A study of the time complexity of an Elitist Genetic Algorithm used for associating optical observations of MEO and GEO Space Debris

M. Zittersteijn ⋆†, A. Vananti⋆, T. Schildknecht⋆, J.C Dolado-Perez⋆⋆ and V. Martinot⋆⋆⋆
⋆Astronomical Institute of the University of Bern (AIUB)
Sidlerstrasse 5, 3012, Bern, Switzerland
⋆⋆Centre National d'Etudes Spatiales (CNES)
18 Avenue Edouard Belin, 31400 Toulouse, France
⋆⋆⋆ Thales Alenia Space (TAS)
26 Avenue Jean François Champollion, 31100 Toulouse, France
†Corresponding author: michiel.zittersteijn@aiub.unibe.ch

Abstract

Currently several thousands of objects are being tracked in the MEO and GEO regions through optical means. The problem faced in this framework is that of Multiple Target Tracking (MTT). In this context both the correct associations among the observations, and the orbits of the objects have to be determined. The complexity of the MTT problem is defined by its dimension S. Where S stands for the number of 'fences' used in the problem, each fence consists of a set of observations that all originate from different targets. For a dimension of $S \geq 3$ the MTT problem becomes NP-hard. As of now no algorithm exists that can solve an NP-hard problem in an optimal manner within a reasonable (polynomial) computation time. However, there are algorithms that can approximate the solution with a realistic computational effort. To this end an Elitist Genetic Algorithm is implemented to approximately solve the $S \geq 3$ MTT problem in an efficient manner. Its complexity is studied and it is found that an approximate solution can be obtained in a polynomial time. With the advent of improved sensors and a heightened interest in the problem of space debris, it is expected that the number of tracked objects will grow by an order of magnitude in the near future. This research aims to provide a method that can treat the correlation and orbit determination problems simultaneously, and is able to efficiently process large data sets with minimal manual intervention.

1. Introduction

Cataloging Space Debris can be put in the more general framework of Multiple Target Tracking. The MTT problem can be summarized as follows. A region contains any number of target objects of which the states are unknown. Starting from a set of S fences (a.k.a. scans/layers, depending on the domain of application) collected by any number of sensors, both the total number of targets and their states have to be estimated. A fence consists of a set of observations that all originate from different targets. The observables depend on the sensor type. In this case only optical sensors are considered, these provide Right Ascension α and Declination δ values. The fields where MTT problems are encountered are numerous; examples are the tracking of targets in a military context, and the tracking of particles resulting from high energy collisions in particle physics. An inherent characteristic of the MTT problem is that a single observation is not enough to estimate the target state. Therefore the problem consists of two interrelated parts, namely data association and state estimation. In the data association part the observations from the different fences have to be associated to the correct targets. The state estimation part then takes these associated groups of observations and estimates the target state. These parts are related to each other since the associations are needed for a state to be estimated. This leads to a search for the permutation that results in the best target state estimates. The number of fences S that are used in the problem correspond to the dimension. For a dimension of $S \geq 3$ the number of possible permutations greatly increases and the problem becomes NP-hard. Despite its challenges, several attempts have been made to solve this problem in an efficient manner. The Multiple Hypothesis Tracking (MHT) algorithm seeks to find the optimum solution to the MTT problem by employing a branch and bound methodology. In order for this algorithm

to have a realistic computation time the MTT problem has to be greatly simplified. Another approach to the problem is to seek an approximate solution that can be obtained in a realistic computation time (preferably in polynomial time). An example of such an algorithm is the Lagrangian relaxation technique [Deb et al., 1997]. The alternative to solving the $S \geq 3$ problem is to solve the $S = 2$ problem. This problem has the favorable computational complexity of $O(n^2)$. Recent work in this area can be found [Fujimoto et al., 2014] [Siminski et al., 2014] [Schumacher et al., 2013]. The drawback of this approach is that with only pairs of observations the information available for the state estimation is minimal. This can in turn lead to wrong association decisions and state estimates of low quality. The severity of these problems depends on the target density in the region of interest. So to solve the MTT problem applied to a densely populated area an efficient way has to be found to search through the possible permutations. In previous work a series of Population-Based metaheuristic algorithms were implemented and tested on simulated data [Zittersteijn et al., 2015]. From this it became clear that the Elitist Genetic Algorithm (EGA) performs as desired, therefore this shall be the only algorithm that is considered in this paper. The EGA is capable of finding a good approximate solution in a consistent manner. Now the question remains whether it can do so in a reasonable (preferably polynomial) computation time. This paper aims at answering that question by investigating the relationship between computation time and number of objects that are being tracked. The paper is organized as follows. In the next section the Optimized Boundary Value Orbit Determination (OBVOD) method is presented, this method is a necessary part of the complete algorithm. In the third part the basic workings of each of the EGA is discussed. The fourth part covers the time complexity study. Finally the paper is closed with the conclusions that can be drawn and the outlook for future research in this area.

