

IAC-15-A6.9.10

ASSOCIATING OPTICAL MEASUREMENTS AND ESTIMATING ORBITS OF GEOCENTRIC OBJECTS THROUGH POPULATION-BASED META-HEURISTIC METHODS

M. Zittersteijn
Astronomical Institute of the University of Bern, Switzerland, michiel.zittersteijn@aiub.unibe.ch

Dr. A. Vananti
Astronomical Institute of the University of Bern, Switzerland, alessandro.vananti@aiub.unibe.ch
Prof. T. Schildknecht
Astronomical Institute of the University of Bern / SwissSpace Association, Switzerland, thomas.schildknecht@aiub.unibe.ch
Dr. J.C. Dolado Perez
Centre National d'Etudes Spatiales, France, Juan-Carlos.DoladoPerez@cnes.fr
Dr. V. Martinot
Thales Alenia Space, France, vincent.martinot@thalesaleniaspace.com

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. The number S corresponds to the number of fences involved in the problem. Each fence consists of a set of observations where each observation belongs to a different object. The $S \geq 3$ MTT problem is an NP-hard combinatorial optimization problem. There are two general ways to solve this. One way is to seek the optimum solution, this can be achieved by applying a branch-and-bound algorithm. When using these algorithms the problem has to be greatly simplified to keep the computational cost at a reasonable level. Another option is to approximate the solution by using meta-heuristic methods. These methods aim to efficiently explore the different possible combinations so that a reasonable result can be obtained with a reasonable computational effort. To this end several population-based meta-heuristic methods are implemented and tested on simulated optical measurements. 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.

I. 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[1]. Despite its challenges, several attempts have been made to solve this problem in an efficient manner. The Multiple Hypothesis Tracking (MHT) algorithm[2,3] 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[4].

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[5,6,7]. 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.

An algorithm is sought 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. A key feature of a PBMH algorithm is that it seeks to improve upon each iteration (a.k.a. generation), instead of attempting to formulate the optimum solution in one step. Therefore they are generally able to find a reasonable approximate solution within a relatively short time span. In this work a novel method of orbit determination is employed. The new orbit determination method is heavily based on the previous work done by Siminski[7] and should be seen as an extension to that method allowing for orbits to be determined with three or more series of closely spaced observations (a.k.a. tracklets).

The paper is organized as follows. In the next section the Optimized Boundary Value Orbit Determination (OBVOD) method is presented. In the third part the basic workings of each of the PBMH algorithms are discussed. The fourth part covers the application of the resulting algorithm to three test cases. In these test cases the observations are taken from simulations of three different scenarios ranging from easy to difficult. 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.

II. 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:

$$\left(\alpha, \dot{\alpha}, \delta, \dot{\delta}\right)_t \qquad [1]$$

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[7] 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 (ρ_1, ρ_2) are unknown. The method by Siminski[7] seeks to find the (ρ_1, ρ_2) combination that leads to the best fitting orbit w.r.t. the angular rates.

Search for the optimum hypothesis

In the original method described by [7] the quantity in Equation [2] has to be minimized.

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

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 $\Sigma_{\dot{z}}$ denotes the covariance matrix which consists of two parts. The first part is the $C_{\dot{z}}$ matrix which represents the uncertainties on the estimated angular rate (which comes directly from the tracklets). The second part is the $C_{\hat{\dot{z}}}$ which contains the uncertainties on the computed angular rates. The k stands for the number of orbital revolutions between the first and second epoch. Finally the $\bar{p}$ is the hypothesis (ρ_1, ρ_2). 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 (here the

Broyden-Fletcher-Goldfarb-Shanno algorithm is used). The procedure to evaluate the loss function is simple. For a given hypothesis $\bar{p}$ 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. By exploiting the fact that the Mahalanobis distance is χ^2 distributed, a threshold can be set. If the Mahalanobis distance is below that threshold, a decision is made to correlate those two tracklets.

The MTT 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 $\dot{z}$ and $\dot{\hat{z}}$ respectively 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 $S \geq 3$ in size.

