Associating optical measurements of MEO and GEO objects using Population-Based Meta-Heuristic methods

M. Zittersteijn[a,*], A. Vananti[a], T. Schildknecht[a], J. C. Dolado Perez[b], V. Martinot[c]

[a] *Astronomical Institute of the University of Bern, Sidlerstrasse 5, 3012 Bern, Switzerland*
[b] *Centre National d'Etudes Spatiales, 18 Avenue Edouard Belin, 31400 Toulouse, France*
[c] *Thales Alenia Space France, 26 Avenue Jean François Champollion, 31100 Toulouse, France*

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). The MTT problem quickly becomes an NP-hard combinatorial optimization problem. This means that the effort required to solve the MTT problem increases exponentially with the number of tracked objects. In an attempt to find an approximate solution of sufficient quality, several Population-Based Meta-Heuristic (PBMH) algorithms are implemented and tested on simulated optical measurements. These first results show that one of the tested algorithms, namely the Elitist Genetic Algorithm (EGA), consistently displays the desired behavior of finding good approximate solutions before reaching the optimum. The results further suggest that the algorithm possesses a polynomial time complexity, as the computation times are consistent with a polynomial model. 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 association and orbit determination problems simultaneously, and is able to efficiently process large data sets with minimal manual intervention.

*Corresponding author, Tel: +4131 631 8819
Email address: michiel.zittersteijn@unibe.aiub.ch (M. Zittersteijn)

Keywords: Multiple Target Tracking, Data Association, Orbit Determination, Combinatorial Optimization

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 scans, collected by any number of sensors, both the total number of targets and their states have to be estimated. A scan is defined as a set of observations that all originate from different targets. For a mathematical formulation of the MTT problem, the reader is referred to (Deb et al., 1997). The fields where MTT problems are encountered are numerous, examples are the tracking of targets in a military context (Rakdham, 2009), and the tracking of particles resulting from high energy collisions in particle physics (Pusztaszeri et al., 1996).

The MTT problem also takes the occurrence of both false alarms (sporadic measurements) and missed detections into account. The problem consists of two interrelated parts, namely data association and state estimation. In the data association part the observations from the different scans have to be associated to the correct targets. The state estimation part then takes these associated groups of observations and estimates the target state. This leads to a search for the permutation that results in the target state estimates that best approximate the measurements, according to a certain metric (e.g. the Mean Squared Error). The number of scans S that are used in the problem correspond to its dimension. For a dimension of $S \geq 3$ the number of possible permutations greatly increases and the problem becomes NP-hard (Poore and Gadaleta, 2006). For instance, in the case where $S = 2$ with two observations per scan, there will be a total of seven possible permutations. However for the $S = 3$ case with two observations per scan, there will be 87 of these permutations (Aristoff et al., 2013). 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 e.g. n^2 (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 2^n. The computation time of an NP problem will quickly

become unrealistically large. Despite its challenges, several attempts have been made to solve the $S \geq 3$ MTT problem in an efficient manner. The Multiple Hypothesis Tracking (MHT) algorithm (Blackman, 2004; Aristoff et al., 2013) 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. An algorithm has a realistic computation time when it has a Polynomial time complexity. Examples of algorithms that seek an approximate solution are the Lagrangian relaxation technique (Deb et al., 1997), and the GRASP algorithm (Robertson, 2001). The alternative to solving the $S \geq 3$ problem is to solve the $S = 2$ problem. This problem has the favorable computational complexity of $\mathcal{O}(n^2)$. Recent work in this area can be found in (Siminski et al., 2014; Fujimoto et al., 2014; Schumacher et al., 2013). These methods evaluate pairs of observations, and take a definitive decision on whether to associate the two observations or not. This can lead to wrong associations, since no information besides those two observations are taken into account (an $S \geq 3$ MTT approach would consider more observations, as well as the possibilities of false alarms and missed detections). The severity of this problem depends on the target density in the region of interest. So to solve the MTT problem applied to a densely populated area (e.g. satellite clusters, break-up events) an efficient way has to be found to search through the possible permutations. The MTT problem represents a so-called track based approach, where the state of each object is explicitly resolved. Another possible approach to cataloging space debris is by using a population based approach. Such an approach aims to statistically represent the debris population. An example of such a method is the AEGISS-FISST method (DeMars et al., 2015).

The goal of this paper is to identify an algorithm that can overcome the pitfalls of an $S = 2$ algorithm, without possessing an unfavorable computational complexity. To this end a series of Population-Based Meta-Heuristic (PBMH) algorithms are proposed. These algorithms are a popular choice when faced with a (combinatorial) NP-Hard optimization problem (Reeves, 1995; Beasley and Chu, 1995). The Genetic Algorithm (GA) is a popular PBMH algorithm which was conceived in the 1960s (Goldberg, 1999). Since then the GA has been successfully applied to many different (combinatorial) NP-hard problems spread out over many different domains. A part of the appeal of the GA is its versatility, i.e. its structure does not depend on the

problem. Since the coming of the GA there have been other developments in the field of PBMH algorithms. Other algorithms with a high potential are the Population Based Incremental Learning (PBIL) (Baluja, 1994) and the Differential Evolution (DE) (Onwubolu, 2009) algorithms. Both of these algorithms have shown some promise when applied to combinatorial optimization problems.

Only optical sensors are considered in this work, these provide Right Ascension α and Declination δ values. A few important terms that are used throughout the paper are defined as follows:

- Observation: a Right Ascension and Declination pair at epoch t: $(\alpha, \delta)_t$.
- Tracklet: a series of seven observations at 30 second intervals, already determined to stem from the same object.
- Fence: a set of tracklets that all belong to different objects.

In this work also a method of tracklet association is used that is heavily based on the previous work done by Siminski et al. (2014). It should be seen as an extension to that method. The paper is organized as follows. In the next section the Optimized Boundary Value Initial Orbit Determination (OBVIOD) method is presented. In the third part the basic principles of each of the PBMH algorithms are discussed. The fourth part covers the application of the resulting algorithm to three test cases. In the fifth part attention is given to the computational complexity of the algorithms. 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 Initial Orbit Determination (OBVIOD) method

The Boundary Value method developed by (Siminski et al., 2014) is used as a foundation for this algorithm. The method employs a loss function with a favorable topography. The association uses the Mahalanobis distance as a criterion, this is a χ^2 distributed statistically significant value. Besides this, it gives an initial orbit which can be refined further with e.g. the Least Squares method. The algorithm exclusively uses tracklets. It is assumed that the tracklets are formed correctly from the individual observations, we do not concern ourselves with the tracklet formation process such as the one

detailed in (Früh and Schildknecht, 2012). Using tracklets instead of individual observations provides an advantage, which is that now also information on the angular rates is available. 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. This gives the attributable (Tommei et al., 2007) at epoch t which is denoted as follows:

$$\bar{\theta}_t = \left(\alpha, \dot{\alpha}, \delta, \dot{\delta}\right)_t^T \tag{1}$$

In this method no 'raw' observations are used, only those that are derived from the linear fit. Henceforth the angular rate and position that have been derived from the line fit to the individual observations are referred to as the 'attributed' angular rate and position. The angular rate and position that follow from a computed orbit are referred to as the 'computed' values. The four attributed values are accessible from the observations. However an orbit has at least six orbital parameters that uniquely define it. Therefore two additional values are needed. By considering two tracklets, two attributables (eight values) are available. The two angular positions (four values) together with the ranges at the epochs of the first and second tracklet give the six necessary values. The angular rates of the attributables are used to quantify the quality of the computed orbit. With any range hypothesis $\bar{p} = (\rho_1, \rho_2)$ and two angular positions, two geocentric position vectors can be derived. These vectors together with the time of flight between the two epochs define the Lambert problem (Battin, 1999). The Lambert problem can be solved to provide a Keplerian orbit that intersects these two geocentric positions with the given time of flight. The Lambert problem has two sets of solutions, a short way and a long way solution. Here it is assumed that the object travels in the direction of the short way solution. This is true for the test cases that are presented in this work. The method seeks to find the optimum range hypothesis $\bar{p}^*$ and number of revolutions m^* that lead to the best fitting orbit according to the loss function in Equation 2.