2. The Optimized Boundary Value Orbit Determination (OBVOD) method

In the proposed algorithms there is a need for an orbit determination method that can determine an orbit for any set of tracklets larger or equal to two. A tracklet consists of a series of closely spaced observations, typically spaced about 30 seconds apart. A tracklet is normally about 4 - 7 observations in length. This technique of collecting observations provides a major advantage, which is that now also information on the angular rates is available. It is assumed that the series of observations can be approximated with a straight line. By fitting such a line, each tracklet provides us with an estimate of the topocentric angular velocity of the object. These observables at epoch t are denoted as follows:

$$\left(\alpha, \delta, \dot{\alpha}, \dot{\delta}\right)_t \tag{1}$$

Four observables are directly accessible from the observations. However an orbit has at least six orbital parameters that uniquely define it. Therefore an additional two observables are needed. In the method developed by [Siminski et al., 2014] the angular positions of two tracklets were taken. The two additional observables are the ranges towards the first and second tracklet. The derived angular rates are used to quantify the quality of the computed orbit. The range values are unknown. The method by [Siminski et al., 2014] seeks to find the (ρ_1, ρ_2) combination that leads to the best fitting orbit w.r.t. the angular rates.

In the original method described by Siminski et al. [2014] the quantity in Equation 2 has to be minimized.

$$L_{N=2}(k, \bar{p}) = \left(\dot{z} - \hat{\dot{z}}\right)^T \Sigma_{\dot{z}}^{-1} \left(\dot{z} - \hat{\dot{z}}\right) \tag{2}$$

Here the z denotes the angular positions, the dot accent denotes the first derivative w.r.t. the time and the hat accent denotes the computed value. The Σ denotes the covariance matrix. The k stands for the number of orbital revolutions between the first and second epoch. Finally the $\bar{p} = (\rho_1, \rho_2)$ is the hypothesis . The quantity in Equation 2 is also known as the Mahalanobis distance. It was shown that for a given k the topography of this loss function is smooth and contains one unique minimum point. Therefore a gradient descent scheme can be used to search for the optimum solution. The procedure to evaluate the loss function is simple. For a given hypothesis two geocentric position vectors can be derived. This gives the classical Lambert problem, which can be solved to obtain the six orbital parameters. This orbit is subsequently used to compute the angular rates, these values are then used in Equation 2 to evaluate the loss function. This method can treat two tracklets at a time, and give an indication of whether they might be correlated to each other. When the observation errors are Gaussian the Mahalanobis distance is χ^2 distributed, this can be exploited by setting a threshold. If the Mahalanobis distance is below that threshold, a decision is made to correlate those two tracklets.
The algorithm described in this paper aims to approximately solve the $S \geq 3$ MTT problem. Therefore the method needs to be expanded such that it can determine the orbit and Mahalanobis distance for more than two tracklets. This expansion is straightforward. For a set of N number of tracklets, the first and last tracklets are used in the search for the optimum range hypothesis. The estimated and computed angular rate values of the tracklets in between are added to the vectors in Equation 2. Additionally, there is now a group of tracklets that is not involved in the Lambert problem. This

means that there will be a non-zero residual between the observed and computed angular positions for those tracklets. Including those quantities leads to the loss function in Equation 3 for a tracklet set of $N \geq 3$ in size.

$$L_{N\geq 3}(k, \bar{p}) = \left(\dot{z} - \hat{\dot{z}}\right)^T \Sigma_{\dot{z}}^{-1} \left(\dot{z} - \hat{\dot{z}}\right) + (z - \hat{z})^T \Sigma_z^{-1} (z - \hat{z}) \tag{3}$$

An example of the topography of this loss function is shown in Figure 1. It can be seen that for a fixed k there is one unique minimum point, and that the topography is smooth. Therefore a gradient descent algorithm can also be used in this case to minimize the quantity in Equation 3. This quantity is also χ^2 distributed.

