POLITECNICO DI TORINO

Collegio di Ingegneria Aerospaziale

Master of Science in
Aerospace Engineering

Master of Science Thesis

Missions for Removal of Orbital Debris

Supervisor

prof. Lorenzo Casalino

Author

Adrián Tejeda

Year 2019

Abstract

During the last 40 years, the number of artificial objects in orbit increased quite steadily, leading to a big problem to be dealt at present and in near future. Most of these objects correspond to space debris in LEO which represent a latent threat to all the missions performed in such a region. Current studies suggest that to mitigate this problem as effective as possible, active debris removal (ADR) must be carried out in the following years. In this line, this strategy of debris removal needs to be done in the most efficient way, needing deep analysis of each mission. This thesis analyses the validity and versatility of an algorithm capable of finding optimal solutions for the manoeuvres involved in the active debris removal missions. Due to the great number of possibilities that the mission could have, the algorithm has been developed as a genetic algorithm. The work analyses how an ADR mission could remove target debris which correspond to some stages of the Russian rockets from the family *Cosmos*. With this approach, the algorithm was able to demonstrate its ability to find different possible sequences of debris to be deleted, not being far away from the best solution reached. The best solutions where the vehicle must remove several objects from orbit can be considered in terms of fuel consumption, in terms of mission time or taking these two parameters into consideration. In addition, it also probed its good behaviour when changing the number of space debris to be deleted, together with variations in the problem parameters. In fact, the algorithm versatility enables this tool to be used with other target debris, and hence, to be implemented in different active removal debris missions.

Sommario

Durante gli ultimi 40 anni, il numero di oggetti artificiali in orbita è aumentato abbastanza costantemente, portando ad un grosso problema da affrontare al momento e nel prossimo futuro. La maggior parte di questi oggetti corrisponde a detriti spaziali in LEO che rappresentano una minaccia latente per tutte le missioni eseguite in tale regione. Gli studi attuali suggeriscono che per mitigare questo problema nel modo più efficace possibile, la rimozione attiva dei detriti (ADR) deve essere effettuata nei prossimi anni. Secondo questa linea, la strategia di rimozione dei detriti deve essere eseguita nel modo più efficiente possibile, facendo un'analisi approfondita di ciascuna missione. Questa tesi analizza la validità e la versatilità di un algoritmo in grado di trovare soluzioni ottimali per le manovre coinvolte nelle missioni di rimozione di detriti attivi. A causa del gran numero di possibilità che la missione potrebbe avere, l'algoritmo è stato sviluppato come un algoritmo genetico. Il lavoro analizza come una missione ADR potrebbe rimuovere detriti bersaglio che corrispondono ad alcune fasi dei razzi russi dalla famiglia Cosmos. Con questo approccio, l'algoritmo è stato in grado di dimostrare la sua capacità di trovare diverse possibili sequenze di detriti da eliminare, non essendo lontano dalla migliore soluzione raggiunta. Le migliori soluzioni in cui il veicolo deve rimuovere diversi oggetti dall'orbita possono essere considerate in termini di consumo di carburante, in termini di tempo della missione o tenendo conto di entrambi questi due parametri. Inoltre, ha anche testato il suo corretto funzionamento al cambiare del numero di detriti spaziali da eliminare, insieme alle variazioni dei parametri del problema. In effetti, la versatilità dell'algoritmo consente a questo strumento di essere utilizzato con altri detriti bersaglio e, quindi, di essere implementato in diverse missioni di rimozione attiva dei detriti.

Index

List of Tables

List of Figures

CHAPTER 1

Introduction

1.1 Aims and scope

The problem of space debris surrounding Earth is a problem that is being dealt at present by the international space community. Considering that such space debris could result in collisions with future missions, these must be deleted from the Earth's orbit. One of the countermeasures that is being under study consists on the removal of some objects sending a space vehicle to do so, these are called active debris removal missions.

The objective of the present thesis is to develop an algorithm capable of finding solutions to the complex problem of which space debris to be removed. In particular, it focuses on the sequences of objects that have to be removed in active removal missions. This means, that an algorithm is developed which is able to find the best order of removal of space debris that must be carried out by the vehicle. Due to the great number of possibilities that a space vehicle can perform when removing several objects out of a large population, it is not possible to try all the existing sequences in order to achieve the best one.

In the present work only a population of debris which corresponds to the stages of the rockets within the *Cosmos* family is analysed. However, the software must be able to adapt to other debris not taking into consideration how the debris are deleted. This is, the thesis intention is not providing solutions on how the space debris are removed, but only which is the combination of debris that should be removed. Furthermore, the algorithm developed must prove to be feasible and efficient when performing its functions. In addition to this, the limitations of the algorithm and the best configuration of the code are some other main objectives, as well as to optimise the code strategies in order to reach valid solutions.

All this is tested by numerous probes performed with the developed algorithm and comparing the obtained results between them. What is more, the necessary changes are also introduced in order to cover all the main objectives of the work. While doing so, the computational time is also measured, as it is the aspect that determines the feasibility, being in fact, the main reason why this thesis has been done.

Finally, it must be said that the concerning thesis does not intend to provide a single solution for each case studied. Instead, it tries to give a wide range of valid solutions so that the user can make his choice, according to other aspects of the missions which are not treated in the present work.

1.2 Historic context

In 1957, the Soviet Union performed a mission consisting on an object of 58.5 centimetres that emitted radio pulses, its name, the *Sputnik*. One part of the rocket, the superior stage of the launcher Thor-Ablester, that had carried the Sputnik was the first body of the space debris in history. However, the accumulation of space debris the world is facing does not start with this event as it was burnt when the re-entry into the atmosphere. One year later, mission Explorer 1 of the United States add a new body to the space debris too.

Through years, space activity has experimented an exponential growth, with the direct consequence of generation of new space debris. It was not until August of 1964 that the first geostationary satellite was deployed, called Syncom-3. However, it was not until 1979 that the first explosion in GEO occurred. The following year, Lubos Perek presented some recommendations in order to mitigate this problem. These recommendations included the re-orbiting of satellites when reaching their end of life into a disposal orbit.

By the same time, LEO was the place where new technologies were tested and were most of the spatial debris were placed. Figure 1.2 shows the distribution of space debris through history, according to the orbit in which are placed. In addition, due to the dimensions of the space, it was in LEO where the problem started to worry the international community. In 1965 the Cosmos-50 satellite was exploded intentionally after a mission of critical failure. In 1968 Cosmos-249 was the first anti-satellite weapon, destroying another satellite during a rendezvous operation. In these years it was found that the main contributor to the increase of debris was the explosions of nine Delta second stages. The problem was rapidly solved, so, this could be denominated as the first implementation of space debris mitigation measures.

In July 1996, the first accidental collision between two catalogued objects was recorded, corresponding to the collision between the Cerise satellite and a fragment of the Ariane orbital stage exploded in 1986. Since 2001 the ISS has been permanently manned, with space debris having played important roles in the design of the different modules of the station. They are protected by debris and meteoroid protection shielded, being able to stand after impacts with bodies of 1 cm. After an accident that took place in 1986, a flight rule was established for Space Shuttle operations, indeed, it was a procedure for collision avoidance of trackable space objects. In fact, at present, the ISS performs one avoidance manoeuvre per year.

Of the entire population of tracked space debris, the 66% have decayed, being burned due to re-entry. Nevertheless, in 1964, after a launcher failure, the Transit 5BN-3 satellite re-entered above the Indian Ocean, being the first risk object re-entry. Due to the possibility of these type of events, the preparation of the “UN Principles Relevant to the Use of Nuclear Power Sources in Space” was carried out. There were also some other objects that represented danger due to the dimensions of these ones. Figure 1.1 represents the number of objects that have being re-entering in Earth’s atmosphere, most of them being burned.

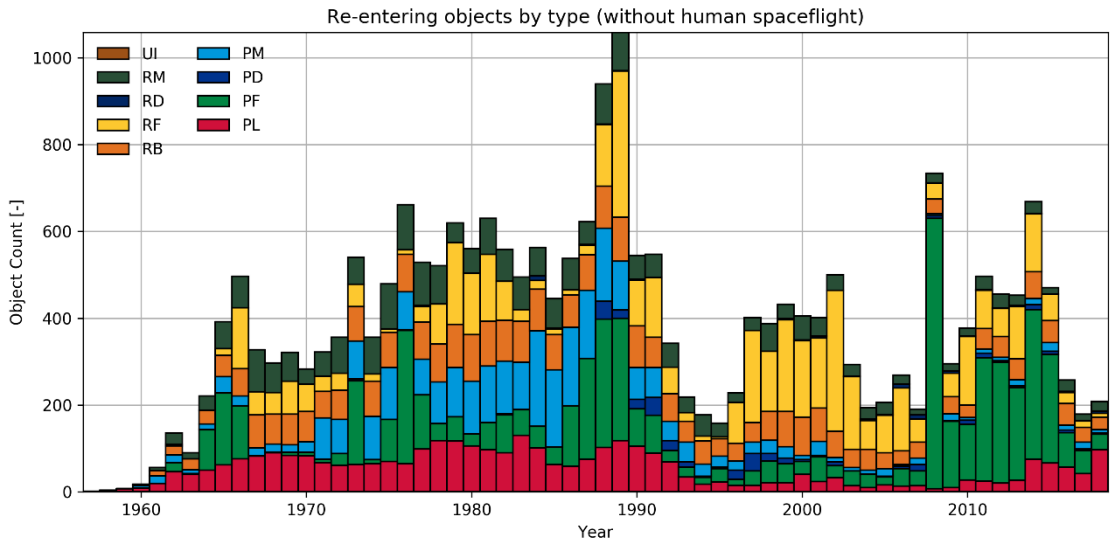

Figure 1.1 – Debris distribution through history, classified by types

All this said, the concern about this real problem has grown, and therefore, the number of accidents and of generation of debris is much lower than before. Nonetheless, all the debris generated through history are still in orbit, generating much more debris. That is the reason why not only countermeasures of preventions but of active debris removal are being under study. Thus, the last ten fragmentation events (2018) can be seen in table 1.1. This number is low in comparison to the number of events in previous years when space debris was not considered to be such a big deal. Moreover, the number of fragments created is not very big, thanks to the new technologies and techniques used for this reason.

Name	Int. Designator	Break up epoch	Number of fragments
Centaur-5 SEC (Atlas V 401)	2014-055B	30 Aug 2018	0
Proton-K/DM-2 ullage motor (SOZ)	2005-050F	24 Aug 2018	0
L-14B-res (YF40B-res)	2013-065B	17 Aug 2018	4
Proton-M/DM-2 ullage motor (SOZ)	2010-007H	22 May 2018	3
Titan Transtage	1969-013B	28 Feb 2018	18
Fregat SBB	2017-086C	12 Feb 2018	0
Proton-M/DM-2 ullage motor (SOZ)	2010-041G	03 Sep 2017	9
Telkom 1	1999-042A	25 Aug 2017	0
AMC 9 (GE 12)	2003-024A	01 Jul 2017	0
AMC 9 (GE 12)	2003-024A	17 Jun 2017	0

Table 1.1 – Latest fragmentation events [21]

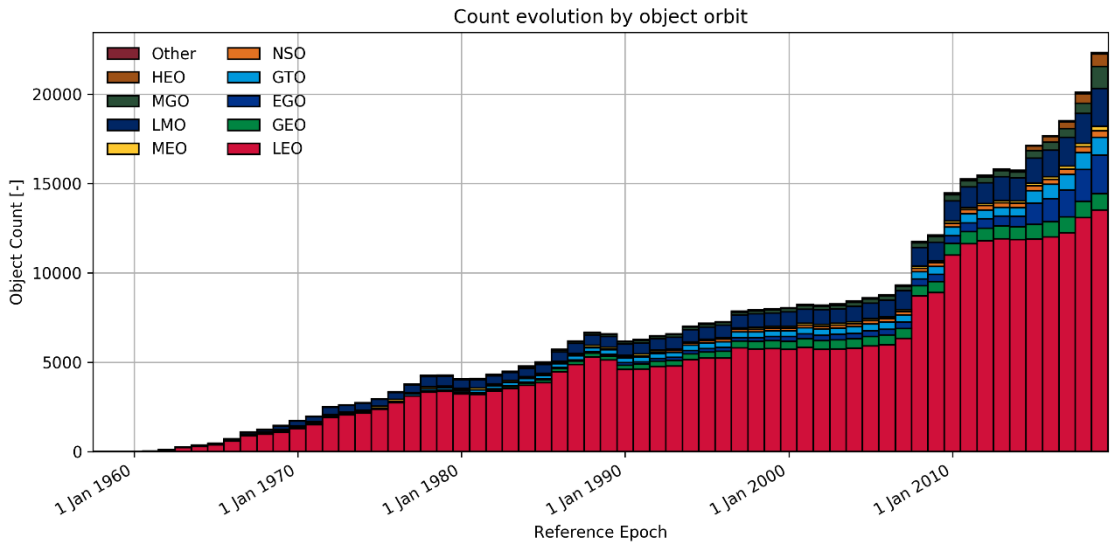

Figure 1.2 – Debris distribution through history, classified in orbit regimes

The above figure shows how the most critical orbits are LEO orbits, where most of the satellites are launched. Furthermore, there are two important events that can be appreciated in the graph. One, creating 3443 fragments in the year 2007, when an old Chinese meteorological spacecraft called Fengyuun-1C, was hit and destroyed by a ground-to-space missile during an anti-satellite test. While the accident of the Cosmos-2251 in 2009 created 1667 fragments. This satellite collided with the U.S. commercial communications satellite Iridium. Except for these ones, it can be appreciated that debris, at least in LEO, increase slowly nowadays.