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

Here the $\dot{\bar{z}} = \left(\dot{\alpha}_1, \dot{\delta}_1, \dot{\alpha}_2, \dot{\delta}_2\right)$ denotes the attributed angular rates, and $\dot{\hat{\bar{z}}} = \left(\dot{\hat{\alpha}}_1, \dot{\hat{\delta}}_1, \dot{\hat{\alpha}}_2, \dot{\hat{\delta}}_2\right)$ denotes the computed values following from the Lambert problem solution. The $\Sigma_{\dot{\bar{z}}}(\mathrm{m}, \bar{p})$ denotes the covariance matrix which

is the summation of two parts. The first part is the $C_{\dot{\bar{z}}}$ matrix which represents the covariance on the attributed angular rates. The second part is the $C_{\dot{\hat{\bar{z}}}}(\mathrm{m}, \bar{p})$ which is the covariance of the computed angular rates. This covariance has to be propagated from the covariance on the individual measurements. The m stands for the number of orbital revolutions between the first and second epoch. The quantity in Equation 2 is also known as the Mahalanobis distance. This method can treat two tracklets at a time, and give an indication of whether they might be associated to each other. Given the assumptions (Keplerian motion and a Gaussian error), the minimized Mahalanobis distance $L_{N=2}(\mathrm{m}^*, \bar{p}^*)$ is χ^2 distributed. By exploiting this fact a threshold can be set. If the Mahalanobis distance is below that threshold, those two tracklets are associated to each other. The Mahalanobis distance is χ^2 distributed when the mean of the random variable is zero. Therefore $\left(\dot{\bar{z}} - \dot{\hat{\bar{z}}}(\mathrm{m}, \bar{p})\right) \sim \mathcal{N}\left(\left(\dot{\bar{z}} - \dot{\hat{\bar{z}}}(\mathrm{m}, \bar{p})\right), (\bar{\mu}_{\dot{\bar{z}}} - \bar{\mu}_{\dot{\hat{\bar{z}}}} = 0), \Sigma_{\dot{\bar{z}}}\right)$ should be the case. Here $\mathcal{N}(\bar{x}, \bar{\mu}_x, \Sigma)$ is the multivariate normal distribution as given in Equation 3.

$$\mathcal{N}(\bar{x}, \bar{\mu}_x, \Sigma_{\bar{x}}) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_{\bar{x}}|}} \mathrm{e}^{-\frac{1}{2}(\bar{x}-\bar{\mu}_x)^T \Sigma_{\bar{x}}^{-1} (\bar{x}-\bar{\mu}_x)} \tag{3}$$

Where $\bar{x}$ is the vector containing the random variables, $\bar{\mu}_x$ is the vector containing the mean values, $\Sigma_{\bar{x}}$ is the covariance matrix and n is the dimension of the vector $\bar{x}$. As long as the tracked objects follow a Keplerian motion this is true. When the objects are subject to perturbations then $\bar{\mu}_{\dot{\bar{z}}} - \bar{\mu}_{\dot{\hat{\bar{z}}}} \neq 0$, and the χ^2 distribution will become a non-central χ^2 distribution.

The MTT algorithm described in this paper aims to approximately solve the $S \geq 3$ MTT problem. Therefore the previously described method needs to be generalized such that it can determine the orbit and Mahalanobis distance for more than two tracklets. The decision to extend the method developed by (Siminski et al., 2014), in favor of using an existing method such as the Gauss method for three observations or a Least Squares approach for more than three observations, is to provide consistency. The final goal is to be able to compare orbits that differ in the number of tracklets N. In order to do this, a consistent quantity has to be used, the extension of the (Siminski et al., 2014) method provides this. Any tracklet has the attributable given in Equation 1. Any set of N tracklets has a collection of attributables, this collection is denoted as $\bar{\Theta} = \left(\bar{\theta}_0, \bar{\theta}_1, \bar{\theta}_i, \ldots, \bar{\theta}_N\right)^T$. With this vector, Equation

2 can be generalized to account for $N \geq 2$ tracklets, resulting in Equation 4.

$$L_{N\geq 2}(\mathrm{m}, \bar{p}) = \left(\bar{\Theta} - \hat{\bar{\Theta}}(\mathrm{m}, \bar{p})\right)^T \Sigma_{\bar{\Theta}}^{-1}(\mathrm{m}, \bar{p}) \left(\bar{\Theta} - \hat{\bar{\Theta}}(\mathrm{m}, \bar{p})\right) \tag{4}$$

Here $\hat{\bar{\Theta}}$ denotes the vector of the computed values. The $\Sigma_{\bar{\Theta}}(\mathrm{m}, \bar{p})$ is defined as $\Sigma_{\bar{\Theta}} = C_{\bar{\Theta}} + C_{\hat{\bar{\Theta}}}(\mathrm{m}, \bar{p})$ where $C_{\bar{\Theta}}$ is the covariance matrix of the attributed values and $C_{\hat{\bar{\Theta}}}(\mathrm{m}, \bar{p})$ is the covariance matrix of the computed values. Now that $N \geq 3$ is possible, there is a choice in which pair of tracklets to use to define the Lambert problem. From the point of view of the Initial Orbit Determination (IOD) it is favorable to cover a maximum portion of the arc in order to improve the accuracy of the solution. Therefore, it is chosen to always use the $(1, N)$ combination.

An example of the loss function topography is shown in Figure 1. Here, three tracklets are considered. The individual contributions of the angular positions and angular rates are shown. The summation of these two parts would give the total loss function topography. In both cases there is one minimum point within the search space, and the topography of both parts are smooth. Therefore a gradient descent algorithm can be used to minimize the quantity in Equation 4. In this work the Broyden-Fletcher-Goldfarb-Shanno (BFGS) (Press et al., 2007) algorithm is used for this purpose. Generally, there is one unique minimum point in the loss function topography. However, this has not been proven in a rigorous, mathematical way. Therefore caution is advised when using this method, as the possibility of having multiple local minima points is not excluded.

The minimized loss function values are χ^2 distributed. The Degree of Freedom f of the χ^2 distribution depends on the number of tracklets N that are used. Each tracklet provides four attributed values, and there are two values being estimated, namely ρ_1 and ρ_N. Furthermore the angular positions of the first and last tracklet $(\alpha_1, \delta_1, \alpha_N, \delta_N)$, will be exactly intersected because they are used to define the Lambert problem. Therefore the degree of freedom for N tracklets is determined by $f(N) = 4N - 6$.

It can be opted to constrain the (ρ_1, ρ_N) search space. This is done by applying the so-called Admissible Region (Milani et al., 2004, 2011; Tommei et al., 2007; Schumacher et al., 2013). This method allows for the restriction of the search space by placing constraints on e.g. the semi-major axis. The constraints only use relationships that do not require much additional computation time.