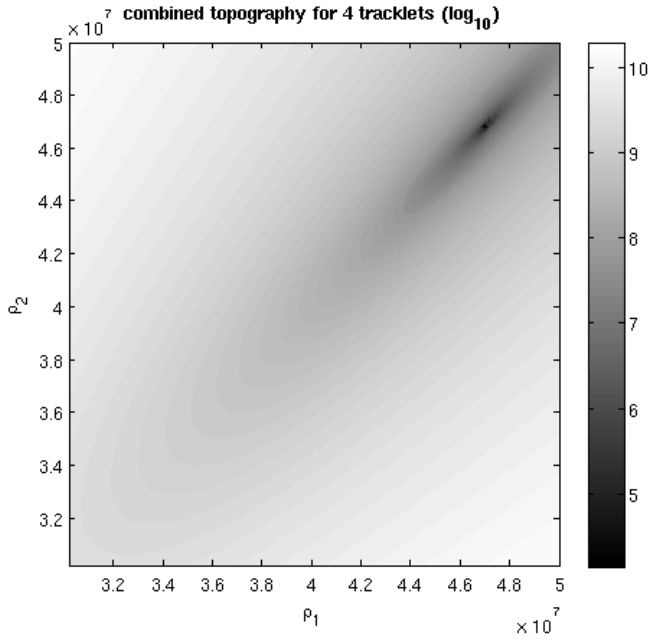

Figure 1: Example of convergence behavior for the EGA. Here it is applied to five objects, with each three observed tracklets at one hour intervals.

3. Genetic Algorithm in Multiple Target Tracking

The GA is inspired by the selection that occurs in nature. A generation of a certain population of individuals is evaluated in order to assign a fitness value to each individual. Some individuals will have a better fitness than others. These individuals then have a higher probability to pass on their information to the next generation. In Figure 2 the flowchart of a GA can be seen. The first step is to create an initial population, this is done at random in order to have a good distribution of individuals throughout the search space. In the second step this random initial population gets evaluated. Each individual gets a fitness assigned to it based on, among others, the quality of the determined orbits. A new population is created by applying the two basic GA operators to the individuals. The first operator is the crossover operator, which works as follows. Two individuals are chosen based on their relative fitness. This sampling is done according to the discrete probability density function that describes the relative fitness of the individuals in the population. In this way the relatively fit individuals have a higher probability of being chosen for crossover. The crossover operator takes the two selected individuals and recombines them in order to make two new k-matrices. In this work the popular uniform crossover operator is used. This version of crossover exchanges the rows of the two parent matrices if a randomly generated number between zero and one is below a certain crossover probability, this has to be done for each row in the matrix. The crossover probability is defined by the user. Crossover is seen as being the exploiting operator in the GA. In order to ensure a certain extent of exploration, the mutation operator is used. The mutation operator is applied to every new individual. Mutation can randomly change an individual, effectively assigning a tracklet to another object. The two operators are applied until a new population is created. This population is evaluated by calculating the fitness of each individual. In the classical GA this new population will replace the entire former population. Therefore the GA does not necessarily improve its solution from one generation to the next; it can also discard the best solution so far and only be able to formulate a solution of worse quality. A simple variation on the standard GA exists which is called the Elitist Genetic Algorithm (EGA). Here the top few percentage of the population is always copied to the next population, this ensures that the information contained within the best individuals is never

lost. This process is repeated until a stopping criterion is met. In this case the stopping criterion is a maximum number of generations. Alternative criteria could be a certain fitness value, or a maximum number of generations where no improvements were found.

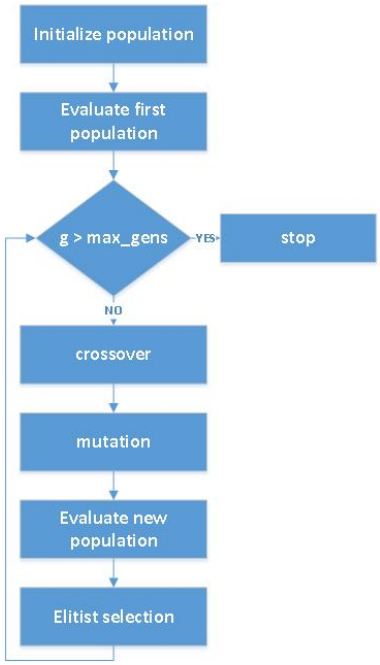

Figure 2: Flowchart of a GA.