1.3 Spatial debris at present

At present, January of 2019, there are more than 84000 tonnes of mass of all space objects in Earth orbit. This number correspond to estimations by statistical models which suggest: 34000 objects greater than 10 cm, 900000 objects of 1 cm to 10 cm and 128 million objects from 1 mm to 1 cm. From all these objects only 22300 number of debris objects are regularly tracked by Space Surveillance Networks and maintained in their catalogue. What is more, according to ESA's estimation the number of break-ups, explosions, collisions, or anomalous events resulting in fragmentation if greater than 500.

Since the start of the spatial activities, 5450 rocket launches have been performed, placing into Earth orbit 8950 satellites. From these satellites, 5000 satellites are still in space while only 1950 are still functioning, being these a source of debris. These big bodies are tracked in order to avoid them in future missions due to the high danger that a possible collision represents. The previous given numbers are referred to *Jan 2019*, while figure 1.3 represents rocket bodies and inactive satellites in *July 2012*. That is the reason why the numbers could not match.

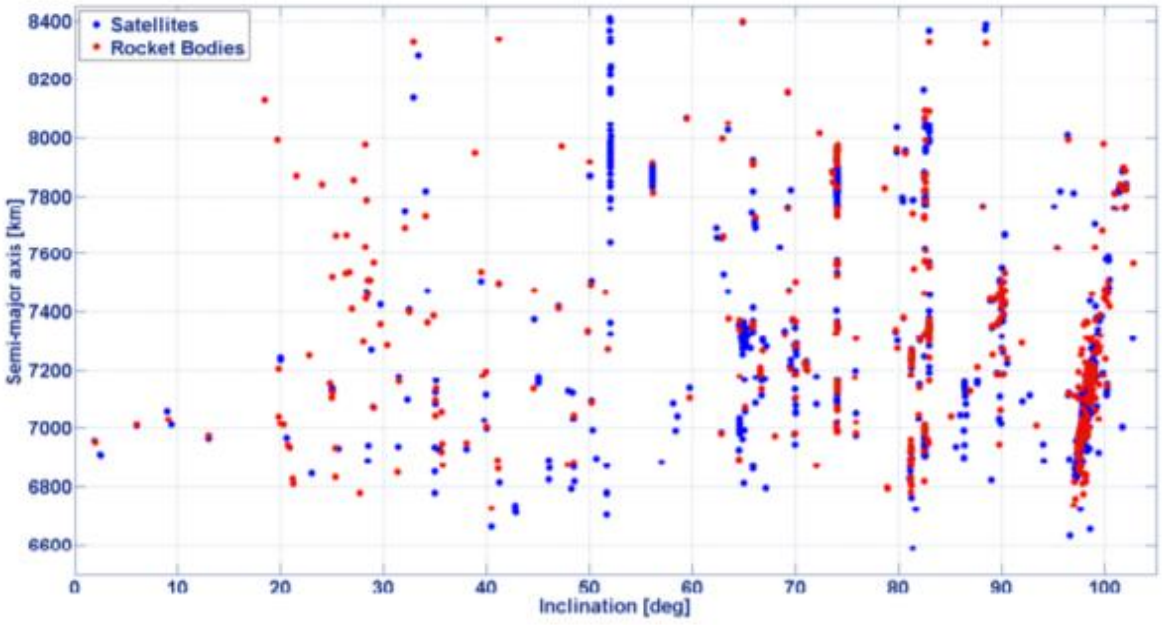

Figure 1.3 – Distribution of inactive satellites and rocket bodies residing in LEO as a function of semi-major axis and inclination (19 July 2012)

As it can be seen, there are some inclinations in which there is a concentration of spatial debris of this type. This happens due to two different reasons: For the satellites (represented in blue) there are orbits that are more used than other for certain missions, therefore when they become inoperative, they are still in these inclinations of interest. The same happens with the altitude of the orbits, where there is also concentration at the altitudes of interest. These reasons also apply to the rocket bodies, where the injection is strongly related to the orbit wanted for the satellite. In fact, it can be seen a concentration of debris at the inclination of 82 degrees most of whom correspond to the *Cosmos* family, debris involved in this work.

Now, the most important events of creation of spatial debris are ranked. It must be outlined that the ranking's criteria are the number of fragments created, not the dimensions or danger of these ones. Furthermore, this ranking is referred to the date: *20-Dec-2018 14:47 UTC*.

Rank	Name	Int. Designator	Breakup epoch	Fragments
1	Fengyun 1C	1999-025[a]	11 Jan 2007	3443
2	Cosmos-2251	1993-036[a]	10 Feb 2009	1667
3	HAPS	1994-029B	03 Jun 1996	753
4	Iridium 33	1997-051C	10 Feb 2009	627
5	Cosmos-2421	2006-026[a]	14 Mar 2008	509
6	H8	1986-019C	13 Nov 1986	497
7	Cosmos-1275	1981-053[a]	24 Jul 1981	478
8	NOAA 16	2000-055[a]	25 Nov 2015	458
9	Agena D	1970-025C	17 Oct 1970	375
10	PSLV-CA fourth stage	2001-049D	19 Dec 2001	371

Table 1.2 – Top10 fragmentation numbers by number of fragments [21]

On the other hand, table 1.3 shows the most important events in terms of number of space debris which are in orbit the 20th of December of 2018.

Rank	Name	Int. Designator	Breakup epoch	Fragments
1	Fengyun 1C	1999-025A	11 Jan 2007	2830
2	Cosmos-2251	1993-036A	10 Feb 2009	1075
3	NOAA 16	2000-055A	25 Nov 2015	457
4	Cosmos-1275	1981-053A	24 Jul 1981	419
5	Iridium 33	1997-051C	10 Feb 2009	329
6	Agena D	1970-025C	17 Oct 1970	234
7	DMSP Block 5D-2 F13	1995-015A	03 Feb 2015	218
8	DELTA P	1975-052B	01 May 1991	197
9	DSV-3H-4	1973-086B	28 Dec 1973	197
10	Zenit-2 second stage	1992-093B	26 Dec 1992	196

Table 1.3 – Top10 fragmentation numbers by number objects in orbit [21]

1.3.1 Collision probability

The probability of collision with a spatial object depends on the dimensions and velocity of the debris. Oftenly, space vehicles and satellites are impacted with debris up to a millimetre as these ones are impossible to be tracked and consequently avoided. However, the dimensions of these debris are not enough to cause significant damage on the vehicles.

While for debris which are not trackable but could cause a significant dame on the satellites, some countermeasures are taken. One of these consists on orientating the satellite in an appropriate way, so the damage caused by the debris is not so harmful as could be. Another one is to shield the vehicle correctly, with different materials and layers.

For the calculation of the probability of collision between an operative satellite and a debris, *NASA* uses various approach, from which one is presented hereafter.

$$P_C = \frac{1}{\sqrt{(2\pi)^2|C^*|}} \iint exp\left(-\frac{1}{2}\vec{r}^T C^{*-1}\vec{r}\right) dXdZ \tag{1.1}$$

Where C^* is the projected covariance, and P_C is the probability of collision which is also defined as the portion of density that falls in the control volume of the satellite. Furthermore, it must be said that this results only provides a nominal estimation.

The US Space Surveillance Network can track individually objects whose dimensions are superior to 10 centimetres. Hence, the DOD informs the agencies if there is a vehicle that is in danger of collision with trackable objects. A collision is termed catastrophic if it results in the complete fragmentation of impactor and target. NASA estimates this with the following empirical condition:

$$0.5 * \frac{M_{imp} {V_{imp}}^2}{M_{targ}} > 40\, J/g \tag{1.2}$$

1.3.2 Expected future

At present, there is a common concern about this growing issue, however, efforts to solve such a big deal are just beginning. In figure 1.4 three different possible futures are presented, all of them

if this problem is not faced. As it can be seen, even in the best case where there are no more launches, the cumulative number of collisions grows, as there would still be numerous debris from previous missions that evolve in other debris and provoke other collisions. Apart from that, if space industry continues with the usual activity, it is predicted to be an exponential growth of collisions and space debris. This could lead to a future where no more launches could be performed due to the lack of free-danger space available.

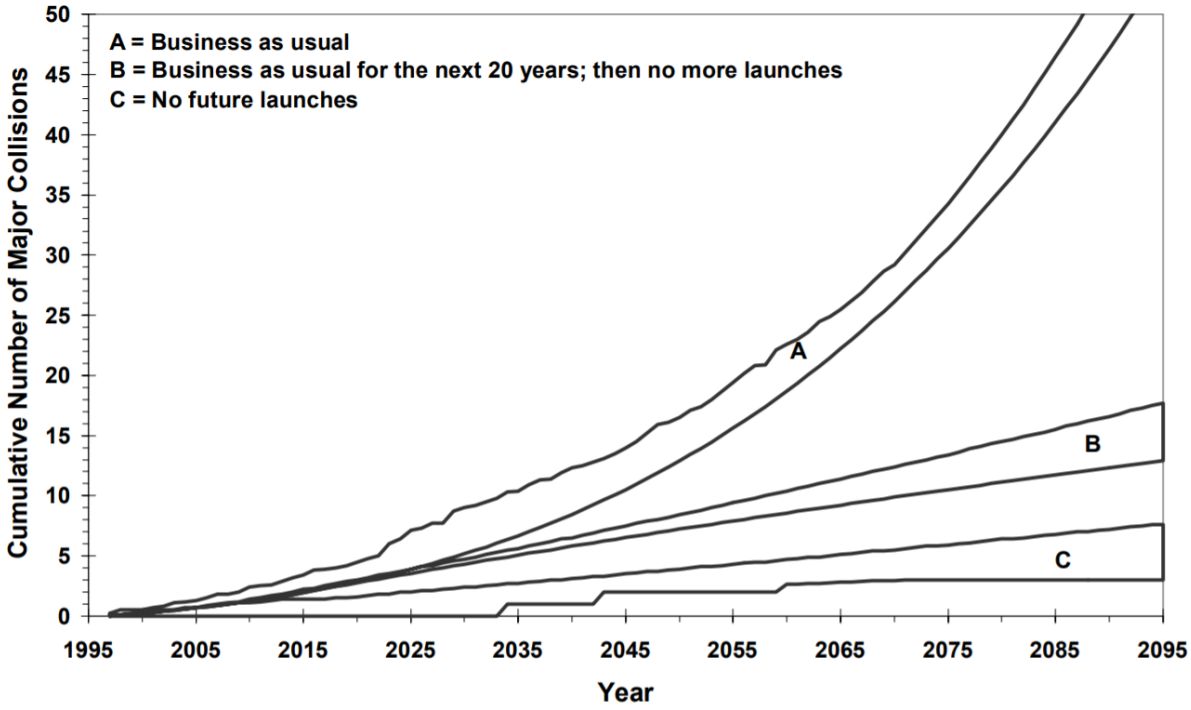

Figure 1.4 – Projected collisions among catalogued numbers [23]

However, as it will be later explained in this chapter, agencies and companies from all over the world are starting to purpose some initiatives in order to reduce all this said. Despite at present being in early stages, these projects objective is to reduce the population of space debris, so the graph previously presented could be changed.

1.4 Classifications

Space debris can be classified according to very different criteria, not always being clear the differentiation between the categories. In this section some categories are stablished, and lastly the format TLE is introduced, that, despite not being a classification, is a way of denominating the space debris and objects around Earth.

1.4.1 Origin of the debris

Space debris is defined as all man-made objects including fragments and elements thereof, in Earth orbit or re-entering the atmosphere, that are non-functional (*IADC* definition).

Three countries in particular are responsible for roughly 95% of the fragmentation debris currently in Earth's orbit: China (42%), the United States (27.5%), and Russia (25.5%). The sources of the human made debris are: Inoperative satellites, tools and objects from missions and fragmentation. The space debris can be classified into different groups according to what is the cause that had created them.