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

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.

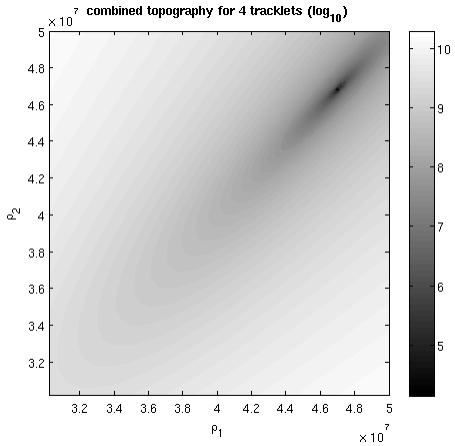

Figure 1: The topography of the sum of the Mahalanobis distances of the angular positions and rates for a group of four tracklets that belong to the same object.

Bounds on the range hypothesis

To restrict the search for the best hypothesis (ρ_1, ρ_2) a so-called Admissible Region (AR) is used. The AR allows to constrain the (ρ_1, ρ_2) search space by using simple to evaluate relationships between bounds on the orbital elements and the topocentric range values. Four of the basic bounds are used in this work, they are described by Schumacher[5]. The first bound is on the minimum and maximum range values. This bound is based on the maximum allowed apogee and the minimum allowed perigee, each geocentric vector should fall within this range. Equation [4] represents the allowed range of orbital radii.

$$(a_{\min}(1 - e_{max}))^2 \leq \|\bar{r}\| \leq \left(a_{max}\ (1 + e_{max})\right)^2 \quad [4]$$

These bounds on the geocentric vector $\bar{r}$ can be translated to bounds on the range values through the geometric relationship $\bar{r} = \bar{R} + \rho\bar{u}$. Here $\bar{R}$ is the geocentric position of the observer and $\bar{u}$ is the unit vector in the direction of the observation.

The remaining three bounds are implied by Lambert's theorem. The first bound is introduced by constraining the eccentricity. This relationship is given by Equation [5].

$$0 \leq e_0 = \frac{|(\|\bar{r_1}\| - \|\bar{r_2}\|)|}{\|\bar{r_2} - \bar{r_1}\|} \leq 1 \quad [5]$$

In Equation [5] the e_0 denotes the minimum possible eccentricity, $\bar{r_1}$ and $\bar{r_2}$ are the geocentric positions of the object at the epochs of the first and second tracklets respectively. The second constraint is given by [6].

$$4a_0 = \|\bar{r_1}\| + \|\bar{r_2}\| + \|\bar{r_2} - \bar{r_1}\| \quad [6]$$

Here the a_0 is the minimum semi-major axis possible of the orbit that intersects both geocentric positions. If $a_0 > a_{max}$ or $e_0 > e_{max}$ then the (ρ_1, ρ_2) hypothesis can be rejected. The last constraint is imposed on the time of flight. Equation [7] denotes the zero energy time of flight, which corresponds to a parabolic orbit. Eccentric orbits (negative energy orbits) will always have a longer time of flight. Therefore if $t_2 - t_1 < \Delta t_p$ the hypothesis can be rejected.

$$\Delta t_p = \frac{4}{3}\sqrt{\frac{a_0^3}{\mu}}(1 - s\lambda^3) \quad [7]$$

In Equation [7] the μ is Earth's gravitational constant, s is either +1 for the short way trajectories or -1 for the long way trajectories. The λ is a function of the geocentric positions as shown in Equation [8].

$$\lambda = \frac{\|\bar{r_1}\| + \|\bar{r_2}\| - \|\bar{r_2} - \bar{r_1}\|}{\|\bar{r_1}\| + \|\bar{r_2}\| + \|\bar{r_2} - \bar{r_1}\|} \leq 1 \quad [8]$$

III. META-HEURISTIC POPULATION-BASED (MHPB) ALGORITHMS IN MTT

As mentioned in the introduction, a few meta-heuristic methods have been investigated within the framework of MTT[1]. Examples of these algorithms are the Lagrangian relaxation technique[4] and the GRASP algorithm[8]. However, both these methods work with a single solution as opposed to a population of solutions.