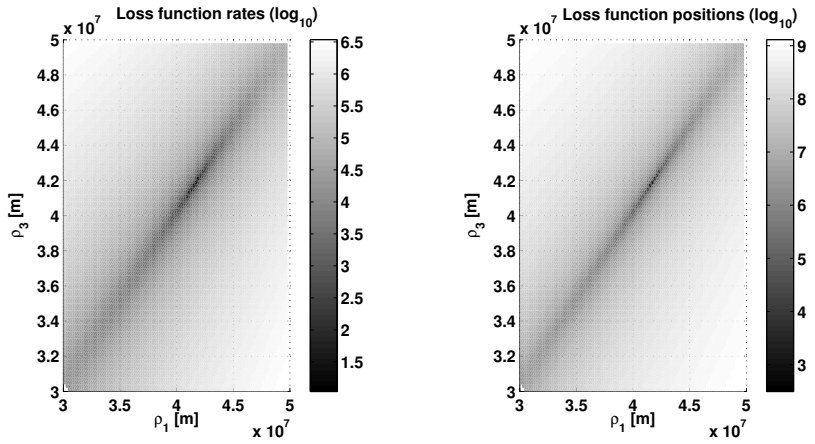

Figure 1: The individual contribution of the angular rates (l) and the angular positions (r) to the loss function. Three tracklets are used in this example.

3. Population Based Meta-Heuristic (PBMH) algorithms in Multiple Target Tracking

As mentioned in the introduction, a few meta-heuristic methods have been investigated within the framework of MTT (Poore and Gadaleta, 2006). In this section we define how an 'individual' is represented, and how such an individual is evaluated through a so-called 'fitness function'. These are two important aspects, that are the same for all of the investigated algorithms.

Definition of an individual and the fitness function

A PBMH 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. The individual has to be able to accurately represent any valid solution in the search space. In this work an individual represents the associations between the tracklets. Equation 5 shows the general notation of a k-matrix. The k-matrix notation was introduced in the work of (Schneider, 2012), where the most likely k-matrix was sought through the application of Markov Chain Monte Carlo computations.

$$K = \begin{pmatrix} 1 & 0 & \dots & 0 \\ k_{2,1} & k_{2,2} & \dots & 0 \\ \dots & \dots & \ddots & 0 \\ k_{i,1} & k_{i,2} & \dots & k_{i,j} \end{pmatrix} \tag{5}$$

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 associated to the object in column j. The k-matrix is defined in such a way that the first tracklet is always associated to the first object. Following this logic the k-matrix becomes a lower triangular matrix. Besides this, each row may only contain one non-zero element such that $\Sigma_{j=1}^{j=N} k_{i,j} = 1$, $i = 1, \dots, N$.

The second design element is the fitness function. The fitness function has to fulfill 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 the detection probability P_d and the false alarm probability P_f have to be taken into account. The P_d describes the probability to detect the object. This probability depends on e.g. the image processing software and the brightness of the object. The P_f describes the probability of having a sporadic measurement due to e.g. measurement noise. We define the event A as the event where none of the N tracklets are a false alarm. The event B is the event where all of the N tracklets are detected. Event C is the event where the object is not detected in the remaining $S - N$ fences. Event D occurs when the measurements stem from the object that is represented by the modeled object. The probability of this event is modeled with a multivariate Normal distribution, since the errors are assumed to be Gaussian. Note that this probability cannot be evaluated in the $N = 1$ case, since the OBVIOD method cannot define an orbit for a single tracklet. The probabilities of having each of these events occur are given in Equations 6 to 9.

$$P(A) = (1 - P_f)^N \tag{6}$$

$$P(B) = P_d^N \tag{7}$$

$$P(C) = (1 - P_d)^{S-N} \tag{8}$$

$$P(D) = \begin{cases} \mathcal{N}\left(\bar{\Theta}, \hat{\bar{\Theta}}(\mathrm{m}^*, \bar{p}^*), \Sigma_{\bar{\Theta}}\right) & \text{if N} \geq 2 \\ 1 & \text{if N} = 1 \end{cases} \tag{9}$$

In case of Equation 9 a value of '1' is used for the $N = 1$ case. An empirical parameter will be introduced that allows the $N = 1$ fitness to be tuned.

All the events are independent from one another. The conjunction of all the events is given in Equation 10. It describes the total probability that a single object exists with the given set of tracklets.

$$P_x = \begin{cases} \mathcal{N}\left(\bar{\Theta}, \hat{\bar{\Theta}}(\mathrm{m}^*, \bar{p}^*), \Sigma_{\bar{\Theta}}(\mathrm{m}^*, \bar{p}^*)\right) (1 - P_d)^{S-N} P_d^N (1 - P_f)^N & \text{if N} \geq 2 \\ (1 - P_d)^{S-1} P_d (1 - P_f) & \text{if N} = 1 \end{cases} \tag{10}$$

Here $x = 1, 2, .., X$ where X is the total number of hypothetical objects that the k-matrix proposes. To obtain the total probability that a k-matrix is correct, all the probabilities of the individual objects have to be combined as shown in Equation 11.

$$P_y = \prod_{x=1}^{X} P_x \tag{11}$$

Where $y = 1, 2, .., Y$ denotes the k-matrix, and Y denotes the population size. When we take the negative log-likelihood of Equation 10, and introduce the tuning parameter γ, we obtain Equation 12.

$$f_x = \begin{cases} L_{N\geq 2}(\mathrm{m}^*, \bar{p}^*) - \ln\left(\dfrac{1}{\sqrt{(2\pi)^n |\Sigma_{\bar{\Theta}}(\mathrm{m}^*, \bar{p}^*)|}}\right) + & N \geq 2 \\ -(S-N)\ln(1-P_d) - N\ln(P_d) - N\ln(1-P_f) & \\ -\ln\left((1-P_d)^{S-1} P_d (1-P_f)\right) + \gamma & N = 1 \end{cases} \tag{12}$$

And Equation 11 becomes Equation 13.

$$f_y = \sum_{x=1}^{X} f_x \tag{13}$$

Expressions similar to Equation 13 can be found throughout the MTT literature, examples being (Blackman, 2004) and (Deb et al., 1997). In Figure 2 the possible fitness values that a group of three tracklets can have is shown. Three tracklets can be arranged as three single tracklets (N=1/N=1/N=1), a group of two tracklets and a single tracklet (N=2/N=1), and a group of three tracklets N=3. Note that the (N=1/N=1/N=1) fitness value and the (N=2/N=1) fitness values can be shifted by changing the γ parameter. For now a value of $\gamma = 0$ is maintained. This value ensures that the fitness value distributions do not overlap. The γ could be further optimized, but this is beyond the scope of this paper.

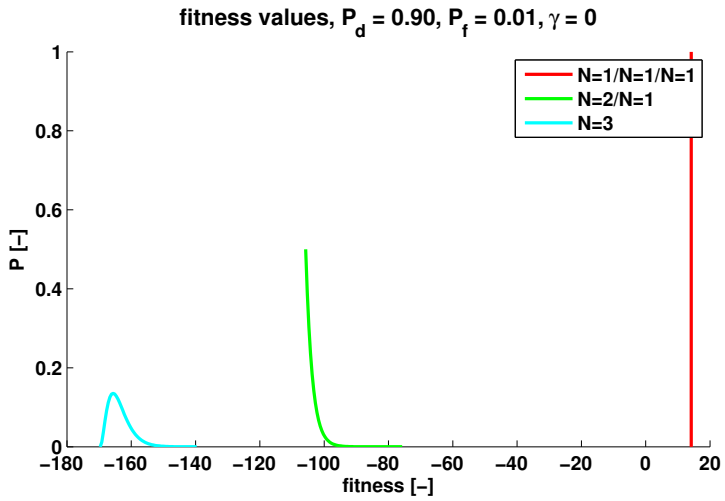

Figure 2: Possible fitness values that a group of three tracklets can have. Three tracklets can be arranged as three single tracklets (N=1/N=1/N=1), a group of two tracklets and a single tracklet (N=2/N=1) and a group of three tracklets N=3.