The event causes of the debris formation, are the following:

- **Collision:** Collision between catalogued objects, until now only 4 events.
- **Small impact:** Small fragments of debris or micro-meteoroid that impact to a catalogued object.
- **Propulsion:** Stored energy for non-passivated propulsion-related subsystems might lead to an explosion.
- **Electrical:** Stored energy for non-passivated batteries might lead to an explosion.
- **Aerodynamics:** The reason of the break up is the interaction with the atmosphere.
- **Accidental:** Subsystems which showed design flaws ultimately leading to breakups in some cases.
- **Deliberate:** Satellites that were deliberately destroyed for several reasons.
- **Anomalous:** Normally happens at low velocities, not expected separation of one or various detectable objects from a satellite that remains essentially intact.
- **Unknown:** As the name indicates, the causes are unknown, this is, evidence is lacking to support any of the other categories.

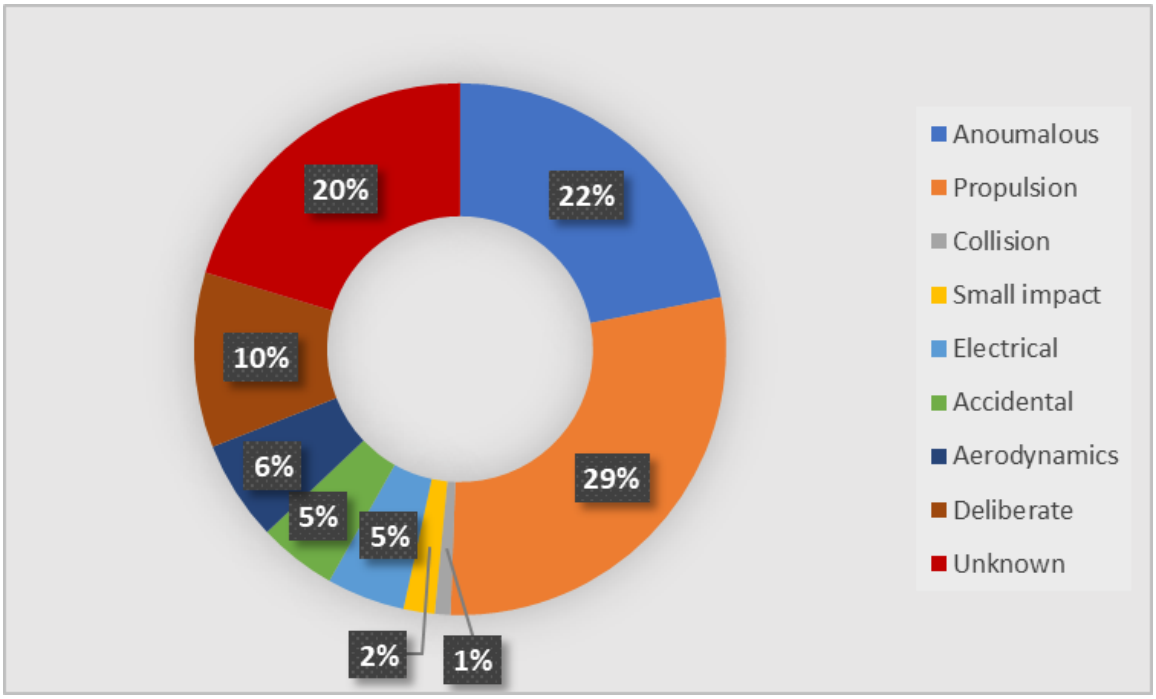

Figure 1.5 – Event causes and their relative share for all past fragmentation events [21]

It must be mentioned, that the data that appears in this subsection comes from ESA's *DISCOS* and represents a snapshot from: 20-Dec-2018 14:47 UTC. More information can be found in [21].

1.4.2 Taxonomic acronym

The taxonomy described in this section consists of two different parts, one for the debris class and the other one for the debris hazard. Thanks to this acronym (see figure 1.6), it is easy seen at first glance the main characteristics of the debris, and hence, easier to choose which is the ADR mission that suit most to deorbit them.

The first part of the acronym is defined as the debris class, where the most prominent physical and dynamical properties of the objects are presented. Each letter refers to a specific characteristic:

- **Orbital state:** Capability of an object to control its orbital position. C for controlled and U for uncontrolled.
- **Attitude state:** Capability of an object to control its attitude. S for actively stabilized, R for regular rotating and T for tumbling.
- **External shape:** Description of the shape of the body. X for convex shapes, P for regular polyhedral and I for irregular shape.
- **Overall size:** Mean size of an object. S for small objects (< 10cm), M for medium (< 1m) and L for large objects (> 1m)
- **Area-to-mass ration (AMR):** Used as provides more information than the mass parameter. lo for low (< 0.8m^2/kg), me for medium (< 2m^2/kg) and hi for high (> 2m^2/kg).

Only analysing this part of the acronym it is possible to draw some conclusions about the type of ADR missions that adapt best with the needs. However, some other characteristics are needed, that is why the second part of the acronym gives more information. Nevertheless, it is enough for a more programmatic classification of space debris and can be used to filter classes that are non-economically viable or practical for ADR.

The second stage is named as debris hazard. It consists just in two different aspects, the break-up risk index and the level of non-cooperativeness. Both are complex indexes which are briefly described in the present work, however, if a deeper knowledge is wanted more detailed information can be found in [24].

- **Break-up risk index:** Defined by calculating its criticality number (CN) as a product between severity number (SN) and probability number (PN). The two numbers definitions can be found in the ESA's standard [25]. It must be highlighted that methods such as robotic/tether-based are suitable for CN<4 while for CN>4 ne/contactless methods are the most appropriate.
- **Level of non-cooperativeness:** Difficulty that a capture manoeuvre is likely to face due to the angular rate, berthing feature existence, material properties and mechanical clearance of a capturing interface of a target. Therefore, 14 levels are considered in table 1.3. The levels are expressed as a combination of an Arabic numeral (from 1 to 7, with 1 being the least non-cooperative and 7 the most non-cooperative level) and a letter indicating the dimensions of the mechanical clearance of the capturing interface (large (L) or small (S))

Levels	Rate			Capture interface & ADR association					
				Berth		Material		ADR	
	Low	Med	High	Y	N	Iso	An	L	S
1	X			X		X		Manipulator	
2	X				X	X		Clamp w sync. /Tether	Tether
3	X				X		X	Clamp w sync. /Net	Net
4		X		X		X		Manipulator w sync.	
5		X			X	X		Clamp w sync. /Tether	Tether
6		X			X		X	Clamp w sync. /Net	Net
7			X					Contactless	

Table 1.4 – Levels of non-cooperativeness of a target

Two other higher-levels characteristics are also used. Firstly, it is distinguished whether the object is man-made or natural, the object type. The second one is the orbit type, this is the altitude of the orbit. They are classified into:

- **LEO**: 80-2000 km.
- **MEO**: 2000-35786 km.
- **GEO**: 35786 km
- **HEO**: beyond 35786 km

Now, as an example, a Cosmos 3M vector is presented following the explained taxonomic acronym. This debris is closely related to the present work, as it is part of the Cosmos family.

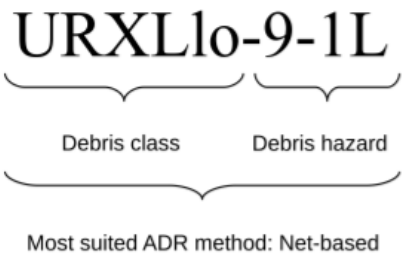

Figure 1.6 – Example of the taxonomy application to the 1967-045B Cosmos-3M 2nd stage

Debris class		Debris hazard	
U	Uncontrolled	**9**	Risk of breakup
R	Regular rotation	**1L**	Difficulty of approach
X	Regular convex		
L	Large		
lo	Low are-to-mass ration		

Table 1.5 – Cosmos 3M 2nd stage taxonomic acronym

To conclude the discussion on the methods of classification of space debris it must be outlined that what has been said is only an overview of the techniques of possible analysis. The choice of the most suitable mission profile and the system of capturing a debris is essential to increase the probability of success of a hypothetical ADR mission. In the case examined in this thesis, where it is necessary to remove four, five or eight different objects with a single aircraft chaser, making the most correct choices could allow you to extend the capabilities of the spacecraft and maximize the results.

1.4.3 Two Line Elements (TLE) Format

This is a data format into a list, which contains the main characteristics of the given object. It is used only for Earth-orbiting object for a given point in time, the epoch. Therefore, if the evolution of the object needs to be obtained, a propagation model must be used. Any algorithm using a TLE as a data source must implement one of the S*GP* models to correctly compute the state at a time of interest. Another variant of this format is the *Three Line Elements*, which is exactly the same but adding another line for the name of the mission. A catalog of all the known objects orbiting Earth can be found in [19], and in fact, is what has been used in this thesis as inputs for the concerning debris.

```
ISS (ZARYA)
1 25544U 98067A   04236.56031392  .00020137  00000-0  16538-3 0  9993
2 25544  51.6335 344.7760 0007976 126.2523 325.9359 15.70406856328906
------------------------------------------------------------------
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890   reference number line
        1         2         3         4         5         6         7
```

Figure 1.7 – Two-line element set example

	Column	Description	Example
Line 1	1	Line Number	1
	3-7	Catalog Number	25544
	8	Elset Classification	U
	10-17	International Designator	98067A
	19-32	Epoch (UTC)	04236.56031392
	34-43	1st derivative of Mean Motion with respect to Time	0.00020137
	45-52	2nd derivative of Mean Motion with respect to Time	00000-0
	54-61	B* Drag Term	16538-3
	63	Element Set Type	0
	65-68	Element Number	999
	69	Checksum	3
Line 2	1	Line Number	2
	3-7	Catalog Number	25544
	9-16	Orbit Inclination (deg)	51.6335
	18-25	Right Ascension of Ascending Node (deg)	344.7760
	27-33	Eccentricity	0007976
	35-42	Argument of Perigee (deg)	126.2523
	44-51	Mean Anomaly (deg)	325.9359
	53-63	Mean Motion (rev/day)	15.70406856
	64-68	Revolution Number at Epoch	32890
	69	Checksum	6

Table 1.6 – Two Line Element description

1.5 Technology under study

According to various experts, the different technologies to deorbit debris are shown in the graph of figure 1.8. Furthermore, some characteristics are also presented, nevertheless, most of them are a mere estimation as they have not been implemented yet. In this section some of the most importants are presented with a brief description.

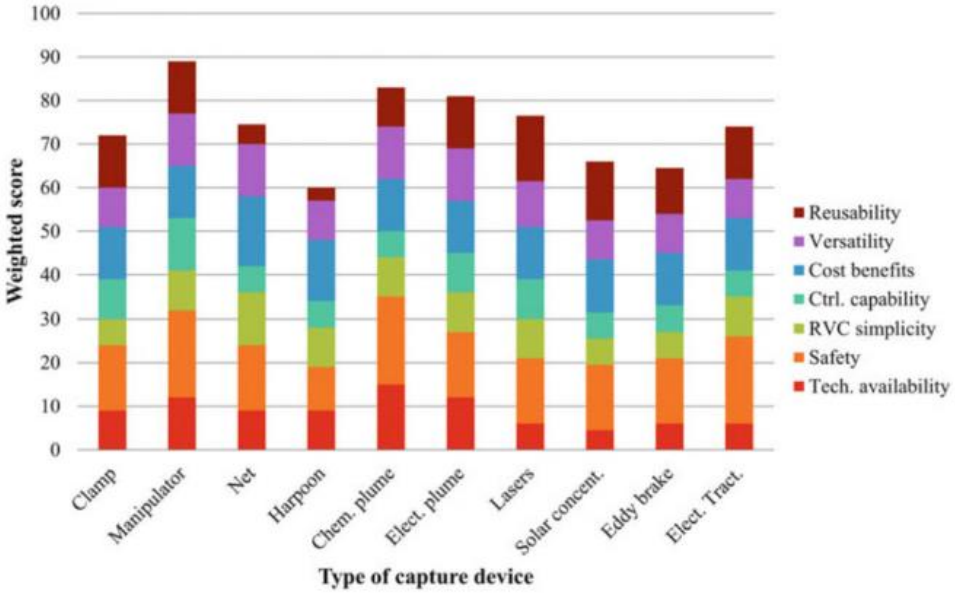

Figure 1.8 – Technologies for removal of space debris

1.5.1 Mechanic capture

The main idea of this active strategy of removal of debris is to capture somehow an object with a chaser, and afterwards carrying it to an orbit that will lead in a re-entry. Furthermore, another variation is the use of a vehicle capable of achieving various space debris to which attach devices for de-orbiting. This mission architecture is suitable and feasible for objects between 600 and 2000 kilometres of altitude. It must be said that this technique has not been used yet, having been performed several simulations by different entities.