The Genetic Algorithm (GA) is a popular MHPB algorithm which was conceived in the 1960s[9]. 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 MHPB algorithms. Other algorithms with a high potential are the Population Based Incremental Learning (PBIL)[10] and the Differential Evolution (DE)[11] algorithms. Both of these algorithms have shown some promise when applied to combinatorial optimization problems, such as the travelling salesman problem.

Definition of an individual and the fitness function

An MHPB 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 any MHPB 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 [9] the general notation for a k-matrix can be seen. The k-matrix notation was introduced in the work of Schneider[12], where the most likely k-matrix was sought through the application of Markov Chain Monte Carlo computations.

$$K = \begin{matrix} 1 & \cdots & 0 \\ \vdots & \ddots & 0 \\ k_{i,1} & \ldots & k_{i,j} \end{matrix} \quad [9]$$

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 [10] is found.

$$P_{N\geq 2} = \mathcal{N}\left(\dot{z}, \hat{\dot{z}}, \Sigma_{\dot{z}}\right)\mathcal{N}\left(z, \hat{z}, \Sigma_z\right) (1 - P_d)^{L-N} P_d^N \left(1 - P_f\right)^N \quad [10]$$

In Equation [10] 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 [11] is obtained.

$$
\begin{aligned}
L_{N\geq 2} = & -\ln\left(\frac{1}{\sqrt{(2\pi)^k|\Sigma_{\dot{z}}|}}\right) \\
& -\ln\left(\frac{1}{\sqrt{(2\pi)^k|\Sigma_z|}}\right) \\
& +\frac{1}{2}\left(\dot{z}-\hat{\dot{z}}\right)^T\Sigma_{\dot{z}}^{-1}\left(\dot{z}-\hat{\dot{z}}\right) \\
& +\frac{1}{2}(z-\hat{z})^T\Sigma_z^{-1}(z-\hat{z}) \\
& -(L-N)\ln(1-P_d) \\
& -N\ln(P_d)-N\ln(1-P_f)
\end{aligned}
\qquad [11]
$$

The k in Equation [11] stands for the dimension of the vector (either $\dot{z}$ or z). In Equation [11] 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 independent from each other, therefore their probabilities can be added in order to determine the combined probability. Adding these probabilities and taking the negative log-likelihood gives the expression in Equation [12].

$$L_{N=1} = -\ln((1-P_d)^{L-1}P_d + P_f) \qquad [12]$$

Equation [12] is currently used in the algorithm. It seems to fulfil the requirements of the fitness function (which are listed at the start of the section). However the definition of the fitness function for $N = 1$ is still a point of discussion. Alternative definitions could involve setting threshold values (based on e.g. the χ^2 distributions), or an empirical formulation.

III. ALGORITHMS

As mentioned in the introduction, several different algorithms are implemented and tested. All algorithms work with a population of individuals and use the same representation of an individual and the same fitness evaluation function. The way in which each algorithm uses the information contained within the population is where the difference lies. In this chapter each algorithm is briefly presented along with several references that provide more detailed information on the used methods.

Genetic Algorithm

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 i.a. on 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. Each row can be mutated with a user defined mutation probability. The change is applied by randomly redefining 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 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.

For further reading the work of Goldberg[9] is recommended which provides a thorough discussion of

GAs. In the works of Chen[13] and Turkmen[14] the GA is applied to an MTT problem.

