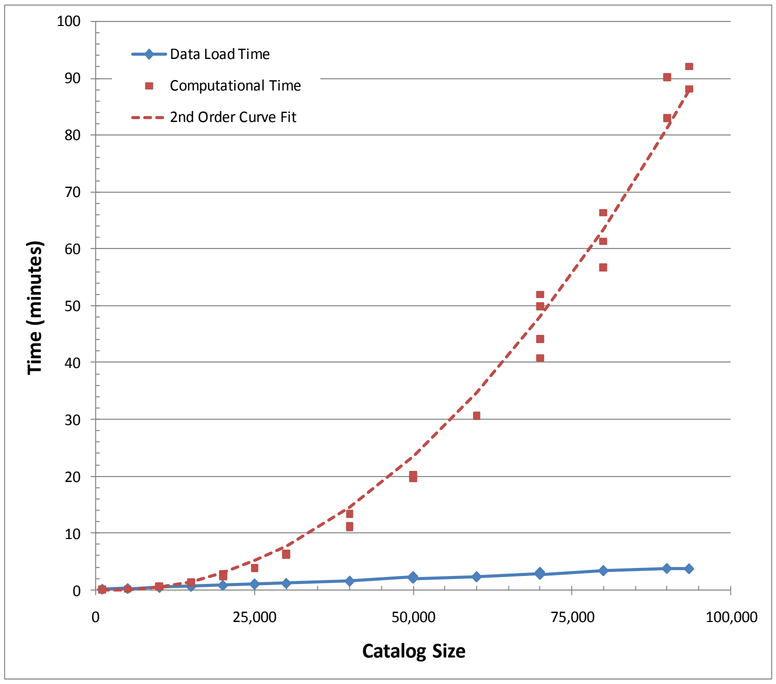
Computational Performance

The following data is based on analysis completed on a 24 core server (4x AMD Opteron Model 8431 CPU, 6 Cores/CPU) with 128GB RAM and running 64 bit CentOS v5.5.

To simulate the computational load of future catalogs, a large TLE data set generated by NASA debris models was used. This data set contains 93,500 objects. As received, all objects in this data set had an epoch of 2030-01-01 00:00:00 and a Mean Anomaly of 0, meaning that all objects were at perigee at the catalog epoch. In an attempt to make this catalog more realistic, the mean anomaly of all objects was reset to a random draw. However, this is not likely to be a realistic catalog in terms of the number of conjunctions produced as a function of the size of the catalog. It is used only as a convenient source of a large catalog for the purpose of evaluating computational performance. Future publications may examine the characteristics of All vs All results sets and the problem of developing data sets to model future large catalogs.

Figure 11 illustrates the performance of the CSieve algorithm as a function of catalog size for a 24 hour analysis. The screening threshold for this analysis was 10 Km. Catalogs of less than the maximum available size were created by randomly selecting subsets of the full catalog. The variation in computation times for catalogs of identical size is primarily due to experimentation with configuration parameters such as Δt, the number of clients and the number of threads per client. The optimum values for these parameters vary significantly with catalog size and the proportion of primary objects in the analysis.

Figure 11 Computation Time for a 24 hour All vs. All Analysis as a Function of Catalog Size



4. Conclusion

The CSieve algorithm demonstrated here puts the All vs. All conjunction analysis screening of very large space object catalogs within reach with modest investments in commodity hardware. The primary challenges at this time remains adequate risk assessment of the events indentified and the development of processes to take advantage of this analysis.

---

[1] Hoots, F. R., Crawford, L. L., Roehrich, R. L., "*An Analytic Method To Determine Future Close Approaches Between Satellites*", AAS/AIAA Paper 1983-333, August 1983

[2] George, E., "*A Comparison of Satellite Conjunction Analysis Screening Tools*", Advanced Maui Optical and Space Surveillance Technologies Conference, 13-16 Sep. 2011, Wailea, Maui, HI

[3] Healy, L. M., "*Close Conjunction Detection on Parallel Computer*", Journal of Guidance, Control, and Dynamics, Vol. 18, No. 4, 1995, pp. 824-829

[4] Alfano, S., "*Review of Conjunction Probability Methods for Short-term Encounters*", Advances in the Astronautical Sciences. Vol. 127, Part 1, pp. 719-746. 2007

[5] Peterson, G.; Gist, R.; Oltrogge, D., "*Covariance Generation for Space Objects Using Public Data*", Proceedings of the 11th Annual AAS/AIAA Space Flight Mechanics Meeting, Santa Barbara, CA; UNITED STATES; 11-15 Feb. 2001. pp. 201-214. 2001

[6] Vallado, D., "Companion Code" for "*Fundamental of Astrodynamics and Applications, 3rd Ed.*", 2007, http://celestrak.com/software/vallado-sw.asp

[7] Abernathy, B, Harvey, S., Surka, D., O'Connor, M., "The CAOS-D Architecture for Conjunction Analysis", *Infotech@Aerospace 2011 Conference*, 29-31 Mar. 2011, St. Louis, MO