Firstly, for the capture phase, different devices are under study. A robotic arm could be used to grab the debris. A net has also been considered to do so, as the impact of the debris on it would have as consequence less structural problems. A simpler alternative to nets is to shoot at close range on the target some kind of resin or sticky foam. Once attacked, it would expand generating more friction. The usefulness of this method would be confined only to satellites in low and small orbit. For larger, heavier and larger objects at high altitude it is not effective. Moreover, one could also use a system capable of launching a harpoon to capture the target. This would allow a distance to be maintained superior between chaser and debris and avoid collisions. More than just a single harpoon can be used in order to increase the number of debris deleted in each mission and time.

While for the phase of deorbiting, different strategies could be used. In fact, most of the bellow presented devices could be used in combination with this technique. Apart from them, for dead or non-functioning satellites, the chaser could provide the necessary resources to make them function, such as: batteries, communication systems…

1.5.2 Drag enhancement devices

Being part of the passive devices, these are at first glance the most economic ones. The objective of them is to increase the aerodynamic drag in order to reduce the height, or the perigee, of the orbit. With this, the object will end in a re-entry. So, if this strategy is used on large objects, the pertinent calculations must be done in order to achieve a controlled re-entry. Due to the importance of the presence of atmosphere, this technique is particularly efficient for low orbits, less than 800 *km* of altitude. Another important point to be highlighted is that as the device must be installed on the debris, they must be reached somehow.

1.5.3 Solar sails

Just as the drag enhancement devices, is a passive strategy. Using the solar radiation pressure, a thrust is achieved in a determined direction. So, with this thrust the orbit of the object can be changed, leading to a lower perigee or altitude. Solar photons transfer their momentum to the sail by impinging on its surface and the efficiency of the momentum exchange process increases with the reflectivity of the material. Nevertheless, if the correct calculations are not done, the decay effect within an orbit can be null. Although an orientable solar sail might seem the solution, it must be outlined that the cost of it could increase greatly. Furthermore, solar sails do not function well below 600-800 km because of the oxygen erosion and residual air drag. Hence, these ones can be used to obtain a lower orbit and afterwards implement other methods such as drag enhancement devices.

1.5.4 Electro-dynamic or momentum exchange tethers

Electrodynamic tethers (EDTs) belong to the passive group of devices. These are deployed from the defunct satellite along the local vertical at the end of its life. Electrons from the ionospheric plasma are collected thanks to an anodic device, Therefore, the operative range of this device is up to 1000 km of altitude. The mentioned collected electrons are returned back to the ionosphere by plasma contactor (cathode), thus an electric current is made flow inside the tether. This in turn interacts with the Earth magnetic field and, according to Lorentz law, produces a drag force that

make the entire object deorbit. The principal advantage of electrodynamic tethers is that, once deployed, it is completely passive and can deorbit objects from whatever inclination and altitudes up to 1000 km, and more, in relatively short time. However, this method needs to be installed on the object to be deorbited, hence, is most suitable for end life deorbit.

1.5.5 Magnetic tugs

This complex system consists on a chaser/tug satellite, equipped with thrusters and a powerful steerable magnetic dipole. This one, creates a magnetic field that attracts the dipole carried by the target object. So, the aim is to change the orbit thanks to such magnetic force. Indeed, when referring to Low Earth Orbits, the magnetic field of the Earth must be considered, as it interacts with the magnetic system. One of the main issues of this method is the position of the magnetic torque rod of the target which may be not located at its centre of mass. Furthermore, this strategy is useful only in dead satellites or debris with suitable magnetic properties. Nonetheless, the advantage of this method is that can be used to remove several objects in a single mission.

1.5.6 Lasers

There are drawbacks of a ground-based laser system in cleaning space debris. Therefore, the placement of a laser system in space is under current investigation. The main advantage of this strategy is that there is no necessity of a rendezvous with the debris. Two different techniques can be distinguished: direct ablation mode and ablation jet mode. The first one consists on deleting directly the debris by burning them. Due to the high energy required to do so, this mode is aimed to be used for tiny debris particles. On the other hand, the jet mode consists on use the laser to create a propulsive force in the debris which decreases the altitude of the orbit until reentry. There is no need of burning down the entire object, hece, it can be used in greater objects. The process to do such a thing is the following:

1. High laser intensity at the debris surface induces ablation
2. Ablated material is vaporized and ionized (plasma)
3. The plasma expands in a velocity much higher than that of sound when the temperature rises to the vaporization point
4. The reaction force modifies the trajectory

An example of a laser system is presented in table 1.6.

Assumption		Laser parameters	
Orbital heigh of laser	420 km	Type	Solid state
Orbital heigh of space station	400 km	Pulse energy	1 kJ
Operating distance	100 km	Repetition frequency	100 Hz
Debris size	1-10 cm	Mirror diameter	2.44 m
Debris mass	< 70 g	Power density	$1.39 \times 10^9 W/cm^2$

Table 1.7 – Laser system for removal of debris

1.6 Initiatives

Since the seminal work leaded by Donald Kessler in the 1970s on the artificial debris exponential growth, the publication of the position paper of the American Institute of Aeronautics and Astronautics (AIAA) in 1981, the release of the report of the European Space Agency Space Debris Working Group in 1988 , the publication of the report of the Scientific and Technical Subcommittee of the United Nations Committee on the Peaceful Uses of Outer Space (UNCOPUOS) in 1999 , and the issuing of the position paper of the Space Debris Subcommittee of the International Academy of Astronautics (IAA) in 20015 , the international space community became progressively aware of the increasing relevance of the orbital debris problem, in order to keep the Earth surrounding available for future missions.

Orbital debris is not addressed explicitly in current international law. International agreements that directly address orbital debris, however, may eventually be needed to deal with several debris-related issues. Indeed, there are some debris space laws not having reached an international agreement yet. However, as the problem is a real fact, there are several initiatives by most of the agencies of the world in order to deal with such a delicate problem.

1.6.1 ESA's Clean Space

This program is born with the idea of achieving a space without debris and with a lower risk of collision with them. Since 2012, ESA's Clean Space initiative has been always considering the entire life-cycle of space activities, from the early stages of the design until the removal of space debris. To do so, the program has three branches that some of them will be later explained deeper. These are: EcoDesign, CleanSat and eDeorbit.

1.6.2 CleanSat

This project, part of the Clean Space program, was presented in March of the year 2015 and is still under study and being developed, for the mitigation, and not removal, of spatial debris. This one is a project with the objective of developing the necessary technologies to support the compliance of future satellites with Space Debris Mitigation requirements. So, the aim is to reduce the number of satellites being abandoned, decrease the risk of orbital collisions from increasing debris as well as reduce the threat of re-entering satellites. Indeed, it is focused in the evolution of LEO objects. This international program can be resumed in 4 different points:

- Pursues the development of technologies for Space Debris Mitigation.
- The project involves several manufacturers, including ESA which is the coordinator.
- Focused in 3 different points: Deorbiting systems, design for demise and passivation.
- Estimated budget required is of 31.5 M€.

According to this program, within 25 years from the end of life of a satellite, this one has to be out of LEO or GEO protected regions. Moreover, equipment from the small satellites shall be redesigned in order to be burnt in an uncontrolled re-entry so not a single piece reach the Earth's surface. Lastly, the passivation of systems such as propulsion and power system must be carried out, so that explosions are prevented.

It must be reminded, that this project is a debris mitigation project, this is, it does not have as an objective the removal of the current space debris but the avoidance of creating new debris.

1.6.3 eDeorbit

This one, also part of the Clean Space program, is a project for active removal of space debris. The development of this project began in 2013, although the studies for it date from 2009, having as target the Envisat Earth-observing satellite, which failed in 2012. The final mission approval took place in 2016, planning to be launched in 2023 on board a Vega launch vehicle.

At present, the idea of the removal is that a spacecraft will be launched into a polar orbit with an altitude of 800 to 1000 *km*. After being in orbit, this one will rendezvous with the target satellite. Once this done, the capture of the satellite will be done in one of two different ways under study: with a mechanical tentacle or a net. The first option includes a robotic arm that will hold the satellite from a point, and afterwards, the other robotic arms will proceed to clamp it. While the second option will envelop the target derelict before the spacecraft will begin to change its orbit. Finally, the spacecraft will perform an atmospheric re-entry carrying the target with it without causing any hazard to other space missions or populations on Earth. One of the greatest challenges of this project is that on operative satellites tend to start tumbling, which must be modelled and studied in order to perform the correct capture.

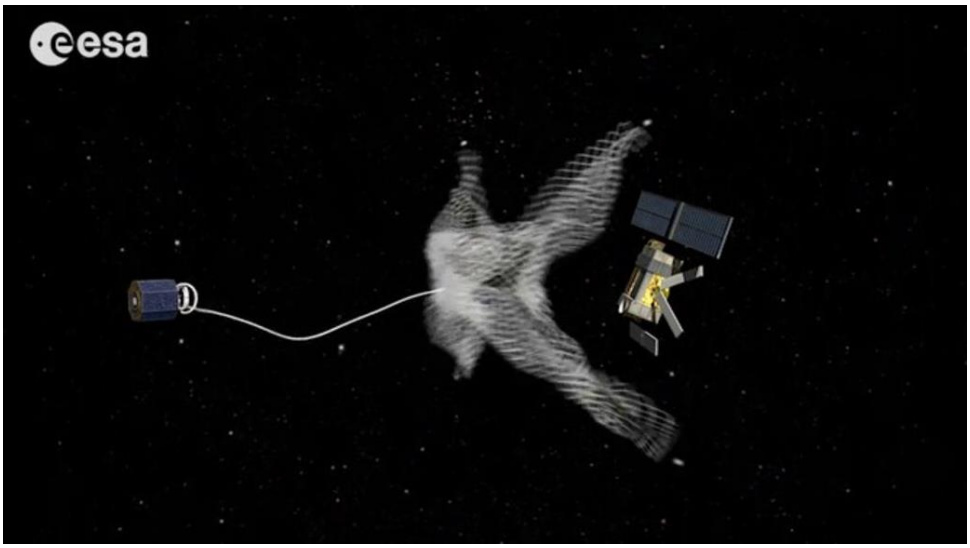

Figure 1.9 – Concept of eDeorbit with net capture

1.6.4 RemoveDebris

This aim of this project is to demonstrate various space debris removal technologies in LEO. Such an experiment is carried out by the company Surrey Satellite Technology using a rocket Falcon 9 FT for his launch the 2 of April of 2018 in Cape Canaveral.

Two different technologies are tested: capture with a net and capture with a harpoon. In both cases the principal satellite (*RemoveDebris*) releases target satellites (*DebrisSat 1* and *DebrisSat 2),* and afterwards the corresponding technology of capture is tested. While doing so, the tests are recorded through video camera, intending to understand how these capture mechanic systems work without gravity, as it is very difficult to simulate such a thing on Earth. Once the target debris are captured, a large sail is deployed in order to make the objects re-entry due to the presence of gases of the atmosphere in LEO.

The 16 of September of 2018, the net test was performed. Whilst on the 8th of February of 2019, the harpoon experiment was performed. These events made this mission to be the first time that ADR technologies are tested in space.

1.6.5 SpaDe

This project, still in phase 1, consists on the demonstration of space debris removal technology. This one, due to its early stage, cannot give all the details of the project yet. The project is being under study by *Raytheon BBN Technologies* in collaboration with *NASA*.

SpaDe will use pulses of atmospheric gases to accelerate the rate of decay on debris by creating a temporary drag that causes the re-entry into the atmosphere sooner than would naturally occur. These pulses do not affect LEO satellites, soon falling back into the atmosphere and being burned. In this case, the dimensions of the target debris are small, centimetres. In contrast to other proposed methods, a big advantage is that if an accident occur, SpaDe does not create new space debris. SpaDE should provide a lower cost alternative to remove a orbiting system of numerous space debris.

1.6.6 NASA's Robotic Refuelling Mission (RRM)

Is an external International Space Station investigation designed to demonstrate and test the tools, technologies and techniques needed to robotically refuel and repair satellites in space, especially satellites that were not designed to be serviced. Is a project of *NASA* in collaboration with the

Canadian Space Agency. The beginning of the development dates of 2009 with the development of the RRM module at the *ISS*.