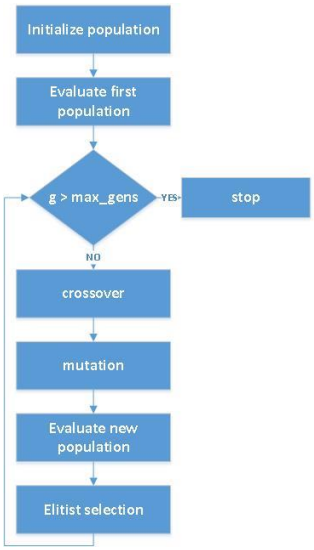

Figure 2: Flowchart of an elitist GA

Population Based Incremental Learning

The PBIL algorithm is a close relative of the GA. However, it is even simpler and in some cases also more powerful. It aims at learning a probability distribution from which highly fit individuals can be sampled. 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 $k_{i,j}$ 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. This algorithm is straightforward to implement and only consists of a few steps that are repeated until a certain criterion is reached. The flowchart for this algorithm is given in Figure 3. The probability distribution P is initialized to a uniform distribution for each tracklet. 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 [13].

$$P_{i,j} = \left(P_{i,j}(1 - LR)\right) + \left(LR \cdot K_{i,j}\right) \qquad [13]$$

Here the subscript i stands for the tracklet number and j stands for the object number. 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. The LR parameter is an important parameter since it effectively controls the convergence rate. If it is set too high, the PBIL algorithm will converge quickly but to a local minimum. If it is set too low, the algorithm will converge slowly. Different update schemes exist, however after some experimentation with these it was opted to use the simplest scheme of only using the best solution. 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[10]. One pitfall of this algorithm however is that it assumes that all the parameters 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. Research is being done in this area at the moment. Instead of learning the probability distribution as in PBIL, the goal is to learn the inter dependencies between the parameters. In this case both a model and the inter dependency values have to be learned, together this makes up a Bayesian Network. Examples of such algorithms can be found in[15,16]. For further reading about the PBIL algorithm the reader is referred to the work of Baluja[10].

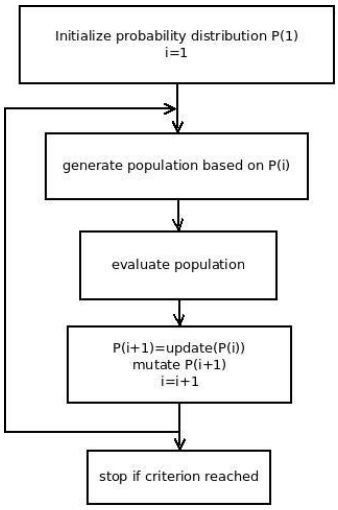

Figure 3: Flowchart for the PBIL algorithm

Differential Evolution

Differential Evolution was first thought of in the 1990s in order to find optimal 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[1711]. These efforts have been successful in some cases, however the applicability of DE to combinatorial optimization problems remains debatable. The main difference between DE and GA is the way in which new individuals are produced. In GA this is done by the crossover and mutation operators. In DE the difference between (real valued) candidate solution vectors is used to guide the search process. The notation $x_{g,i}$ is adopted. Here g denotes the generation number and the i is the number of the individual. The 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 $v_{g,i}$ as shown in Equation [14].

$$v_{g,i} = x_{g,r_1} + F(x_{g,r_2} - x_{g,r_3}) \quad [14]$$

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. Now the mutation vector is used to construct a trial vector $u_{g,i}$. This is done by combining the mutation vector $v_{g,i}$ with $x_{g,i}$ by exchanging parameter values between the two vectors. This is done by applying the crossover operator (as in the GA). The final step is to evaluate the new vector $u_{g,i}$, if it is better than the current solution $x_{g,i,j}$ 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 [15] is applied.

$$v_{g,i} = x_{g,best} + F(x_{g,r_2} - x_{g,r_3}) \quad [15]$$

As can been seen in the equation, the difference vector is always applied to the best solution in the population. Furthermore, the r_2 and r_3 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 of the algorithm as given in Equation 2. 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[17,11]. Here it was opted to transform the solution vectors to the real domain. 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 [16] the forward transformation is given.

$$\hat{x}_{g,i} = -1 + \alpha x_{g,i} \quad [16]$$

In the above equation α is fixed to a value of 0.01. 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 [17].

$$x_{g,i} = \text{round}\left(\frac{1}{\alpha}\left(1 + \hat{x}_{g,i}\right)\right) \quad [17]$$