4. Algorithms

All the presented algorithms work with a population of individuals and use the same representation of an individual and the same fitness function. The goal of each algorithm is to minimize Equation 13, this would lead to

finding the k-matrix that has the maximum probability of being correct. In this chapter each algorithm is briefly presented along with several references that provide more detailed information on the used methods. Each of the algorithms is based on a heuristic which guides a stochastic process. This means that it is not possible to derive any meaningful characteristics (e.g. convergence rate, computational complexity) of the algorithms through an analytical way (Oliveto et al., 2007). Therefore, the discussion of the algorithms is limited to a qualitative description.

4.1. Genetic Algorithm (GA)

The GA is inspired by natural selection. The fitness function is evaluated for a generation 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 4 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 attempt to have a uniform distribution of individuals throughout the search space. In the second step this random initial population gets evaluated. A new population is created by applying the two basic GA operators to the individuals. The first operator is the uniform crossover operator. This operator needs two individuals as an input. The two individuals that it needs are selected based on the relative fitness values, as defined in Equation 14.

$$f_{y_{rel}} = \frac{1}{f_y} \frac{1}{\sum_{y=1}^{Y} (1/f_y)} \tag{14}$$

The $f_{y_{rel}}$ values describe a discrete probability density function (pdf), as shown in Figure 3. The inverse values of f_y are used, since we want the individuals with a low fitness value to be favorable (Equation 13 has to be minimized).

From the discrete pdf of $f_{y_{rel}}$ two individuals are sampled. The uniform crossover operator switches a row between the two individuals with a probability $P_x = 0.5$. Switching a row between k-matrices means that the tracklet corresponding to that row changes the object to which it is associated.

Crossover is seen as the exploiting operator in the GA. To ensure a certain extent of exploration, the mutation operator is used. The mutation operator is applied to every new individual. Each row can be mutated with a user defined mutation probability. The change is applied by randomly redefining

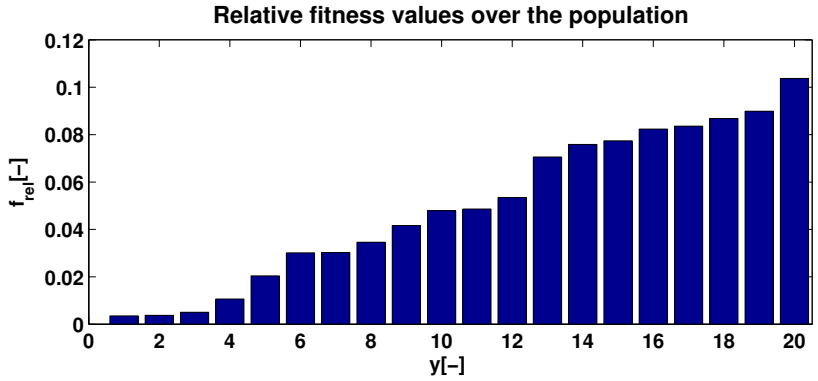

Figure 3: An example of a discrete probability density function made by the relative fitness values in the population.

the column where the '1' occurs, effectively assigning the 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 percent of the population is always copied to the next population, this ensures that the information contained within the best individuals is never lost. The process is repeated until a stopping criterion is met. In this case the stopping criterion is $g >$ max_gens where g is the current generation and max_gens the maximum number of generations allowed. Alternative criteria could be a certain fitness value, or a maximum number of generations where no improvements were found. 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.

4.2. Population Based Incremental Learning (PBIL)

The PBIL algorithm is a close relative of the GA. However, it uses less operators and in some cases converges faster than the GA. It aims at learning a probability distribution from which highly fit individuals can be sampled.

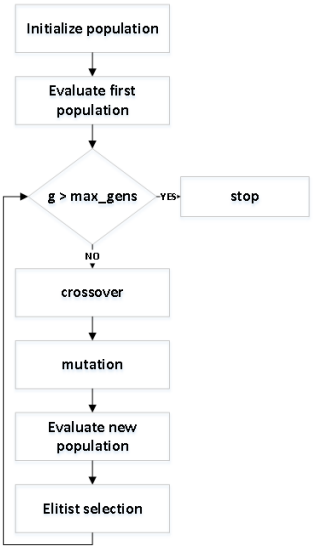

Figure 4: Flowchart of the Elitist Genetic Algorithm

This goal is achieved by using the information that is present in a given population of k-matrices. The probability distribution P for k-matrices is defined by the probability that a given tracklet is associated with a certain object. In other words, the probability given at a certain position (i, j) in the matrix gives the probability of the value at that position being '1'. From this distribution a new k-matrix can be sampled by sampling an object at random (according to the probability distribution) for each tracklet. The flowchart for this algorithm is given in Figure 5. The probability distribution P is initialized to a uniform distribution. This means that initially $P_{i,1:i} = 1/i$ for $i = 1, 2, .., N$. From this distribution the first population is sampled. These individuals are all evaluated as usual. In the current implementation of PBIL, only the very best individual is used to update the probability distribution. This is done according to the update rule in Equation 15 (Baluja, 1994).

$$P_{i,j} = (P_{i,j}\,(1 - LR)) + \left(LR \cdot k^*_{i,j}\right) \tag{15}$$

The matrices P and k^* are the probability distribution and the best k-matrix in the population respectively. The parameter LR is the Learning Rate parameter. It dictates how much impact the current best solution has on the probability distribution. Different update schemes exist, where also the worst individual of the population is used. Experimental results showed

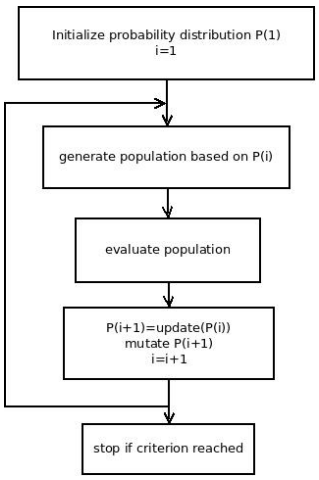

Figure 5: Flowchart of the PBIL algorithm

that the scheme that only uses the best individual in the population performs best. In PBIL there is also a mutation operator in order to ensure versatility in the population and to provide a mechanism to escape from local minima. The mutations can either be performed directly on the individuals (as in GA) or on the probability distribution itself. Here it was opted to mutate the distribution. This is done by randomly adding or subtracting a fixed value from a parameter in the P matrix. This algorithm has shown promise, since in many cases it outperforms the standard GA (Baluja, 1994). One pitfall of this algorithm is that it assumes that all the parameters (in this case tracklet associations) are independent from one another. In the GA these dependencies are taken into account, because it uses the crossover operator. The strength of the GA lies with this crossover operator and its capability of successfully combining 'building blocks' (thus with strong inter dependencies) to come up with promising solutions. It can therefore be expected that if these inter dependencies are very strong (this is a problem dependent issue), the PBIL algorithm could have poorer performance than the classical GA. For further reading about the PBIL algorithm the reader is referred to the work of (Baluja, 1994).