For further reading the work of [Goldberg, 1999] is recommended which provides a thorough discussion of GAs. In the works of [Chen and Hong, 1997] and [Turkmen et al., 2006] the GA is applied to an MTT problem.

Definition of an individual and the fitness function

The EGA algorithm works with a population of individuals. Each individual represents a potential solution and is evaluated to determine its so-called fitness. The fitness of an individual is a measure of the quality of that solution. In an EGA there are two important design choices. The first choice is how to define an individual. This is an important choice since the individual has to be able to accurately represent any valid solution in the search space. In this work an individual represents only the correlations between the tracklets. In Equation 4 the general notation for a k-matrix can be seen. The k-matrix notation was introduced in the work of [Schneider, 2012], where the most likely k-matrix was sought through Markov Chain Monte Carlo computations.

$$K = \begin{pmatrix} 1 & 0 & \dots & 0 \\ k_{2,1} & k_{2,2} & \dots & 0 \\ \dots & \dots & \ddots & 0 \\ k_{i,1} & k_{i,2} & \dots & k_{i,j} \end{pmatrix} \tag{4}$$

In the k-matrix any entry $k_{i,j}$ can only have a value of 1 or 0. If $k_{i,j} = 1$ it signifies that the tracklet in row i is correlated to the object in column j. The k-matrix is defined in such a way that the first tracklet is always correlated to the first object. Following this logic the k-matrix becomes a lower triangular matrix.
The second design element is the fitness function. This function is used to assign a fitness value to each individual. The user is free to choose the fitness function as wanted, as long as it fulfils two requirements.

- The individual with the best fitness value should be the optimal solution.
- An improvement in a solution should lead to an improvement in fitness value.

In the fitness function also the external probabilities have to be taken into account. These external probabilities account for the possibilities that detections are missed and that some of the measurements could be spurious. When quantifying the total probability that a certain group of tracklets belongs together the expression in Equation 5 is found.

$$P_{N\geq 2} = \mathcal{N}\left(\dot{z}, \hat{\dot{z}}, \Sigma_{\dot{z}}\right) \mathcal{N}\left(z, \hat{z}, \Sigma_z\right) (1 - P_d)^{L-N} P_d^N \left(1 - P_f\right)^N \tag{5}$$

In Equation 5 the P_d stands for the detection probability, P_f stands for the false alarm probability, the L stands for the total number of fences considered, and N stands for the number of tracklets used in the orbit determination. When we take the negative log-likelihood of the above expression, Equation 6 is obtained.

$$L_{N\geq 2} = -\ln\left(\frac{1}{\sqrt{(2\pi)^k|\Sigma_{\dot{z}}|}}\right) - \ln\left(\frac{1}{\sqrt{(2\pi)^k|\Sigma_z|}}\right) + \frac{1}{2}\left(\dot{z} - \hat{\dot{z}}\right)^T \Sigma_{\dot{z}}^{-1}\left(\dot{z} - \hat{\dot{z}}\right) + \frac{1}{2}\left(z - \hat{z}\right)^T \Sigma_z^{-1}\left(z - \hat{z}\right) - (L - N)\ln\left(1 - P_d\right) - N\ln(P_d) - N\ln(1 - P_f) \tag{6}$$

The k in Equation 6 stands for the dimension of the vector (either z or $\dot{z}$). In the equation also the two Mahalanobis distances can be found. The sum of these two distances is what is minimized in the OBVOD method. There is a special case of the fitness function. This is when only one tracklet is considered. For a single tracklet no orbit can be determined (and thus no Mahalanobis distances). Therefore we can only quantify its probability of being correct by using the detection and false alarm probabilities. It can be said that a single tracklet belongs to an object that has been missed in all the other fences, or it is a false alarm. These two events are assumed to be disjoint from each other, therefore their probabilities can be added (and not multiplied) in order to determine the combined probability. Taking the negative log-likelihood gives the expression in Equation 7.

$$L_{N=1} = -\ln\left((1 - P_d)^{L-1} P_d + P_f\right) \tag{7}$$

Equation 7 is currently used in the algorithm. It seems to fulfill the requirements of the fitness function (which are listed at the start of the section). However the definition of the fitness function for is still a point of discussion. Alternative definitions could involve setting threshold values (based on e.g. the distributions), or an empirical formulation.