IV. COMPARATIVE STUDY

With the algorithms in place several test cases can be studied. In this section three cases are presented. Each case is based on simulated observations, where the objects stem from the TLE-Catalog* and are propagated with a Keplerian motion. Note that in this case the model used in the simulations is the exact same as the one used in the computations, since solving the Lambert problem employs a Keplerian model as well. In each case the observations are collected at three epochs within the same night. A series of seven observations

* https://www.space-track.org/

are made at a rate of one image per 30 seconds. This series represents one tracklet. 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. 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 $\sigma = 1"$. The settings of the algorithms are kept constant, they can be found in Table 1. To reduce the computation time a look up table is maintained. Whenever the fitness function is evaluated for a given set of tracklets the result is stored in this table. Before each fitness evaluation it is checked whether the tracklet group has already been treated, if so its fitness is taken from the table. The first test case involves four objects that are clearly separated in inclination, 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 in the case of the cluster. 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 correlations among the four 'easy' targets relatively soon, and will need more time to correctly distinguish the cluster. These tests are performed without any gating or limitations on the search space, which are topics of future research. 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 it is common practice to present the average performance over multiple runs. In this work the algorithms are applied 100 times for each test case. Also the average k-matrix found at the last generation is shown and compared to the true k-matrix.

	GA	EGA	PBIL	DE
Pop. Size	2 x N	2 x N	2 x N	2 x N
$p_{mutation}$	1 / N	1 / N	1 / N	1 / N
$p_{crossover}$	0.5	0.5	0.5	0.5
% copied	10	N/A	N/A	N/A
α	N/A	N/A	N/A	1e-3
F	N/A	N/A	N/A	0.8
LR	N/A	N/A	0.05	N/A
Mutation shift	N/A	N/A	1 / *i*	N/A

Table 1: Parameter settings of the algorithms. These settings are kept constant throughout all the tests. The *i* in the mutation shift parameter of PBIL denotes the row number in the k-matrix.

First test case

In Figure 4 the observations for the first test case can be seen. Here it is seen that all the tracklets are spaced at considerable distances from each other. 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.

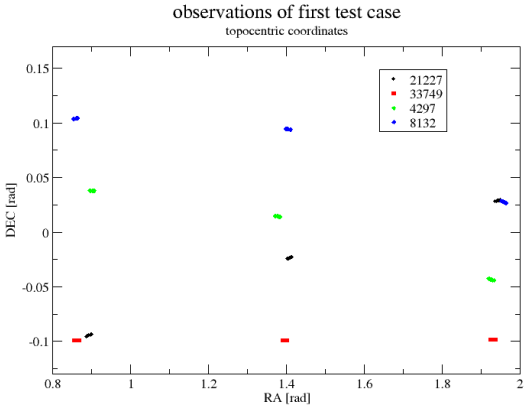

Figure 4: Right Ascension and Declination of the observations used in the first test case.

In Figure 5 the average best solution per generation can be found. The straight red line represents the optimum solution.

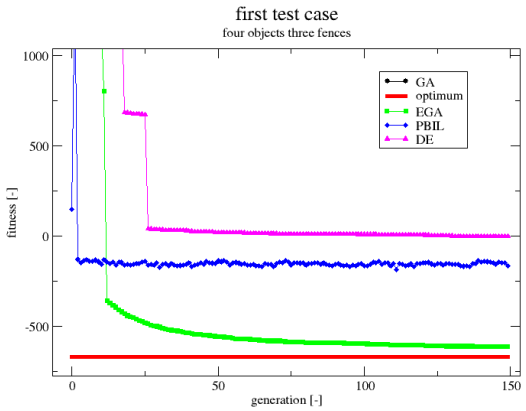

Figure 5: Average best fitness per generation for each of the algorithms applied to the first test case.