4.3. Differential Evolution (DE)

Differential Evolution was developed to find solutions to problems with continuous variables. Since then efforts have been made to adapt the original algorithm so that it can be applied to discrete (combinatorial) problems

(Onwubolu, 2009; Prado et al., 2010). These efforts have been successful in some cases, however the applicability of DE to combinatorial optimization problems remains debatable. In DE the difference between (real valued) candidate solution vectors is used to guide the search process. The notation $\bar{x}_{g,i}$ is adopted. Here g denotes the generation number and the i is the number of the individual. The $\bar{x}$ means that it is a current member of the population. From these population members three are selected to construct the so called mutation vector $\bar{v}_{g,i}$ as shown in Equation 16.

$$\bar{v}_{g,i} = \bar{x}_{g,r_1} + F\left(\bar{x}_{g,r_2} - \bar{x}_{g,r3}\right) \tag{16}$$

Here $r_1 \neq r_2 \neq r_3$. The parameter F is the scaling parameter. It dictates in what measure the difference vector will perturb the other solution. The mutation vector is combined with the original vector by applying the crossover operator. This results in a new vector called the trial vector $\bar{u}_{g,i}$. The final step is to evaluate the new vector $\bar{u}_{g,i}$, if it is better than the current solution then it will replace it. Otherwise the original solution is kept and copied to the next generation without making any changes. Different schemes exist to create the mutation vector. They differ by the way in which they select the solution vector to change and the number of vectors that are used. In the current implementation the scheme in Equation 17 is applied.

$$\bar{v}_{g,i} = \bar{x}_{g,best} + F\left(\bar{x}_{g,r_1} - \bar{x}_{g,r2}\right) \tag{17}$$

As can been seen in the equation, the difference vector is always applied to the best solution in the population. Furthermore, the r_1 and r_2 solutions are randomly selected according to the relative fitness of each individual. As mentioned earlier, this method was first conceived for applications on real valued problems. There are two ways to adapt DE to handle discrete variables. One way is to adapt the main operator (given in Equation 17). The other way is to transform the solution vectors from the discrete domain to the real domain and to leave the differential operator as it is. Different options are outlined in (Onwubolu, 2009) and (Prado et al., 2010). Here it was opted to transform the solution vectors to the real domain. Note that a k-matrix is easily converted to a vector with integers by setting each entry i in the vector to the value j where $k_{i,j} = 1$. This choice was made based on the simplicity of its implementation as well as its good performance on other combinatorial problems (Onwubolu, 2009). The transformation used is called the forward-backward transformation. The idea is to first transform the population from

the discrete to real domain using the forward transformation, then DE can be applied as usual. After this the new candidate solution vectors are transformed back to the discrete domain with the backward transformation. In Equation 18 the forward transformation is given.

$$\hat{\bar{x}}_{g,i} = -1 + \alpha \bar{x}_{g,i} \tag{18}$$

In the above equation α is set to the constant value of $\alpha = 1 \cdot 10^{-3}$. The value for α is not very important, it only has to ensure a transformation to real (non integer) numbers. The backward transformation is the inverse of the above equation, giving Equation 19.

$$\bar{x}_{g,i} = \text{round}\left(\frac{1}{\alpha}\left(1 + \hat{\bar{x}}_{g,i}\right)\right) \tag{19}$$

It is expected that the DE algorithm will encounter difficulties. This is because the algorithm works with difference vectors. These difference vectors have a physical meaning in a continuous problem. However, in a combinatorial problem there is no physical meaning of the parameter values. The parameter values are symbolic. A difference vector of these symbolic parameter values therefore does not necessarily represent a direction toward an improvement. The vectors will instead point in a more random direction.

5. Comparative study

In this section three cases are presented. All the objects treated in these test cases are in geosynchronous, near circular orbits. Each case is based on simulated observations, where the objects stem from the TLE-Catalog [1] and are propagated with a Keplerian motion. At this stage only objects that follow a Keplerian motion are considered. The exact effects of tracking objects subject to perturbed motion is a topic of future research. In each case the observations are collected at three epochs within the same night. The fences are spaced at two hour intervals in Right Ascension. Furthermore, the observations are made in the topocentric frame associated with the Zimmerwald observatory (Herzog et al., 2010). No visibility conditions other than the elevation of the object are considered here. A Gaussian noise is added to each individual observation with a standard deviation of $\sigma = 1$".

[1] https://www.space-track.org/

Table 1: Settings of the parameters

-	GA	EGA	PBIL	DE
pop. size	2N	2N	2N	2N
$p_{mutation}$	$1/N$	$1/N$	$1/N$	$1/N$
% copied to next generation	10	-	-	-
α	-	-	-	$1 \cdot 10^{-3}$
F	-	-	-	0.8
LR	-	-	0.5	-
Mutation shift	-	-	$1/i$	-
max_gens	150	150	150	150

The settings of the algorithms are kept constant, they can be found in Table 1. Note that the stopping criterion is the same for each algorithm. This is deemed to be fair, since all algorithms have a similar computation cost when evaluating and creating a generation. To reduce the computation time a look up table is maintained. In this table the resulting Mahalanobis distance of each IOD is stored, therefore the same orbit is not determined twice. The first test case involves four objects that are clearly separated, this should be a relatively easy task. For the second study case one of the ASTRA clusters is used. These are again four objects but spaced close together (about 0.06 degrees in declination). It is expected that the convergence of the algorithm will be slower when compared to the first case, since the difference between the correct and erroneous solutions are smaller. For the last test case the two previous situations are mixed. This gives a total of eight objects. It is expected that the algorithm will find the correct associations among the four 'easy' targets relatively soon, and will need more time to correctly distinguish the cluster. For each test case the average convergence is presented. As the algorithms are stochastic it is unrealistic to try to derive any bounds on the convergence behavior through analytical means. Therefore the average performance over multiple runs is presented. In this work the algorithms are applied 100 times for each test case. Each time an algorithm is applied it starts from a different random seed. Also the average k-matrix found at the last generation is shown and compared to the true k-matrix.

First test case

In Figure 6 the observations for the first test case can be seen. Here it is seen that all the tracklets are spaced at considerable distances (about three

degrees in position and $6 \cdot 10^{-4}$ [deg/sec] in angular rates) from each other in all dimensions. Therefore this test case should be an easy one to solve for the algorithm. The numbers noted in the legend of the figure correspond to the NORAD ID numbers of the objects.

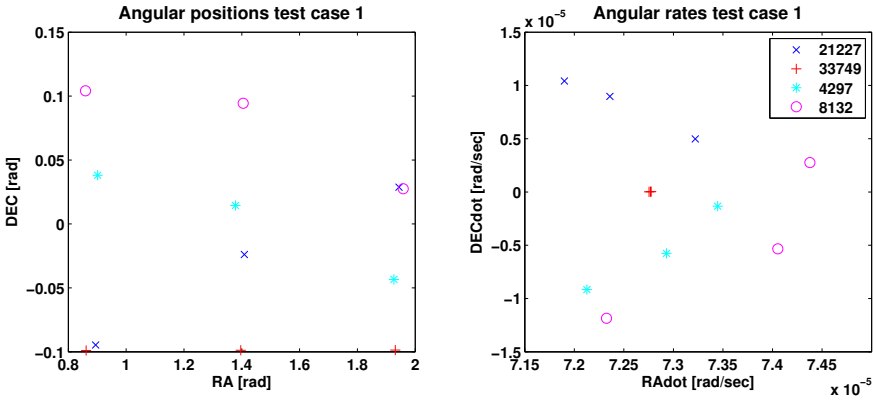

Figure 6: The observations used in the first test case. These are Geosynchronous objects on nearly circular orbit, separated in inclination.

In Figure 7 the average best solution per generation can be found. The straight line represents the optimum solution.