4. Time complexity analysis

As mentioned in the introduction, the challenge is to find a good approximate solution in a reasonable computation time. It has already been found that the EGA is capable of finding a good approximate solution, the question that needs to be answered in this section is whether it can do so in a realistic computation time.

The EGA is a stochastic method. Therefore it is difficult if not impossible to say anything about its behavior in a deterministic way. In cases where it is possible to deduce a characteristic, its use is questionable. An example is that the upper bound on the convergence behavior for a regular GA is the same as that of an algorithm that generates solutions purely at random [Oliveto et al., 2007]. In the work of [Oliveto et al., 2007] an overview is given of the recent investigations in the time complexity of Evolutionary Algorithms (a close relative to GAs). One of the conclusions is that the function to be optimized has to be considered when analyzing the convergence behavior.

With this in mind the following experiment was devised. The EGA is applied to problems of different sizes. As it is a stochastic algorithm, it is applied 100 times and the average performance is studied. The problem is said to change size when the number of tracked objects increases or decreases. In previous work it was seen that the proximity of the targets to one another plays a role in the convergence. When the targets are closely space to each other it is more difficult for the algorithm to distinguish between correct and erroneous solutions, this slows down the convergence. In order to keep the difficulty of the problem the same when increasing the number of objects they are spaced at one degree increments in inclination, but are otherwise in identical orbits. They are geosynchronous objects, a series of seven closely space observations (a tracklet) is observed every hour for a total of three tracklets per object.

We are dealing with a combinatorial NP-hard problem. Any problem can be classified as being either a problem with P or NP complexity. The P stands for Polynomial, which means that it can be solved in a polynomial time. If we say that the problem size is denoted by n then the computation time will be something like e.g. n^2 or n^3 (or any other order). The NP stands for Nondeterministic Polynomial time. This means that the computation time is not described by a polynomial but for instance could be something like 2^n. Obviously the computation time of an NP problem will quickly become astronomical (e.g. $n^2 = 10^4$, $2^n = 1.27x10^{30}$ for $n = 100$). So when we talk about a realistic computation time, a polynomial computation time is meant. In Figure 3 an example of the convergence behavior can be seen, in this case with five objects. This is the average behavior over 100 runs. Clearly, the algorithm quickly converges to a reasonable level after which it shows an asymptotic behavior until it reaches the optimum value. The settings of the EGA are given in Table 1 , they are kept constant over all problems.

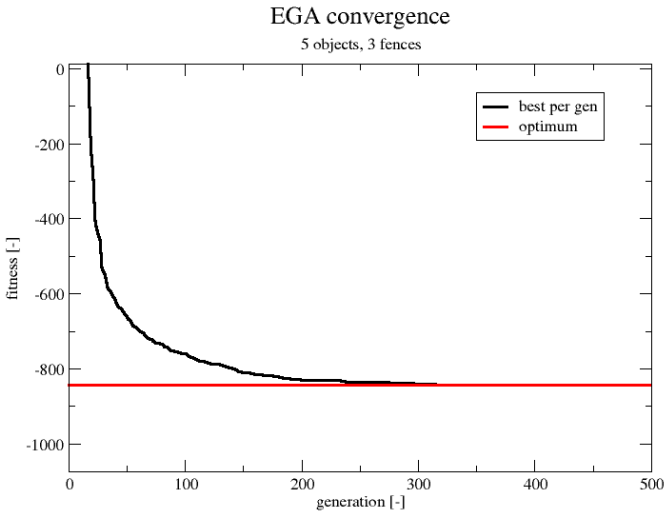

Figure 3: Example of convergence behavior for the EGA. Here it is applied to five objects, with each three observed tracklets at one hour intervals.

Table 1: EGA settings

Population size	$2N$
Crossover probability	0.5
Mutation probability	$1/N$
Percentage copied to next generation	10

Figure 4 shows the average k-matrix at different stages of the computation. Four matrices are shown, at 80%, 90%, 95% of the optimum value, and the true k-matrix. It should be noted that the computational burden is concentrated at the beginning of the run. Here the algorithm needs to perform numerous orbit determinations, the results of which are stored in a look up table. Therefore the computation time needed per generation quickly becomes relatively low. The improvement in the solution from 80% to 95% is therefore relatively cheap from a computational standpoint. It can be said that the required quality of the solution will not dramatically influence the computational complexity. It will not influence the model (e.g. polynomial of a certain degree), but will influence the estimated parameters of that model.