From these results a few 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. Apparently the standard GA is not suitable when attempting to track Space Debris. 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. The PBIL algorithm seems to converge rapidly (the fastest of them all), but it reaches a plateau very early on in the run. This signifies that the algorithm has found a local minimum and is trapped. This could be remedied by setting the LR parameter lower, and the mutation probability and the impact of the mutation operator on the probability distribution higher. 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 exactly 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 generation number 150). In Figure 6 these k-matrices are compared to each other and to the optimum solution.

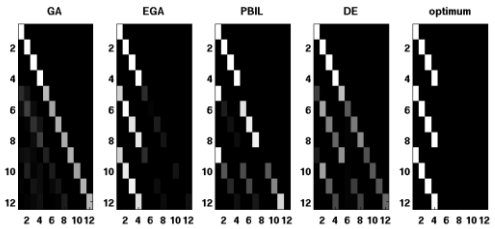

Figure 6: The average best k-matrix found at generation number 150 for each algorithm compared to the true solution.

The results shown in Figure 6 are in accordance with those shown in Figure 5. The GA is barely able to distinguish any correct correlations, 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 correlations 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. It does this through the use of the crossover operator, which preserves the so-called building blocks which contain these dependencies. Besides this, the EGA always keeps the individuals with the best fitness score. In this way these individuals will always be used in the next generation when new individuals have to be made. The regular GA discards the whole previous population, therefore it is not able to improve the previous best solution. In this test case a small mistake in correlation will quickly lead to a bad fitness score due to the fact that the tracklets are spaced so far apart from each other.

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 objects' close proximity to each other. Therefore it is an interesting test case for the proposed algorithms. In Figure 7 the observations for this test case are found.

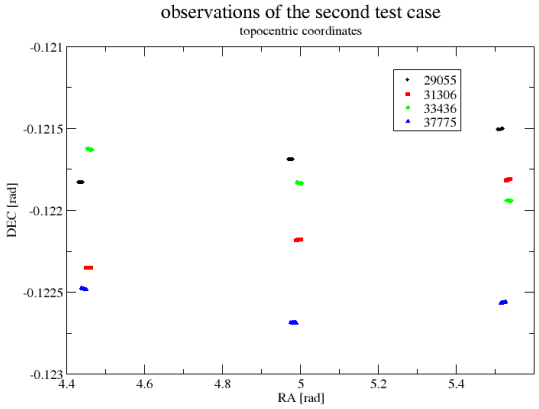

Figure 7: Right Ascension and Declination of the observations used in the second test case (ASTRA cluster).

In Figure 8 the average best solution per generation can be found.

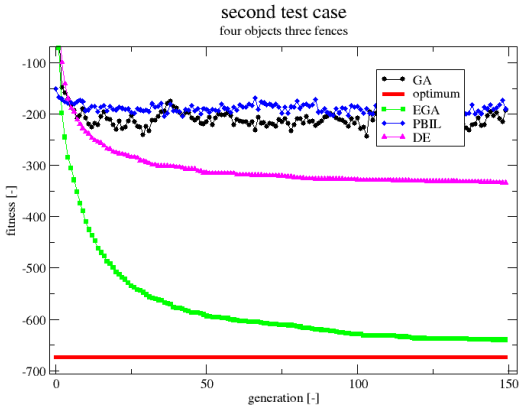

Figure 8: Average best fitness per generation for each of the algorithms applied to the second test case.

The results show a similar behaviour 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 correlation 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 keeping the best solutions is shown. The EGA clearly outperforms the other algorithms. In Figure 9 the average k-matrix at the end of the run can be found.

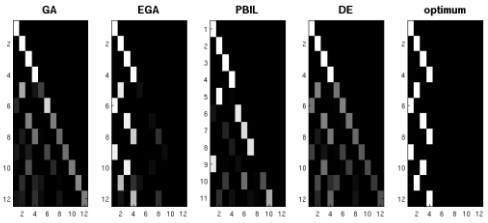

Figure 9: the average best k-matrix found at generation number 150 for each algorithm compared to the true solution.

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 seems to get stuck in a local minimum, and DE consistently finds good correlations but is unable to form the complete groups of three tracklets.