The aim of this project is to fix in orbit all the problems of inoperative satellites, so that they can still be useful, or at least they are able to re-entre in an appropriate way. This project involves tasks such as gas fitting, refuelling, thermal blanket manipulation, screw/unscrew or removal of electrical caps. Furthermore, as the experiments are performed at the *ISS*, the mission cost is low in comparison with other already described projects

CHAPTER 2

The Mission

2.1 Description of the mission

The mission consists on the removal of a certain quantity of spatial debris. This is, a chaser that can reach several debris in order to install a deorbit tool so they entry in the atmosphere and are eliminated. In particular, the debris which are under study, are some debris from cosmos missions. Kosmos-3M (Cosmos) is a Russian space launch vehicle, member of the Kosmos family. This launcher was able of putting in orbit 1500 *kg* in LEO, which is the type of orbit that concerns this work. This launcher has been used in numerous launches, from which some space debris are orbiting around Earth at present.

As consequence, two different clusters of debris can be distinguished, one of 120 objects with an inclination of 74 degrees, and another one of 155 objects with inclination of 82 degrees. The objects orbit at different altitudes from each other, fact that is crucial as will be explained in further sections.

The orbital elements of each debris were found in the NO-RAD catalogue, which can be consulted in [18]. These data are provided in two-line element set (TLE), which was explained in chapter 1, and in a version of this one, three-line element set (3LE). Once known all the orbital elements of the debris, they are propagated using a SGP4 model, which is a simplified model. Therefore, introduces an error around 1-3 *km* each day. So, the results vary depending on the date that the mission is performed.

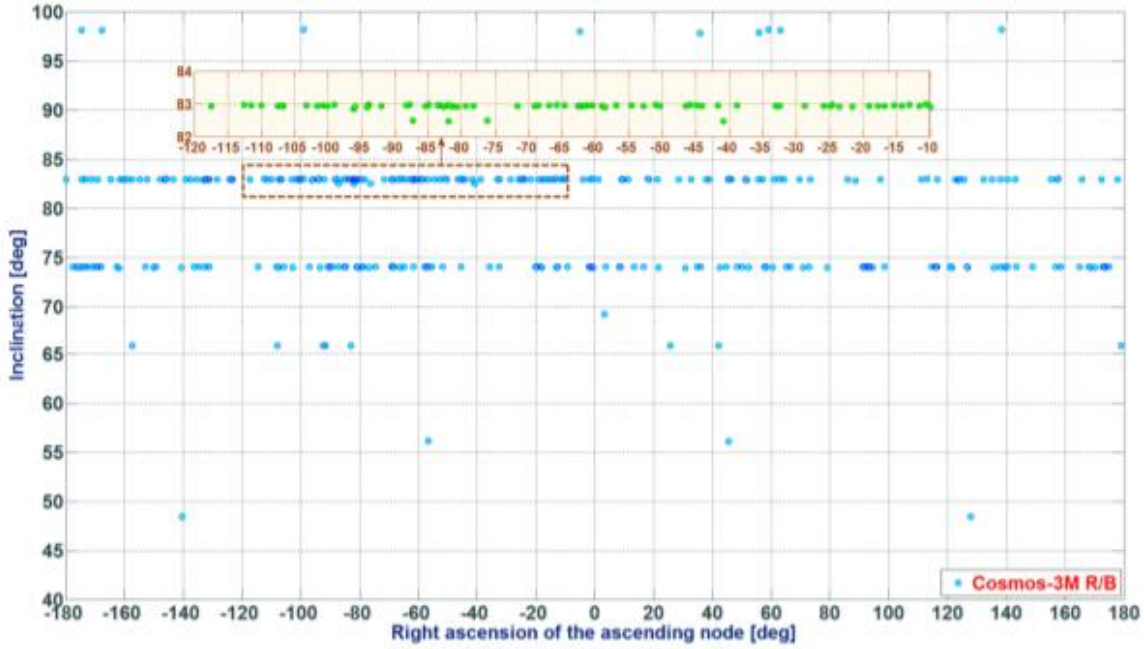

Figure 2.1 – Dispersion of Cosmos3M rocket bodies in LEO (19 July 2012)

The cluster that will be studied is the one of 155 objects. As previously already mentioned, the mission will consist on a chaser that installs a deorbit tool in a few objects, so they are deleted. The number of objects under study are 4, 5 and 8, each of whom will have different time bounds.

2.2 Astrodynamics concepts

This section will provide a brief explanation about different concepts of astrodynamics. Such concepts are considered necessary to be known in order to understand the rest of the work, at least with a global vision of them. Basic concepts such as Newton laws, gravity force, geometry basis… are supposed to be known.

2.2.1 Orbital elements

These parameters, also called the Keplerian elements, describe the main characteristics of the orbit. This one is the most common way of describing an orbit.

The primary body is the one in which the reference is fixed, while for the body that moves around it is called the secondary body, normally a satellite or planet. Depending on the choice of which body is the primary, the orbital elements are different, nevertheless, is commonly assumed that the body with greater mass is the primary.

So, according to Keplerian laws, the orbits are contained in flat conical trajectories, will be explained for an ellipse, despite being also applicable to a parabola or hyperbola.

The six orbital elements, which are shown in figure 2.2, are the following:

- Semimajor axis (*a*): Sum of periapsis and apoapsis distances divided by two. Defines the size of the ellipse.
- Eccentricity (*e*): Elongation in comparison with a circle. Defines the shape of the ellipse.
- Inclination (*i*): Angle between the ecliptic plane and the plane of the orbit. Defines the orientation of the plane of the orbit.
- Longitude of the ascending node (Ω): The position in the orbit where the path of the secondary body passes through the ecliptic plane. Angle from the vernal equinox.
- Argument of periapsis (*ω*): Angle between the ascending node and the perigee of the orbit. Defines the orientation of the ellipse.
- True anomaly (*f*): Position of the secondary body in the ellipse that performs its trajectory.

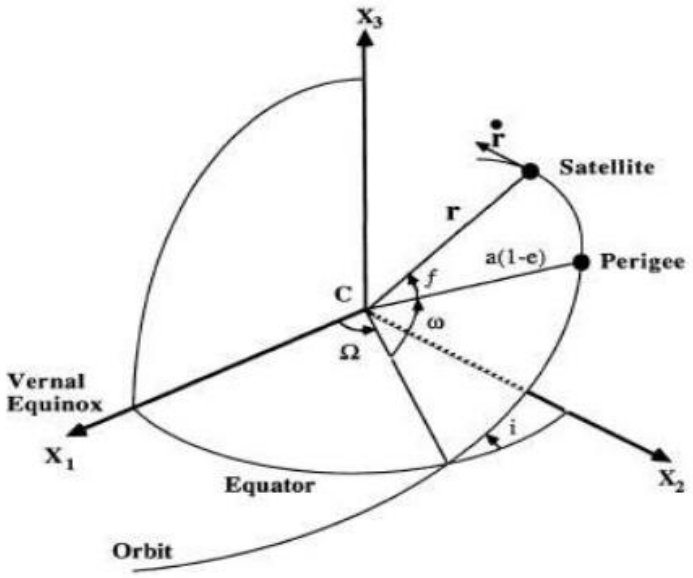

Figure 2.2 – Orbital elements

2.2.2 Manoeuvers

2.2.2.1 Coplanar manoeuvers

Supposed a body that performs a circular orbit, r_2, around a primary body and subjected to its gravity field. Just to keep the example simple, that body, the satellite, must be transferred to another circular orbit of radius r_2. Therefore, the velocity of the satellite must change, introducing a number of impulses depending on the strategy used for the transfer. Those changes of velocities will be supposed to happen in a very short time in order to consider it instantaneous. The last impulse is responsible of the recircularization of the orbit.

The simplest example is the one showed in figure 2.3. In this example only two impulses are performed, obtaining a transfer ellipse, where the impulses are applied at the perigee and apogee. It must be noticed that if the transfer ellipse is tangential to the final circular orbit, is called a Hohman transfer. For small differences between orbit radius, this type of transfer is the most efficient in terms of fuel consumption. Nevertheless, if the objective is to save time, greater impulses must be applied with the consequent change of shape of the transfer ellipse.

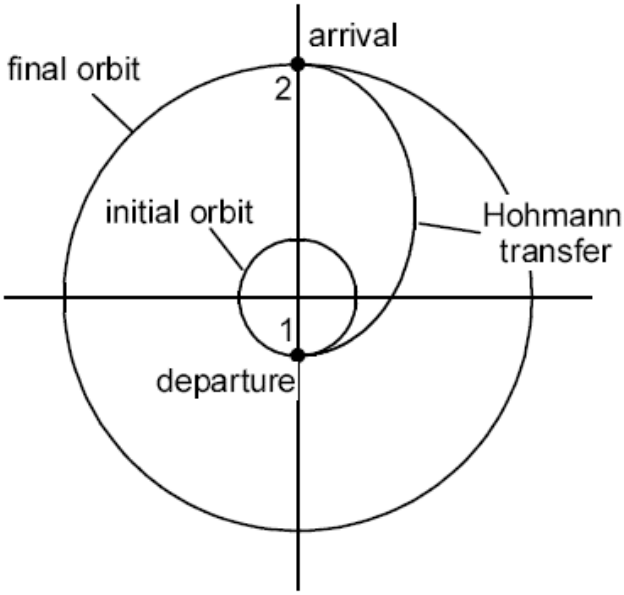

Figure 2.3 – Hohman transfer

The equation of the energy must be fulfilled:

$$\varepsilon = \frac{V^2}{2} - \frac{\mu}{r} = -\frac{\mu}{2a} \tag{2.1}$$

With this equation, introducing the correspondent values the following expressions for the necessary impulses can be reached:

$$\Delta V_1 = \sqrt{\frac{\mu}{r_1}}\left(\sqrt{\frac{2r_2}{r_1 + r_2}} - 1\right) \tag{2.2}$$

$$\Delta V_2 = \sqrt{\frac{\mu}{r_2}}\left(1 - \sqrt{\frac{2r_2}{r_1 + r_2}}\right) \tag{2.3}$$

So, the total impulse is the sum of the two, negative impulses mean the direction of the impulse.

$$\Delta V_{tot} = |\Delta V_1| + |\Delta V_2| = \sqrt{\frac{\mu}{r_1}}\left(\sqrt{\frac{2r_2}{r_1 + r_2}} - 1\right) + \sqrt{\frac{\mu}{r_1}}\left(\sqrt{\frac{2r_2}{r_1 + r_2}} - 1\right) \tag{2.4}$$

Apart from the Hohman manoeuvers, some other can be found that are of great interest. Some of the most important are the biparabolic or bielliptic where three impulses are applied, the second impulse at the infinite in the biparabolics. In the next figure the fuel consumption can be seen according to the transfer strategy and the ratio between final radius and initial radius of the orbits.

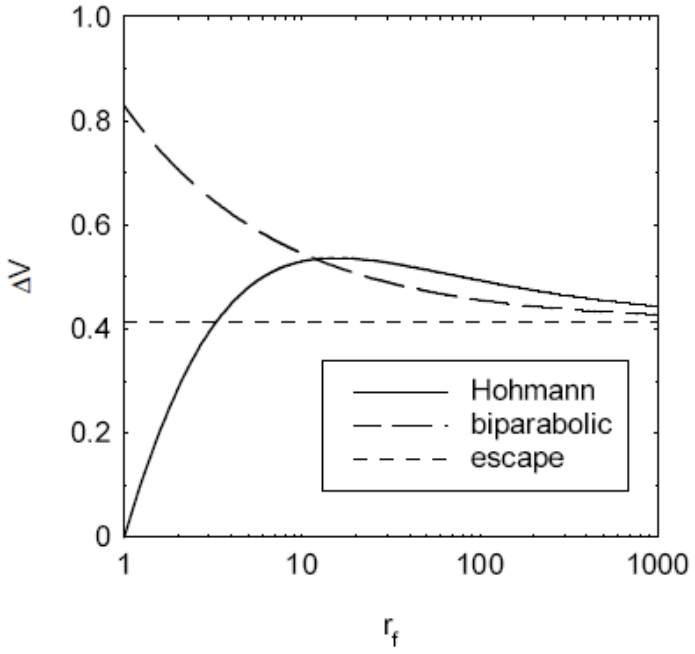

Figure 2.4 – Necessary impulse according to the strategy used

2.2.2.2 Non-coplanar manoeuvers

Until now, the presented type of manoeuvers were for coplanar orbits. However, if the orbit planes are different additional impulses must be applied. Of course, if a manoeuvre must change of radius and orbital plane, the fuel consumption will be greater.