From these results some clear conclusions can be drawn. First of all, the results of the standard GA are not to be seen in this plot. This is because this algorithm is not capable of finding a solution with a fitness value in the range denoted on the y-axis. Surprisingly the DE algorithm already performs much better than the GA. This is unexpected since the DE algorithm is an algorithm that is known to have problems when facing a combinatorial problem (Onwubolu, 2009). The PBIL algorithm seems to converge the fastest, but it reaches a plateau early on in the run. This signifies that the algorithm has found a local minimum and is trapped. The clear winner among these algorithms is the EGA. Recall that the only difference between the GA and the EGA is that in the EGA the best individuals are always copied to the next population. Therefore the computational cost is the same for both algorithms. Apparently it is of paramount importance that the information contained in the top individuals is not lost. This also explains the relative success of DE with respect to the GA, since in DE the best individuals are always kept as well. These results are also reflected in the average k-matrix that each algorithm found at the end of the run (at

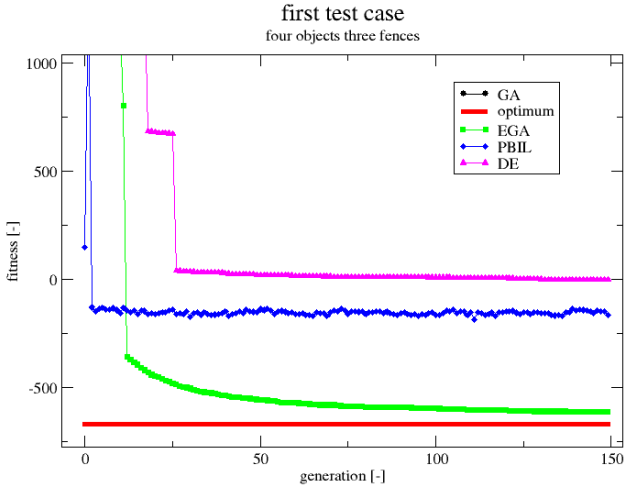

Figure 7: The average best fitness individual found per generation. Average value is based on 100 runs.

generation number 150). The entries of the average k-matrix are given by $\bar{k}_{i,j} = \sum_{r=1}^{R} k_{i,j}^{r}/R$, where $r = 1, 2, .., R$ indicates the number of the run, $R = 100$ is the total number of runs, and $\bar{k}$ is the average matrix. In Figure 8 these k-matrices are compared to each other and to the optimum solution.

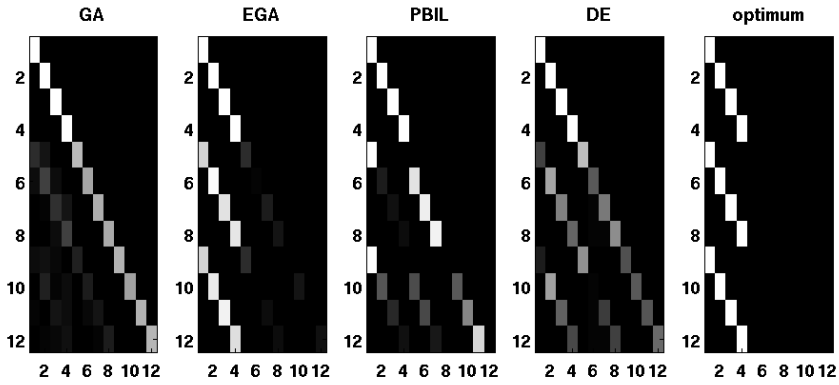

Figure 8: The average k-matrix at the end of the run vs. the true k-matrix for the first test case

The results shown in Figure 7 are in accordance with those shown in Figure

8. The GA is barely able to distinguish any correct associations, instead it tends to put all the tracklets on the diagonal of the k-matrix (effectively assigning each tracklet to a different object). The PBIL algorithm has indeed found a local minimum which is in part correct. It consistently finds one of the objects, however it is not able to form the complete 3-tuples of tracklets of the other objects. Finally the DE algorithm consistently finds correct associations but is not able to form the complete groups of tracklets either. The success of the EGA can be accredited to the fact that it takes the interdependencies among the parameters (tracklets) into account.

Second test case

The second test case concerns a satellite cluster. Satellite clusters form one of the most challenging situations in space debris tracking because of the close proximity that the objects have to each other. Therefore it is an interesting test case for the proposed algorithms. In Figure 9 the observations for this test case are found.

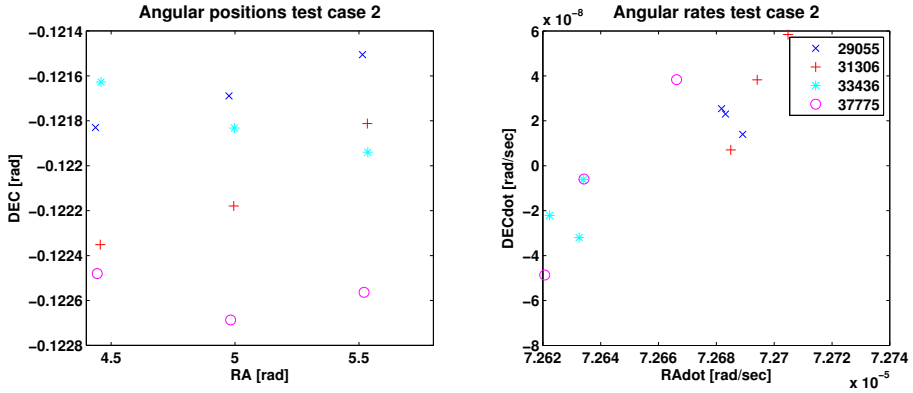

Figure 9: The observations used in the second test case. These are closely spaced objects belonging to one of the ASTRA clusters

In Figure 10 the average best solution per generation can be found.

The results show a similar behavior as in the first test case. An interesting observation here is that the GA manages to parallel the performance of PBIL. This is because a wrong association in this scenario will not necessarily lead to a bad fitness value, due to the tracklets being closely spaced, and their angular rates being similar to one another. Again the importance of

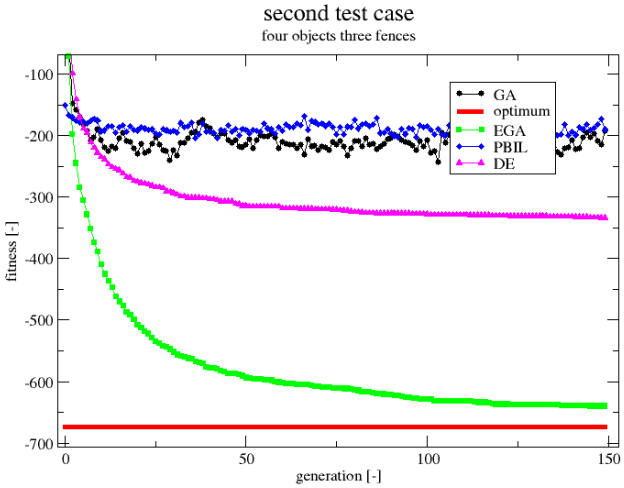

Figure 10: The average best fitness individual found per generation. Average value is based on 100 runs.

keeping the best solutions is shown. The EGA clearly outperforms the other algorithms. In Figure 11 the average k-matrices at the end of the run can be found.