Third test case

In the third test case the previous two cases are mixed. The expectation is that the algorithms are able to quickly find the correct correlations for the 'easy' targets and will focus on the cluster afterwards. There is one single tracklet included of an object that has only been observed in one fence. In Figure 10 the observations used in this test case are seen.

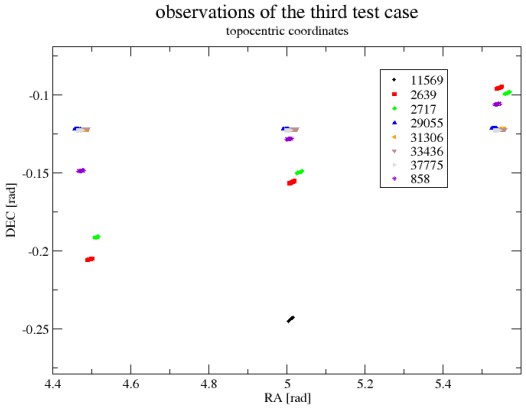

Figure 10: Right Ascension and Declination of the observations used in the third test case.

In Figure 11 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.

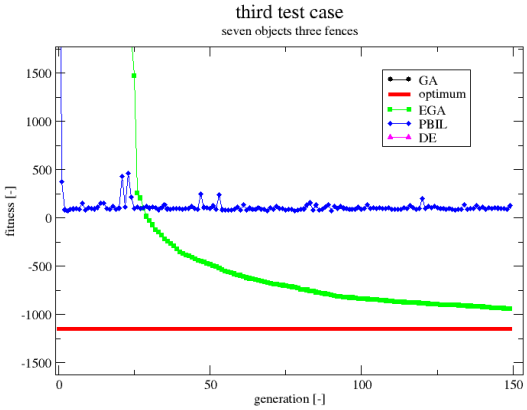

Figure 11: Average best fitness per generation for each of the algorithms applied to the third test case.

Figure 12 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 correlated 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. Note that this should not have posed any problems since this tracklet is spaced far apart from the other observations (both in angular position as in angular rate).

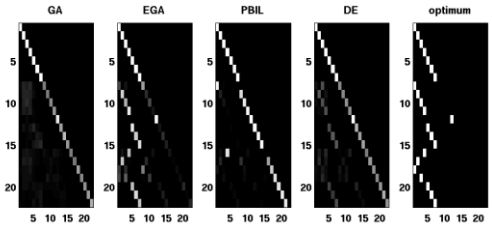

Figure 12: The average best k-matrix found at generation number 150 for each algorithm compared to the true solution.

V. TIME COMPLEXITY

It is notoriously difficult to derive any statement on the time complexity of these types of algorithms. This is because the algorithms are all stochastic in nature, and their behaviour cannot be easily described through analytical means. The results in this section aim to give an impression of the relationship between the computation time and the number of objects that are tracked.

Nine GEO objects are simulated. All the orbits are identical but shifted by 1° increments in inclination. This is to ensure that the difficulty of the problem does not change when the number of objects is increased (as can be seen when comparing the first and second test cases in the previous section). The EGA is applied 100 times for each problem size, from two to nine objects. Again, the average best fitness per generation is shown, but now all values have been normalized by dividing them by the fitness value of the optimum solution. In Figure 13 these results can be found.

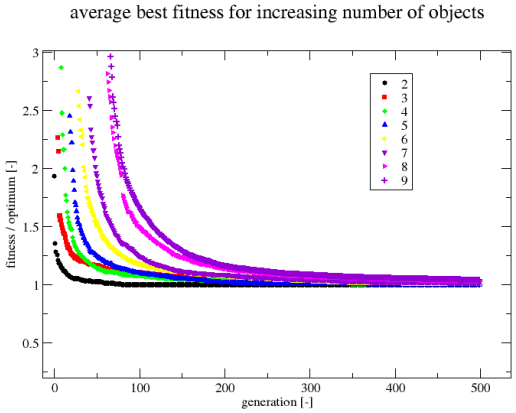

Figure 13: Average best fitness values for problems with an increase in the number of objects.