When the orbit planes differ only in the inclination, the instantaneous impulse must be applied at the equator to minimise the fuel consumption. This is due to the following relation:

$$\sin i \, di = \cos\delta \sin\varphi \, d\varphi \tag{2.5}$$

Where i is the inclination, δ is the latitude and φ the angle between the velocity and the tangent of the trajectory. Taking this into consideration the fuel consumption for a plane change is closely related to the orbital velocity as follows:

$$\Delta V = 2V \sin\frac{\Delta i}{2} \tag{2.6}$$

On the other hand, if the longitude of the ascending node is needed to be change, it cannot be done without a change in inclination, unless for polar manoeuvers. The impulse needed for such a change in a circular orbit follows equation 2.8.

$$\cos\Delta A = \cos i_1 \cos i_2 + \sin i_2 \sin i_1 \cos\Delta\Omega \tag{2.7}$$

$$\Delta V = 2V \sin\frac{\Delta A}{2} \tag{2.8}$$

Once this presented, all these strategies can be mixed so that the optimal transfer for each case is used. In fact, the present work concerns targets at low altitudes and therefore, as can be seen in equation 2.6, the change of plane's cost is considerably high. To avoid such a consumption, the strategy to perform the manoeuvre is to wait until the debris are aligned due to the change of **Ω** caused by the J2 perturbation.

2.3 Perturbations

In all the spatial missions there are several perturbative phenomena that sometimes can be not considered due to the small effects on the missions. These perturbations affect the six orbital elements, which were presented in the subsection 2.2 of the present chapter, changing their expected values from the non-perturbated two body problem.

2.3.1 Classifications

The perturbations that affect any body, in this case the objects to be deleted and the satellite to remove them, can be divided into three big groups:

- **Secular variations:** linear variation in the element. This type of variations has as consequence long-term effects on the orbit.
- **Short-period variations:** periodic in the element with a period less than or equal to the orbital period. Important in case of precise orbit determination.
- **Long period variations:** periodic in the element with a period greater than or equal to the orbital period. Important in case of precise orbit determination.

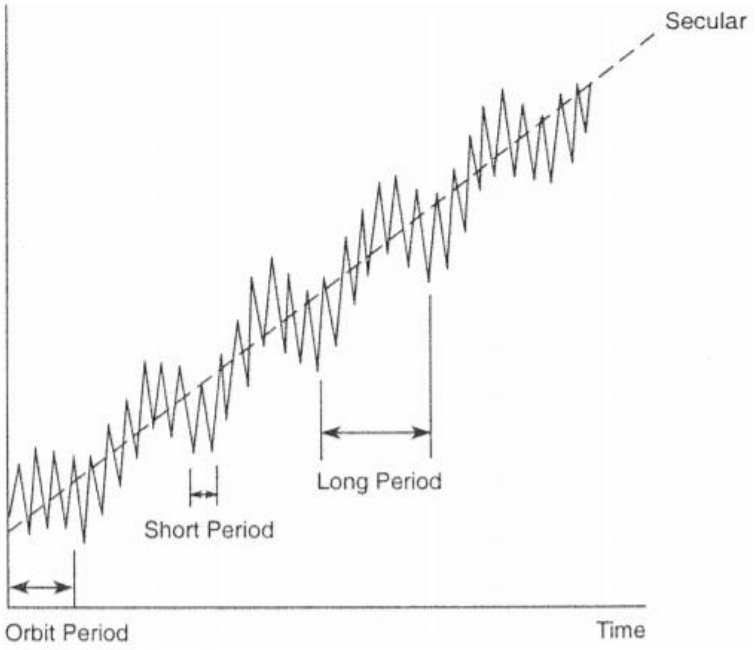

Figure 2.5 - Perturbations

Once this said, the most important perturbation causes are presented below. Firstly, in table 2.1 the effects they cause according to the mission type, and afterwards, a brief description of them.

Cause	LEO	GEO	Interplanetary
Non-sphericity of Earth	Variance of the nodes, perigee argument and mean anomaly	Important. Decreases rapidly with altitude	J_2 important in orbits of planets with high rotation
Third body	Rotation of 0.007 *deg/day*	Relevant	Important
Atmospheric drag	Not considered	Not considered	If close to planets with atmosphere
Solar radiation	Slight increase of eccentricity	Slight increase of eccentricity	Almost never considered
Relative effects	Not considered	Not considered	Generally, not considered

Table 2.1 – Perturbation causes

2.3.1.1 Non-sphericity of Earth

The Earth is assumed as a perfect sphere, nevertheless, in reality this is not true. In fact, the mass distribution is not symmetric, being like a slight pear shape, and flattering at the poles. If the body presents axial symmetry, the gravitational potential can be expressed as:

$$V(r,L) = \frac{\mu}{r}\left(1 - \sum_{n=2}^{\infty} J_n \left(\frac{R_E}{r}\right)^n P_n \cos L\right) \tag{2.9}$$

Where μ is Earth's gravitational constant, R_E is Earth's equatorial radius, P_n are Legendre polynomials, L is geocentric latitude, and J_n are dimensionless geopotential coefficients of which the first three are:

$$J_2 = 0.00108263$$

$$J_3 = -0.00000254$$

$$J_4 = -0.00000161$$

So, it depends on the latitude, with the previous coefficients being named as *zonal coefficients*. There are also some other expressions of the geopotential which are longitudinally dependant, with the *sectoral terms*. Lastly, the terms that depend on the longitude and latitude are called tesseral terms.

Although it causes variation in all the orbital elements, the dominant effects are secular variations in right ascension of the ascending node and argument of perigee because of Earth's oblateness, represented by J_2.

$$\dot{\Omega}\left[\frac{deg}{day}\right] = -1.5n \cdot J_2 \left(\frac{R_E}{a}\right)^2 \frac{cosi}{(1-e^2)^2} \tag{2.10}$$

$$\dot{\omega}\left[\frac{deg}{day}\right] = 0.75n \cdot J_2 \left(\frac{R_E}{a}\right)^2 \frac{(4-5sin^2 i)}{(1-e^2)^2} \tag{2.11}$$

Where n is mean motion in *deg/day*, R_E is Earth's equatorial radius, a is semimajor axis in *km*, e is eccentricity, and i is inclination. As it was anticipated in table 2.1, the variation in the orbital parameters decreases rapidly with the altitude.

2.3.1.2 Third body

Every object has mass, with the consequence of a gravitational field that affect to the other bodies. However, these fields are not considered due to the long distances between celestial bodies. The only bodies considered are the sun and the moon, which cause a periodic variation in all the orbital elements. Nevertheless, just the ascension of the ascending node, argument of perigee, and mean

anomaly experience secular variations. While the secular variation of the mean anomaly is small, the other two orbital parameters must be considered, especially for high-altitude orbits.

	Moon	Sun
$\dot{\omega}$ [deg/day]	$0.00169(4-5sin^2 i)/n$	$0.00077(4-5sin^2 i)/n$
$\dot{\Omega}$ [deg/day]	$-0.00338(\cos i)/n$	$-0.00154(\cos i)/n$

Table 2.2 – Variation of orbital elements due to third body perturbation

2.3.1.3 Atmospheric drag

When there is an atmosphere, which is the case of Earth, the drag due to friction of the atmosphere acts in the opposite direction of the velocity vector, removing energy from the orbit. With this energy reduction, the orbit becomes smaller, consequently, an increase in drag force. This could lead eventually to a re-entrance into the atmosphere. The acceleration for this effect is:

$$a_D \left[\frac{m}{s^2}\right] = -(\frac{1}{2})\rho \frac{C_D A}{m} V^2 \tag{2.12}$$

Where ρ is the air density, A the satellite's cross-sectional area, a is its mass, V is the satellite's velocity with respect to the atmosphere, and C_D is the drag coefficient equal to 2.2. Furthermore, for near circular orbits, we can approximate the changes in semimajor axis, period, velocity and eccentricity for each revolution as follows:

$$\Delta a_{rev} = -(\frac{1}{2})\rho \frac{C_D A}{m} V^2 \tag{2.13}$$

$$\Delta P_{rev} = -6\pi^2 \rho \frac{C_D A}{m} \frac{a^2}{V} \tag{2.14}$$

$$\Delta V_{rev} = \rho a \pi V \frac{C_D A}{m} \tag{2.15}$$

$$\Delta e_{rev} = 0 \tag{2.16}$$

Where P is the orbital period and V is the satellite velocity.

It must be noticed that the expressions are strongly dependant on the air density, which varies a lot between the different layers of Earth. Therefore, some complex models such as *Jacchia* are used in order to estimate the air density in some layers.

2.3.1.4 Solar radiation

Solar radiation causes periodic variations in all the orbital elements, however, is only considered for satellites with low ballistic coefficients. The acceleration can be expressed as follows:

$$a_R \left[\frac{m}{s^2}\right] = -4.5\ x\ 10^{-6}(1+r)\frac{A}{m} \tag{2.17}$$

where A is the satellite cross-sectional area exposed to the Sun in m^2 , m is the satellite mass in kg, and r is a reflection factor. ($r = 0$ for absorption; $r = 1$ for specular

2.3.2 Mathematical models

To predict the changes made in the orbital elements, different techniques are used, generally divided into two different groups:

- **Special perturbations:** these employ direct numerical integration of the equations of motion. The most common among all of them is *Cowell's method*, in which the accelerations are integrated directly to obtain the velocity and integrated again in order to obtain the position.
- **General perturbations:** analytically solve some aspects of the motion of a satellite subjected to perturbing forces. Nevertheless, most perturbating forces cannot be solved

by a direct analytical solution, but by series expansions and approximations. As the orbital elements are nearly constant, general perturbation methods usually solve directly for the orbital elements rather the position and velocity. Solutions are reached much faster than with special perturbations.

The problem scenario is explained now, and later two different set of equations are presented, with a brief explication for them. The mathematical deduction can be easily found in any book of orbital perturbations.

A particle *M* of mass m moves respect an inertial reference system $Ox_Iy_Iz_I$ called solid body. O is the mass centre of the solid body, in this case the Earth. *M* is subjected to a gravitational force and a perturbation force $\vec{F}_P$, which include all the other forces previously explained. So, the movement of the particle is governed by equation:

$$m\frac{d^2\vec{x}}{dt^2} = -\frac{m\mu}{|\vec{x}|^3}\vec{x} + \vec{F}_P \tag{2.18}$$

Where $\vec{x}$ is the position vector of the particle in the inertial reference. If the perturbation force is omitted, the orbit will follow a Keplerian orbit, this is, a conic contained in the orbital plane. However, when it is considered the orbit will follow a warped curve in the most general way. Equation 2.18 is a differential equation of sixth order that can be integrated to obtain the position vector and the velocity vector.

2.3.2.1 Gauss equations

Until now, the orbit was referred in terms of cartesian components, 3 for position and 3for velocity. However, it may be more comfortable to express the orbit in terms of the Keplerian elements previously seen. After a mathematical process which is omitted for the present work, the equations of Gauss are obtained.

$$\frac{da}{dt} = \sqrt{\frac{a}{\mu}}\frac{2a}{\sqrt{1-e^2}}(e\sin f\, p_U + (1+e\cos f)p_V) \tag{2.19}$$

$$\frac{de}{dt} = \sqrt{\frac{a}{\mu}}\sqrt{1-e^2}\left(\sin f\, p_U + \frac{e+2\cos f + cos^2 f}{1+ecosf}p_V\right) \tag{2.20}$$

$$\frac{di}{dt} = \sqrt{\frac{a}{\mu}}\frac{\sqrt{1-e^2}}{1+e\cos f}\cos(\omega+f)\, p_W \tag{2.21}$$

$$\frac{d\Omega}{dt} = \sqrt{\frac{a}{\mu}}\frac{\sqrt{1-e^2}}{1+e\cos f}\frac{\sin(\omega+f)}{\sin i}p_W \tag{2.22}$$

$$\frac{d\omega}{dt} = \sqrt{\frac{a}{\mu}}\frac{\sqrt{1-e^2}}{e}\left(\cos f p_V + \frac{2+e\cos f}{1+ecosf}\sin f p_V\right) - \cos i\frac{d\Omega}{dt} \tag{2.23}$$

$$\frac{dM}{dt} = \sqrt{\frac{a}{\mu}}\frac{1-e^2}{nae}\left[\left(\cos f - \frac{2e}{1+ecosf}\right)p_V - \frac{2+ecosf}{1+ecosf}\sin f\, p_V\right] \tag{2.24}$$

These equations must be integrated numerically or analytically, form the known initial conditions. Apart from that, in the equations the term μ appears which can be replaced by the following expression:

$$\mu = n^2a^3$$

This group of equations is particularly adapted for the integration of the non-gravitational perturbations, these are, the solar radiation and the atmospheric drag.