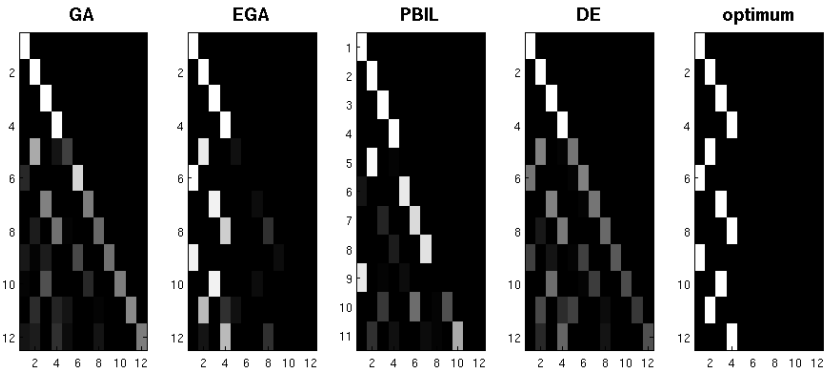

Figure 11: The average k-matrix at the end of the run vs. the true k-matrix for the second test case.

In these results it is again evident that the EGA is the only algorithm capable of approximating the correct solution in a reliable way. PBIL again

gets stuck in a local minimum, and DE finds good associations but is unable to form the complete groups of three tracklets.

Third test case

In the third test case the two previous cases are mixed. The expectation is that the algorithms are able to quickly find the correct associations for the easy targets and will focus on the cluster afterwards. In Figure 12 the observations used in this test case are seen.

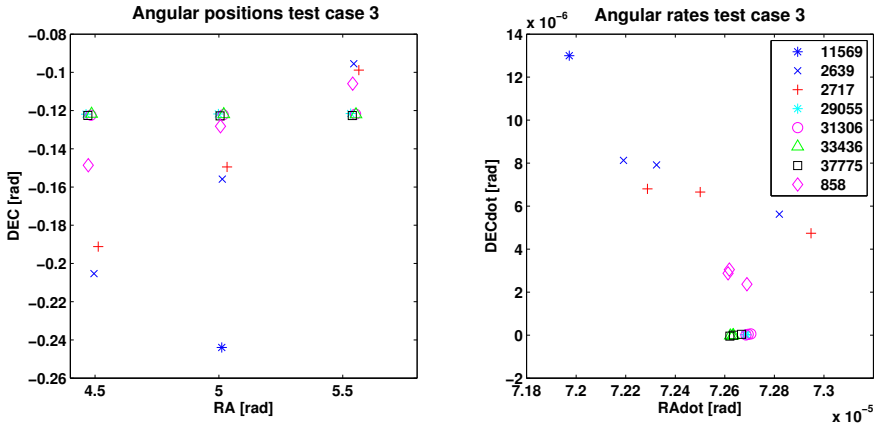

Figure 12: The observations used in the third test case. This test case contains both objects that are separated in inclination as well as an ASTRA cluster and a single tracklet

In Figure 13 the average best fitness value per generation can be seen. These results are again in accordance with the results of the previous test cases. The EGA is clearly the only algorithm capable of finding a reasonable solution

Figure 14 shows the average k-matrix found at generation 150. The first four columns of this matrix correspond to the four objects that are in the cluster. It is clear that the EGA would need more time to distinguish these objects with certainty. Columns five until seven correspond to the objects 2639, 2717 and 858, which are the relatively easy targets. Although the EGA is still unsure about the objects in the cluster, it has associated these three easy targets correctly. This shows that the EGA is able to identify the objects in the more sparsely populated areas with relative ease, after which it focuses on the high density regions such as satellite clusters. Also, it has identified the stand-alone tracklet without any issues.

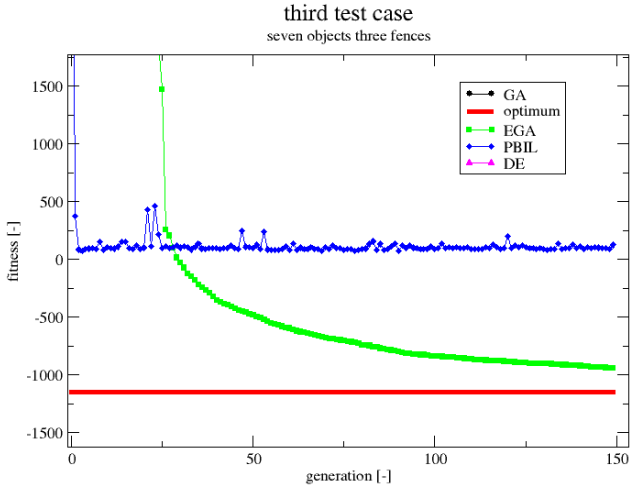

Figure 13: The average best fitness individual found per generation. Average value is based on 100 runs.

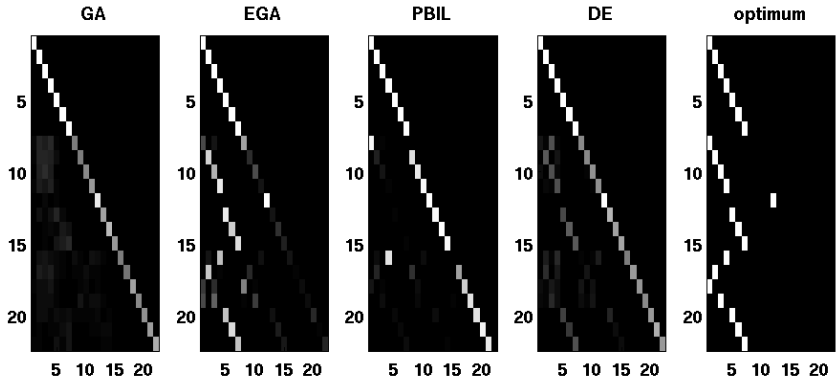

Figure 14: The average k-matrix at the end of the run vs. the true k-matrix for the third test case

6. Time complexity study

As mentioned in the introduction, the challenge is to find a good approximate solution in a reasonable computation time. It is clear that the EGA is capable of finding a good approximate solution, the question that remains is whether it can do so in a realistic (polynomial) 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 (the GA is a type of Evolutionary Algorithm). 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 size is said to change when the number of tracked objects changes. 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 tracklet is observed every hour for a total of three tracklets per object.

Figure 15 shows the average k-matrix at different stages of the computation for a problem with five objects. Four matrices are shown, at 80%, 90%, 95% of the optimum value, and the true k-matrix. The computational burden is concentrated at the beginning of the run. A look up table is used, 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.

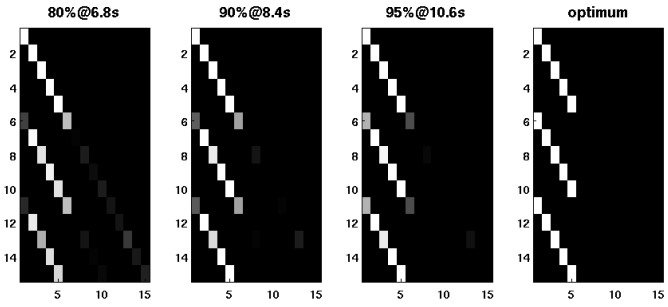

Figure 15: The average k-matrix found at different stages of the search. Both the percentage of the optimum fitness as well as the time at which the solution is found are given.

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

In Figure 16 the absolute computation time in seconds is plotted against the number of objects that are being tracked. It is the time needed to have a solution at 90% of the optimum fitness value. These tests were performed on a machine with a 3.16 GHz CPU. The tests involved 11 tracked objects at maximum. Because for each number of tracked objects the algorithm has to be run 100 times, it was chosen to not go further than 11 objects (about two hours of computation) in order to limit the computation time. 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 data is well described by a polynomial model. 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 $b = 2.69$.

The population size and the number of objects per k-matrix scale linearly with the total number of tracklets (pop.size $= 2N$, $X \sim N$). This means that the computation time needed to evaluate a generation will scale with N^2. The parameter b of the polynomial fit is greater than two for several reasons. The first is that the number of generations needed to arrive at a good approximate solution will increase when more objects are tracked. However, due to the look-up table it is relatively cheap to compute additional generations after the initial few generations. Therefore, this has a minor impact on the total computation time. Also the computation time needed by the crossover and mutation operators is relatively small, but will still make a contribution.