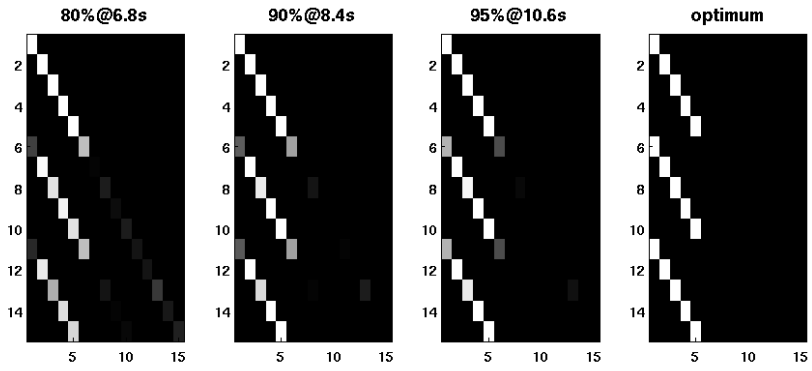

Figure 4: The average k-matrix found at different stages of the search. Both the percentage of the optimum fitness as well as the time at which the solution is found are given.

In Figure 4 it is seen that the algorithm tends to go to a local optimum, where part of object one is identified as being object six. As the number of generations increases the algorithm is able to find this mistake and correct it. At the 95% stage the average k-matrix is correct in the majority of the cases, and when it is wrong it does not make

Table 2: Goodness of fit values for both models.

-	polynomial	exponential
Sum of Squares due to Error (SSE)	4.44	12.52
R-square	0.999	0.998
adjusted R-square	0.999	0.997
RMSE	0.70	1.18

a wrong association. It only mistakenly identifies tracklets six and 11 as being a different object. If a local search method had been used after this stage, this mistake would have been easily identified and corrected. In Figure 5 the absolute computation time in seconds is plotted against the number of objects that are being tracked. These tests were performed on a machine with a 3.16 GHz CPU. It is the time needed to have a solution at 90% of the optimum fitness value. Two lines are fitted to these points, a polynomial and an exponential function. These models were fitted in a least squares sense with the MATLAB curve fitting toolbox. The goodness of fit values are given in Table 2, from this table it becomes apparent that the polynomial model better describes the data points. The SSE is the sum of squares of the differences between the model and the actual data points. The R-squared statistic is the square of the correlation between the predicted and actual response values. When the R-squared value is corrected for the number of degrees of freedom the adjusted R-squared value is found. Finally the RMSE is an estimate of the standard deviation of the data, it can be found by taking the square root of the SSE divided by the number of degrees of freedom. The determined parameters for the exponential function are $a = 3.00$ and $b = 0.29$ and for the polynomial they are $a = 0.11$ and $b = 2.69$.

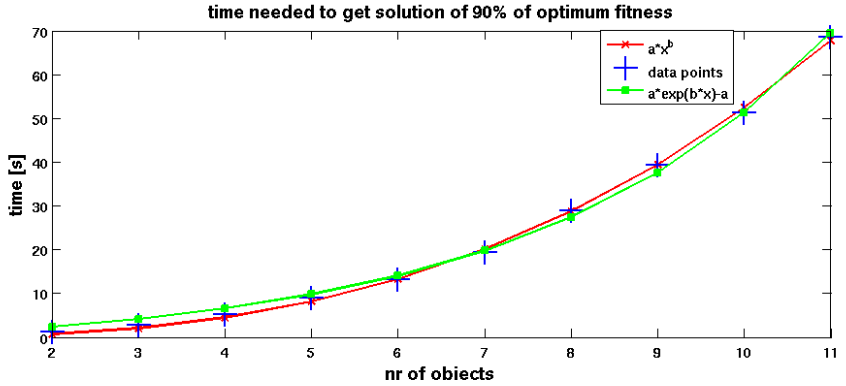

Figure 5: Computation time needed to reach an approximate solution of a certain quality (90% of optimum) versus the number of tracked objects. Two lines are fitted, a second degree polynomial and an exponential function.