2.3.2.2 Lagrange equations

When the acceleration of the perturbation derives from a potential:

$$\overrightarrow{a_p} = -\nabla R_p \tag{2.25}$$

The temporal evolution of the orbital elements is described in the planetary Lagrange equations:

$$\frac{da}{dt} = -\frac{2}{na}\frac{\partial R_p}{\partial M} \tag{2.26}$$

$$\frac{de}{dt} = \frac{\sqrt{1-e^2}}{ena^2}\left(\frac{\partial R_p}{\partial \omega} - \sqrt{1-e^2}\frac{\partial R_p}{\partial M}\right) \tag{2.27}$$

$$\frac{di}{dt} = \frac{1}{na^2 \sin i \sqrt{1-e^2}}\left(\frac{\partial R_p}{\partial \Omega} - \cos i \frac{\partial R_p}{\partial \omega}\right) \tag{2.28}$$

$$\frac{d\Omega}{dt} = \frac{1}{na^2 \sin i \sqrt{1-e^2}}\frac{\partial R_p}{\partial i} \tag{2.29}$$

$$\frac{d\omega}{dt} = \frac{\sqrt{1-e^2}}{ena^2}\left(\frac{e\cot i}{(1-e^2)}\frac{\partial R_p}{\partial i} - \frac{\partial R_p}{\partial e}\right) \tag{2.30}$$

$$\frac{dM}{dt} = n + \frac{2}{na}\frac{\partial R_p}{\partial a} + \frac{1-e^2}{ena^2}\frac{\partial R_p}{\partial e} \tag{2.31}$$

It must be noticed that the singularities appear when $e=0$ and/or $i=0$. The derivates of the angles (Ω, ω, i) tend to infinite when $i \to 0$. The same thing occurs when $e \to 0$ with the derivates of the angles (ω, e, M). In order to delete these singularities, the classic orbital elements must be replaced by others, such as the *equinoctial orbital elements*.

This group of equations is particularly adapted for the integration of the gravitational perturbations, these are, the perturbations of the non-spheritic of Earth and for the third body perturbations.

2.3.3 Simplifications

When a spectral analysis is applied to the different perturbations, some conclusions can be drawn, such as the dominant perturbations and the effects in each orbital element:

- **Semimajor axis, a**: Short period variations due to the non-spherite of the Earth (J_2), with a period of *T/2* and amplitude of $\pm 9km$. Long period variations due to the presence of the sun, presence of the moon and solar radiation. The aerodynamic resistance has as consequence a secular decrease of the axis value.
- **Eccentricity, e**: Short period variations (T and T/3) due to J_2 and long period variations due to the harmonics J_{2n+1}.
- **Inclination, i**: Short period variations (T/2) due to J_2 and long period variations due to the harmonics J_{2n+1}.
- **Longitude of the ascending node, Ω:** Short period variations (T/2) due to J_2 and long period variations due to the harmonics J_{2n+1}. Apart from those, secular variations of 10 degrees each day due to the harmonics J_{2n}.
- **Argument of periapsis, ω:** Short period oscillations (T and T/3) due to J_2. Long period oscillations due to the harmonics J_{2n+1}. Apart from those, secular variations of 20 degrees each day due to the harmonics J_{2n}.

- **Mean anomaly, M**: Subjected to the same perturbations as the inclination, plus a secular variation of 5900 degrees each day.

Taking all this into account, the equation's system presented before can be obtain in term of approximation of first order. As we are considering the trajectory in LEO, the dominant perturbations are the aerodynamic force and the non-sphericity of the Earth. Apart from that, it must be mentioned that in the problem that occupies this thesis all the orbits are approximated as quasi circular.

$$\frac{da}{dt} \approx -C_D \frac{\mathrm{A}}{m} \rho_{pe} r^2 \frac{n}{2\pi} \tag{2.32}$$

$$\frac{de}{dt} \approx 0 \tag{2.33}$$

$$\frac{di}{dt} \approx 0 \tag{2.34}$$

$$\frac{d\Omega}{dt} \approx -\frac{3}{2} n J_2 \left(\frac{r_e}{p}\right)^2 \cos i \tag{2.35}$$

$$\frac{d\omega}{dt} \approx \frac{3}{4} n J_2 \left(\frac{r}{p}\right)^2 (4 - 5sin^2 i) \tag{2.36}$$

$$\frac{dM}{dt} \approx n + \frac{3}{4} n J_2 \left(\frac{r_e}{p}\right)^2 \sqrt{1 - e^2}(2 - 3sin^2 i) \tag{2.37}$$

2.4 Mathematical model for the problem

The mathematical model used for the debris of the Kosmos considers only the non-sphericity of Earth as the only perturbation in the orbit of them. Therefore, the major semiaxis, eccentricity and inclination remain constant over the time. So, the variations of the true anomaly, argument of periapsis and longitude of the ascending node are shown in equations 2.35, 2.36 & 2.37 of the previous section.

In such equations, *n* represents the mean mode, defined as:

$$n = \sqrt{\frac{\mu}{r^3}} \tag{2.38}$$

With μ as Earth's gravitational parameter, and r the radius of the quasi-circular orbit. Furthermore, *p* is defined as:

$$p = r(1 - e^2) \tag{2.39}$$

All these approximations that are done may introduce big errors. Hence, it is compared to a propagation of the orbit performed by the method SGP4, obtaining similar results concerning the shape of the orbit. However, the error in the mean anomaly and the argument of periapsis is considerable. Moreover, the effect of the argument of the periapsis is small due to the almost null eccentricity.

2.4.1 Transfers

It is assumed that the moment when the transfer will be carried out is the optimal, this is, when the difference between the two orbital planes is null. The variation of the longitude of the ascending node allows the orbital planes to become closer. This happens because this variation, as can be seen in equation 2.35, depends on the altitude and the eccentricity. Hence, each object will suffer a different variation of the longitude of the ascending node. Again, the altitude of each object remains constant and does not depend on time. With all this, the objects periodically will be aligned, unless there is a pair that have the same altitude and eccentricity. If the manoeuvre is

performed when objects are not aligned, the transfer cost would be much greater or even unfeasible.

So, the moments when the alignment takes place can be obtained as:

$$t_{jk} = \frac{\Omega_j(t_0) - \Omega_k(t_0) + 2K\pi}{\dot{\Omega}_k - \dot{\Omega}_j} \tag{2.40}$$

As it can be seen, the time in which two objects are aligned depends on both objects, and does not depend of the order of these two, So, it is the same going from object *k* to *j* or from *j* to *k*.

While for the transfer itself, it would be considered that all the transfers are coplanar and follow a Hohman transfer. This type of transfer is the best option when wanting to save fuel but is not the most efficient in terms of time. However, the transfer time is so small in comparison to the time to be waited for the alignment of the debris that is not considered.

For Hohman transfers, there is an empirical expression that relates the semi major axis and the eccentricity in a very simple way:

$$\frac{\Delta V}{V} = 0.5\sqrt{\left(\frac{\Delta a}{a}\right)^2 + \Delta e^2} \tag{2.41}$$

The semimajor axis introduces is the minor between the two objects considered, so the circular velocity is also referred to this one. It must be said that due to the characteristic of equation 2.41, it does not matter the order of the orbits. This is, the important value is the difference between both orbits, not affecting if is negative or positive as they are squared.

So, calculating all the necessary impulses that are needed in each leg, the total impulse to be applied is obtained. As will be explained in section 2.4.2, once the impulse, the initial mass and the propellant used are known, the fuel mass consumed is easily obtained.

On the other hand, the time when the manoeuvre is performed does not follow the same process as the impulse. While for the total impulse applied the impulses of all the legs were summed, for the total time of the mission just the time of the last leg is considered. Furthermore, the time when the manoeuvers between the last two debris is done, must be greater than all the previous times of each leg, considering the service time too.

The estimations of DV and time for each leg are verified with an evolutionary algorithm. With four impulses at:

$$\tau_1 < \tau_2 < \tau_3 < \tau_4 = \tau_{i+1} \tag{2.42}$$

Being τ_{i+1} the time of the next leg. To calculate each leg in an exact and optimised way the terms $p_1, p_2 \ldots p_8$ are introduced, which correspond to the parameters of the previously mentioned evolutionary algorithm.

$$p_1 = \tau_4 - \tau_1 \tag{2.43}$$

$$p_2 = \frac{\tau_2 - \tau_1}{\tau_4 - \tau_2} \tag{2.44}$$

$$p_3 = \frac{\tau_3 - \tau_2}{\tau_4 - \tau_2} \tag{2.45}$$

where p_1 varies in a range of ten days with the centre on the value suggested by global search while p_2 and p_3 vary between zero and one.

There are six other variables that define the components of the velocities after the first and second impulses. The notation used for them is 1^+ and 2^+ for after the impulses and 1^- e 2^- for before the impulses are performed. These variables plus the flight path angle are shown in equations from 2.46 to 2.49. The flight path angle is the angle measured from the local horizontal (perpendicular to *r*.) to the velocity direction.

$$u_{1+} = u_{1-} + p_4 \sin(p_5) \tag{2.46}$$

$$u_{2+} = u_{2-} + p_7 \sin(p_8) \tag{2.47}$$

$$v_{1+} = v_{1-} + p_4 \cos(p_5) \cos(\varphi + p_6) \tag{2.48}$$

$$v_{2+} = v_{2-} + p_7 \cos(p_8) \cos(\varphi + p_9) \tag{2.49}$$

$$w_{1+} = w + p_4 \cos(p_5) \cos(\varphi + p_6) \tag{2.50}$$

$$w_{2+} = w_{2-} + p_7 \cos(p_8) \cos(\varphi + p_9) \tag{2.51}$$

$$\varphi = \tan^{-1}\left(\frac{u_-}{v_-}\right) \tag{2.52}$$

Where p_4 and p_7 are the variations of velocity that are spaced between 0 and 800 *m/s*. The angles p_5 and p_8 vary between +180 and -180 degrees. While angles p_5 and p_8 vary between +60 and -60 degrees. Lastly, the obtention of the last variable p_{10} is related to the las part of the leg, the *rendezvous*. Hence, this one requires of a different approach, as the final position is fixed and therefore necessary to solve a Lambert problem.

Due to the different possible orbits to be chosen, the problem presents two solutions. Thus, p_9 is used in order to decide if take the leftwards solution ($p_{10} < 0.5$) or rightwards solution ($p_{10} > 0.5$). Once all the ten variables are obtained, the impulse for each manoeuvers can be evaluated. In fact, the initial time, position and velocity are known. After the first impulse velocity is calculated, the Kepler problem is resolved, considering the effect of J_2. The same process is done until the last arch of the trajectory, where the Lambert non-perturbed problem must be solved in order to achieve the velocity components. After this, the results are corrected through an iterative Newton method-based process that consider the non-sphericity of Earth.

So, the impulse for the leg follows this expression:

$$\Delta V = \sum_{j=1}^{4} \sqrt{\left(u_{j+} - u_{j-}\right)^2 + \left(v_{j+} - v_{j-}\right)^2 + \left(w_{j+} - w_{j-}\right)^2} \tag{2.53}$$

2.4.2 Mass budget

For each object to be deleted, the vehicle must provide a kit for the phase of deorbit, in order to introduce the necessary impulse for the removal of the object. In this mission, such a process is performed with chemical propulsion, reducing the perigee of the orbit. The reason for this is to obtain a re-entry of the debris. Two different re-entries can be distinguished:

- Controlled re-entry: is the ability to force the entry over a pre-determined area, region, within which the debris is to fall. To do so, the new trajectory of the object must touch the Earth's radius.

- Uncontrolled re-entry: there is less knowledge about the trajectory of the object. Instead of a specific region, one can ensure the fall of the debris within a few orbital revolutions. To do so, the perigee of the object's orbit must be at 125 km of altitude.

For the present work, the recommendation is to choose the uncontrolled atmospheric entry, as it is the most usual when talking about space debris.

The transfer performed will be a Hohman transfer. Thus, the change of velocity the removal kit must provide at the apogee of the orbit is:

$$\Delta V = \sqrt{\frac{\mu}{r_a}}\left(1 - \sqrt{2}\sqrt{1 - \frac{r_a}{r_a + r_p^*}}\right) \tag{2.54}$$

Being the perigee, r_p^*, equal to the R_E for controlled entries and equal to $1.02R_E$ for uncontrolled. Consequently, uncontrolled entries need less propellant to be performed.

To obtain the mass of the chaser, firstly, the rocket equation must be considered. This one relates the fuel consumption in a transfer to the necessary impulse to be performed. So, if the propellant is known, after obtaining the changes in velocity as was explained in the previous section, the mass loss is obtained.

$$\Delta V_i = c_c \ln \frac{(m_f)_i}{(m_0)_i} \tag{2.55}$$

The index i indicates the leg of the mission. Where m_f and m_0 are the mass at the end and at the beginning of the leg respectively.