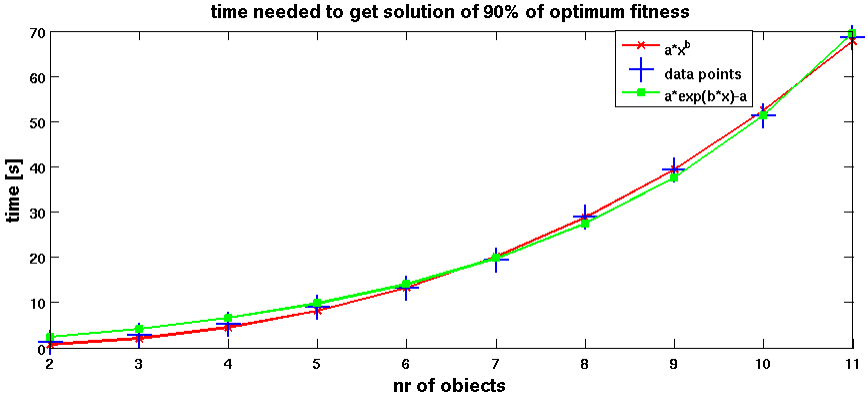

Figure 16: Computation time needed to reach an approximate solution of a certain quality (90% of optimum) versus the number of tracked objects. Two lines are fitted, a polynomial and an exponential function. Each point represents the average computation time over 100 runs.

Therefore, it is to be expected that a polynomial trend fits well to the data points. This polynomial behavior holds true for the range that is shown (two to 11 tracked objects). Although it is expected that the polynomial trend continues outside of this range, there is no way to be certain without adding additional data points.

7. Conclusions

This work aimed to find a method that is able to approximate the solution to the $S \geq 3$ MTT problem in a reasonable computation time. Several algorithms have been tested, namely the GA, EGA, PBIL and DE algorithms. Three test cases were considered, the results throughout these test cases are consistent. In each case the EGA was the only algorithm capable of consistently finding the complete groups of tracklets. A study has been performed to determine the relationship between computation time of the EGA and the number of tracked objects. The outcome of the study is favorable as it shows that the computational complexity is well described by a polynomial model (at least up to 11 tracked objects). A straightforward method to mitigate a large absolute computation cost is to adapt a parallel architecture. Genetic Algorithms are particularly well suited for such an architecture and can benefit from a more than linear increase in performance when an Island-Based structure is used (Calegari et al., 1997).

Future research will focus on finding an efficient way to reduce the search space for the EGA. In this way the EGA can focus on more challenging situations such as satellite clusters and break-ups, effectively reducing the number of objects that are handled. Once such a method is in place the algorithm can be applied to data sets of realistic size (500-1000 tracklets). Finally the goal is to collect real observations through a survey type observation strategy and to test the algorithm on the real observations. To reach that goal an intermediate step will be taken, where the effects of different dynamics models in the tracklet simulations are studied.

Acknowledgments

This study has been performed within the framework of a PhD program which is co-funded by the CNES, TAS and the AIUB.

J. M. Aristoff, J. T. Horwood, N. Singh, et al. Multiple hypothesis tracking (MHT) for space surveillance: theoretical framework. Technical report, Numerica Corporation, 2013.

S. Baluja. Population based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical report, Carnegie Mellon University, 1994.

R. H. Battin. *An introduction to the mathematics and methods of astrodynamics, Revised Edition*. Number ISBN-13 978-1563473425. AIAA Education Series, 1999.

J. E. Beasley and P. C. Chu. A genetic algorithm for the set covering problem. *Eur. J, Oper. Res.*, 1995.

S. S. Blackman. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerosp. Electron. Syst. Mag*, 2004.

P. Calegari, F. Guidec, P. Kuonen, et al. Parallel island-based genetic algorithm for radio network design. *J. Parallel. Distr. Com.* , 47:86–90, 1997.

G. Chen and L. Hong. A genetic algorithm based multi-dimensional data association algorithm for multi-sensor-multi-target tracking. *Math. Comput. Model.*, 26, 1997.

S. Deb, M. Yeddanapudi, K. Pattipati, et al. A generalized S-D assignment algorithm for multisensor-multitarget state estimation. *IEEE Trans. Aerosp. Electron. Syst.*, 33(2), 1997.

K. J. DeMars, I. I. Hussein, C. Früh, et al. Multiple-object space surveillance tracking using finite-set statistics. *J. Guid. Control Dynam.*, 38(9):1741–1756, 2015.

C. Früh, T. Schildknecht. Object image linking of earth orbiting object in the presence of comics. *Adv. Space Res.*, 49:594–602, 2012.

K. Fujimoto, D. J. Scheeres, J. Herzog, et al. Association of optical tracklets from a geosynchronous belt survey via the direct bayesian admissible region approach. *Adv. Space Res*, 53(2), 2014.

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

J. Herzog, C. Früh, T. Schildknecht. Build-up and maintenance of a catalogue of geo objects with zim-smart and zimsmart 2. In *61st International Astronautical Congress, Prague, Czech Republic. IAC-10 A*, volume 6, 2010.

P. Larranaga, R. Etxeberria, J. A . Lozano, et al. Combinatorial optimization by learning and simulation of bayesian networks. In *Uncertainty in Artificial Intelligence Proceedings*, 2000.

A. Milani, G. F. Gronchi, M. D. M. Vitturi, et al. Orbit determination with very short arcs. I admissible regions. *Celestial Mech. Dyn. Astron.*, 90(1): 57–85, 2004.

A. Milani, D. Farnocchia, A. Rossi, et al. Orbit determination of space objects based on sparse optical data. *Mon. Not. R. Astron. Soc.*, 417: 2094–2103, 2011.

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

G. C. Onwubolu. *Differential Evolution: A handbook for global permutation based combinatorial optimization*. Springer, 2009.

A. B. Poore and S. Gadaleta. Some assignment problems arising from multiple target tracking. *Math. Comput. Model.*, 43, 2006.

R. S. Prado, R. C. P. Silva, F. G. Guimaraes, et al. Using differential evolution for combinatorial optimization: A general approach. Technical report, Instituto Federal de Minas Gerais, 2010.

W.H. Press, S.A. Teukolsky, W.T. Vetterling, et al. *Numerical recipes: the art of scientific computing*. Cambridge University Press, third edition, 2007.

J. F. Pusztaszeri, P. E. Rensing, T. M. Liebling. Tracking elementary particles near their primary vertex: A combinatorial approach. *J. Global Optim.* , 9:41–46, 1996.

B. Rakdham. *Efficient multiple hypothesis track processing of boost-phase ballistic missiles using IMPULSEA-generated threat models.* PhD thesis, Naval Postgraduate School, 2009.

C. R. Reeves. A genetic algorithm for flowshop sequencing. *Computers Ops. Res.*, 22(1):5–13, 1995.

A. J. Robertson. A set of greedy randomized adaptive local search procedure (GRASP) implementations for the multidimensional assignment problem. *Comput. Optim. Appl.*, 19(2):145 – 164, 2001.

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

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

J.A. Siminski, O. Montenbruck, H. Fiedler, et al. Short-arc tracklet association for geostationary objects. *Adv. Space Res*, 53(8), 2014.

G. Tommei, A. Milani, A. Rossi. Orbit determination of space debris: admissible regions. *Celestial Mech. Dyn. Astr.*, 2007.

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