From Figure 13 the necessary number of generations needed for a certain approximation can be found. This is done by finding all the intersection points of the curves with a constant value. In Figure 14 these intersection points have been plotted for three different constant values.

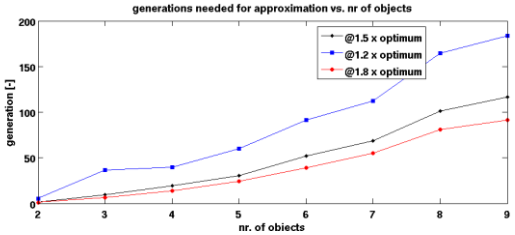

Figure 14: Scaling behaviour of the EGA algorithm when the number of objects is increased.

Recall that the population size grows with 2xN (N being the number of tracklets). Besides this the number of orbit determinations per k-matrix evaluation grows with the number of objects as well. Therefore an increase in the number of tracklets will both cause a linear increase in the population size as well as the number of orbits to be determined per individual. This $\mathcal{O}(n^2)$ cost is not taken into account in Figure 14. Therefore, what Figure 14 shows is the added computation cost on top of the $\mathcal{O}(n^2)$ that comes with increasing the number of tracklets.

From the results it is clear that the scaling of the algorithm depends on the desired quality of the approximate solution. In future research the necessary quality of the approximate solution will be a focus point. It will also be important to find an efficient way to reduce the search space for the EGA. In that way the EGA can focus on difficult scenarios such as satellite clusters and break-ups, where relatively few objects are involved.

VI. 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. The other algorithms fail to do this, however they still are able to match smaller groups of tracklets.

A study has been performed to determine the relationship between computation time and the number of tracked objects. This study has only been performed with the EGA, since this is the only algorithm capable of reliably approximating the correct solution. From the results it becomes clear that the computation costs become significant.

Future research will therefore 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 with 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.

ACKNOWLEGDEMENTS

This work is conducted within the framework of a PhD thesis which is funded by the Centre National d'Etudes Spatiales, Thales Alenia Space and the Astronomical Institute of the University of Bern.

The author's participation in the IAC conference was made possible with the support of ESA, through the International Space Education Board IAC student program.

[1] A. P. Poore, S. Gadaleta, Some Assignment Problems Arising from Multiple Target Tracking, *Mathematical and Computer Modeling*, 2006, 43

[2] S. S. Blackman, Multiple Hypothesis Tracking for Multiple Target Tracking, *IEEE AE Systems Magazine*, 2004

[3] J. M. Aristoff, J. T. Horwood, N. Singh, A. B. Poore, C. Sheaff, M. K. Jah, Multiple Hypothesis Tracking (MHT) for space surveillance: theoretical framework, *Numerica Corporation*, 2013

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

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

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

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

[8] A. J. Robertson, A Set of Greedy Randomized Adaptive Local Search Procedure (GRASP) Implementations for the Multidimensional Assignment Problem, *Computational Optimization and Applications*, 2001, 19(2)

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

[10] S. Baluja, Population Based Incremental Learning: A method for integrating Genetic Search Based Function Optimization and Competitive Learning, Carnegie Mellon University, 1994

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

[12] M. D. Schneider, Bayesian linking of geosynchronous orbital debris tracks as seen by the Large Synoptic Survey Telescopes, *Advances in Space Research*, 2012, 49(4)

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

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

[15] S. Baluja, S. Davies, Fast Probabilistic Modeling for Combinatorial Optimization, *American Association for Artificial Intelligence*, 1998

[16] P. Larranaga, R. Etxeberria, J. A . Lozano, J. M. Pena, Combinatorial Optimization by Learning and Simulation of Bayesian Networks, *Uncertainty in Artificial Intelligence Proceedings*, 2000

[17] R. S. Prado, R. C. P. Silva, F. G. Guimaraes, O. M. Neto, Using Differential Evolution for Combinatorial Optimization: A General Approach, *Instituto Federal de Minas Gerais*, 2010