This polynomial behavior is not fully understood yet. A possible explanation could be that since both the population size and the number of orbits to be determined per k-matrix increase linearly with the number of tracklets in the data set, it leads to a $O(n^2)$ cost. Given that the number of generations (the application of the crossover and mutation operators) is relatively cheap, the total cost remains in the order of $O(n^2)$. Further investigation will have to point out what the exact reason for this behavior is.

5. Conclusions

The goal of this work was to determine the computational complexity of the EGA algorithm applied to the three dimensional MTT problem. An experiment was devised where the EGA was applied to problems of different sizes (different numbers of tracked objects), and the computation time needed to arrive to a solution of a certain quality was observed. Two models were tested to see which one best fits these data points, a polynomial model and an exponential model. Both models were fitted in a least squares sense. By considering the goodness of fit values that result from this (the SSE, R-square, adjusted R-square and the RMSE), it is concluded that the polynomial model is the better fitting model. The implications of this are significant. It means that indeed, a good approximate result can be found in a polynomial time.
Still, the absolute computation time could be a problem. According to the fitted polynomial curve, for a problem with 300 objects the computation time would be about 139 hours (for comparison, the exponential function predicts $4.64x10^{34}$ hours). One way to easily reduce that computation time is to distribute the evaluation of the individuals over

different CPUs. When the same settings of the current EGA implementation are used there would be 1800 individuals in the population (2N, where N is the number of tracklets, we assume that there are three tracklets per object). These 1800 individuals could be spread out over 1800 cores. Besides this linear increase in computation speed gained by parallel computing the EGA could improve its performance even more by employing an Island-Based structure [Calegari et al., 1997]. Here several EGAs compete with each other independently (on different workstations), which allows them to explore different parts of the search space. Information between these 'islands' is exchanged by sometimes allowing an individual to 'migrate'. In the work of [Calegari et al., 1997] it is shown that this can bring significant advantages.
There are still many areas to be investigated regarding this application of the EGA. Future research will focus on reducing the search space for the EGA, this could bring a significant reduction in necessary computation time. Once an acceptable computation time for larger problem (e.g. 1000 tracklets) is reached the algorithm can be applied to real data sets. An intermediate step is needed where the influence of non-Keplerian motion in the simulated tracklets is investigated.

Acknowledgment

This work has been conducted in the framework of a PhD thesis which is funded by the CNES, TAS and the AIUB.

References

P. Calegari, F. Guidec, P. Kuonen, and D. Kobler. Parallel island-based genetic algorithm for radio network design. *Journal of Parallel and Distributed Computing*, 47:86–90, 1997.

G. Chen and L. Hong. A genetic algorithm based multi-dimensional data association algorithm for multi-sensor-multi-target tracking. *Mathematical and computer modeling*, 26, 1997.

S. Deb, M. Yeddanapudi, K. Pattipati, and Y. Bar-Shalom. A generalized s-d assignment algorithm for multisensor-multitarget state estimation. *IEEE Transactions on Aerospace and Electronic Systems*, 33(2), april 1997.

K. Fujimoto, D. J. Scheeres, J. Herzog, and T. Schildknecht. Association of optical tracklets from a geosynchronous belt survey via the direct bayesian admissible region approach. *Advances in Space Research*, 53(2), 2014.

D. E. Goldberg. *Genetic Algorithms in search, optimization and machine learning*. Addison-Wesley, 1999.

P. Oliveto, J. He, and X. Yao. Time complexity of evolutionary algorithms for combinatorial optimization: a decade of results. *International Journal of Automation and Computing*, 4(3):281–293, 2007.

M. D. Schneider. Bayesian linking of geosynchronous orbital debris tracks as seen by the large synoptic survey telescopes. *Advances in Space Research*, 49(4), 2012.

P. Schumacher, M. Wilkins, and C. Roscoe. Parallel alrgorithm for track initiation for optical space surveillance. *Proc. '6th European conference on Space Debris'*, April 2013.

J.A. Siminski, O. Montenbruck, H. Fiedler, and T. Schildknecht. Short-arc tracklet association for geostationary objects. *Advances in Space Research*, 53(8), April 2014.

I. Turkmen, K. Guney, and D. Karaboga. Genetic tracker with neural network for single and multiple target tracking. *Neurocomputing*, 69, 2006.

M. Zittersteijn, A. Vananti, T. Schildknecht, J. C. Dolado-Perez, and V. Martinot. Associating optical measurements and estimating orbits of geocentric objects through population-based meta-heuristic methods. In *International Astronautical Conference*, 2015.