Apart from that, it must be considered that for every object deleted a deorbit kit is installed, so that mass is also removed from the chaser. Calling the deorbit kit mass m_{dk} and supposing that all the debris use the same deorbit kit:

$$m_i = (m_0)_i + (m_{dk})_i \tag{2.56}$$

Where m_i is the mass of the chaser before getting rid of the deorbit tool. It must be noticed the link between the different legs which is:

$$(m_f)_{i-1} = m_i \tag{2.57}$$

With these three equations the initial mass can be obtained, starting by fixing the final mass of the mission as the chaser mass, which is known, without the deorbit kits and the propellant. Furthermore, the propellant consumption for each leg is easily obtained as follows:

$$(m_p)_i = (m_f)_i - (m_0)_i \tag{2.58}$$

To conclude with this section, it must be outlined that the chaser, deorbit kit and propellant have not been chosen yet, hence, they are the reason why this problem cannot be solved yet.

CHAPTER 3

Optimisation Algorithm

3.1 Introduction to evolutive algorithms

The evolutive algorithms are solutions search and optimisation methods based in the biological evolution theory. In them a group of different entities is maintained which represent the possible solutions. These ones are mixed, compete between each other, so the best ones are able to survive over the time and if possible, evolving into better options. Normally, these types of algorithms are used in problems with a wide range of possibilities and these being non-linear. In these type of spaces, other techniques are unable to reach a solution.

There are several subclasses more specialized than evolutionary algorithms, each of which interprets in a different way the philosophy at the base. Among these, the genetic algorithm is the one that clearly shows the natural selection process in its operating mechanism. In figure 3.1 are shown the different subclasses of evolutionary algorithms.

On the right of figure 3.1, some of the most relevant heuristics can be seen. The objective of a heuristic is to produce a solution in a reasonable time. This solution does not have to be the best one but be a good solution. Nevertheless, it is still valuable because finding it does not require too much time. Although heuristics may produce results by themselves, they are normally used in conjunction with optimization algorithms to improve their efficiency.

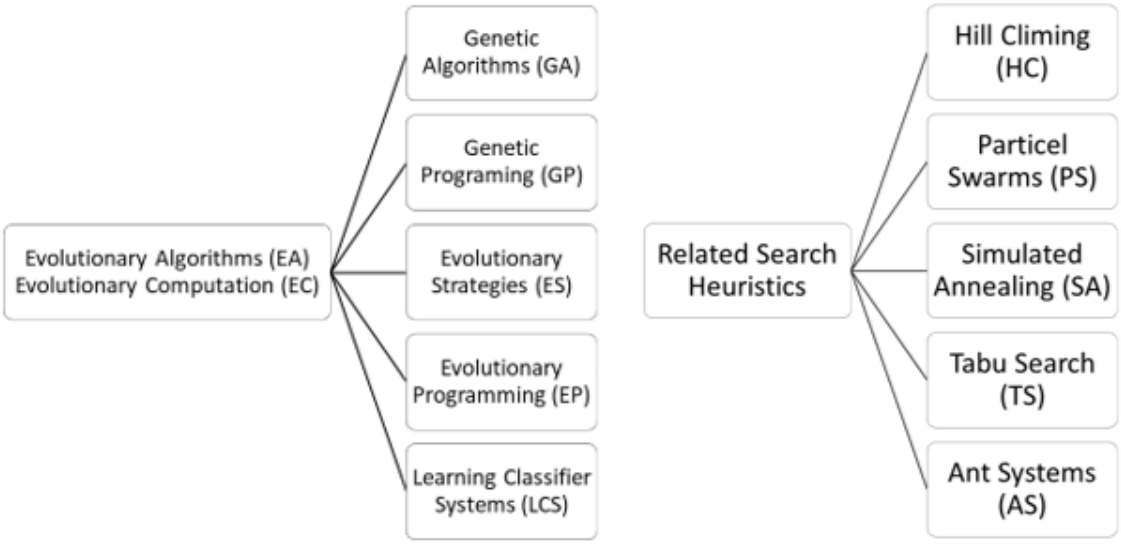

Figure 3.1 – Classification of EA methods

The present work will involve only one type among all these, a genetic algorithm, that will be deeply described in the following sections.

3.2 Genetic Algorithms

The genetic algorithms (GA) are adaptative methods that can be used to solve search and optimisation problems. As said before, they are based in the biological reproduction, with generations, genes, populations that evolve following the natural selection principles stated by Darwing (1859). The basic principles of these algorithms were stablished by Holland in his book *Adaptation in Natural and Artificial Systems* and can be seen described in detail in texts such as Goldberg (1989) and Reeves (1993).

The original motivation for the genetic algorithm approach was a biological analogy. In the selective breeding of plants or animals, for example, offspring are sought that have certain desirable characteristics, characteristics that are determined at the genetic level by the way the parents' chromosomes combine. In the case of GAs, a population of strings is used, and these strings are often referred to in the GA literature as chromosomes. The recombination of strings is carried out using simple analogies of genetic crossover and mutation, and the search is guided by the results of evaluating the objective function f for each string in the population. Based on this evaluation, strings that have higher fitness can be identified, and these are given more opportunity to breed.

The basics of a genetic algorithm are shown in figure 3.2. Firstly, the population is initialised, then, there is an evaluation of all the members in order to obtain which are the best ones. After that, the population goes through crossover and mutation, so the evolution takes place. Once this done, there is a selection process and again the loop is performed. The process is looped until a criterion is reached, normally concerning a number of iterations. Finally, when the loop is over, the best solution is obtained.

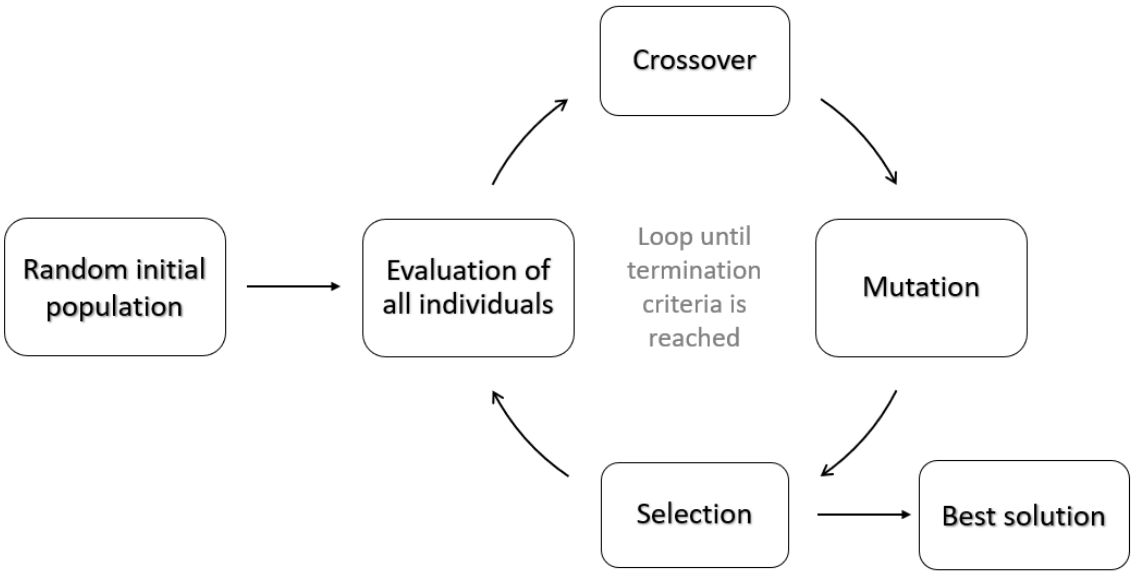

Figure 3.2 – Genetic Algorithm process

It must be highlighted that this type of algorithm does not assure reaching the best solution. Therefore, they are normally run several times providing different results each time. Not even after numerous simulations the best solution can be assured as normally not all the possibilities are explored.

3.3 Initialisation

The process of initialisation is crucial in the correct functioning of a genetic algorithm as it defines the evolution of this one. The initial population should have at least two characteristics:

- A diverse population in order to avoid premature convergence
- A correct number of individuals, not too high because could slow down the algorithm and not too low because could lead to a lack of exploration of the possible solutions

Apart from that, the dimensions of the initial population in the present problem is also defined by the number of objects to be removed. This, one of the main objectives is to determine the optimal dimensions of the population, study that will be done in the following chapter.

The initialisation of a genetic algorithm can be divided in two:

- **Random initialisation:** When there is no knowledge of the solutions that the algorithm will find the population is populated randomly.
- **Heuristic initialisation:** When there is some knowledge about the possible solutions the population is populated using a known heuristic for the problem.

It must be outlined that with the Heuristic initialisation there is less diversity, so it is not recommendable to use this initialisation in the entire population, or at least not every time. However, as it will be seen in further chapters, it could be a better option for the exploration of the vicinity of a solution previously found. A balanced initialisation could be the best option, initialising just a couple of individuals heuristically while the rest of them randomly.

Another important aspect, for the problem that concerns this thesis, is that an initial population with bad fitness values could lead to difficulties in converging. Thus, it may be useful to reinitialise the population until a better population is set. Such a study will be described deeper and presented in chapter 4.

3.4 Selection

After the initialisation of the genetic algorithm, the second step to be performed is the selection. This means, how to choose the individuals in the population that will create offspring for the next generation, and how many to choose. The objective of this process is to select the best individuals in terms of fitness value, so that the future generation could lead to even better solutions. Nevertheless, the selection shall be balanced as a correct exploration of the possibilities is worthy to be done. If the selection is too strong the diversity of the population could be significantly reduced, and therefore, homogenised. Thus, if the selection technique chooses almost every individual, the evolution in the population could be too slow. This is why several techniques have been implemented over the years, some of them described in the present work.

3.4.1 Tournament selection

This type of selection is one of the most popular due to the facility to be implemented and its efficiency. In this method a random number of individuals is chosen, which compete between each other. The best of these ones is selected and then proceeded to the next steps of the algorithm. The number of random individuals chosen can vary, but normally is equal to 2. So, this method allows the next generation to have a wide variety in their individuals, as does not prioritise those with better fitness values. Furthermore, the computational cost is small as there is no need to sort the fitness values and the diversity of the new population is almost assured. It must be noticed that the selection of random individuals can repeat individuals and is repeated the necessary times.

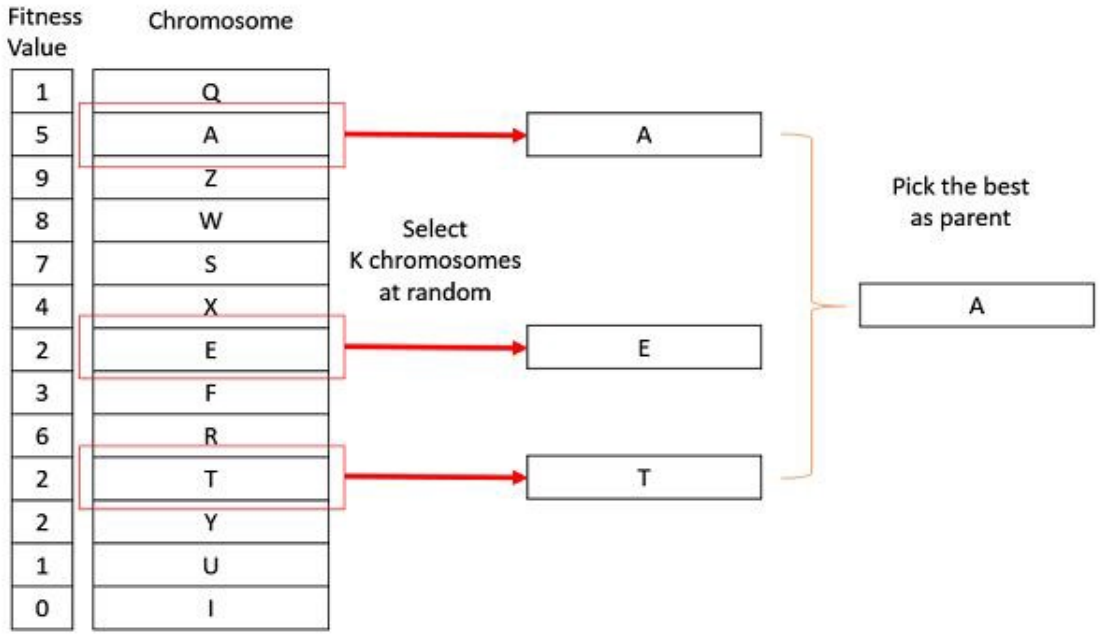

Figure 3.3 – Tournament selection

3.4.2 Roulette wheel selection

Also called fitness proportionate selection, the fitness function assigns a fitness to possible solutions or chromosomes. In this selection method, an individual of the population corresponds to a little part of a wheel of chance. The size of such a part is proportional to the calculated value of the fitness. So, the wheel is then tossed as many times as parents are needed to create the new population and each individual that wins is copied into the new population. Therefore, a single individual can occur multiple times in the parent population. In this selection method the *Darwinist* principle of the survival of the fitness takes place, where the less fit individuals are eliminated. This means, the less probable individuals to be selected are those with the less fit value.

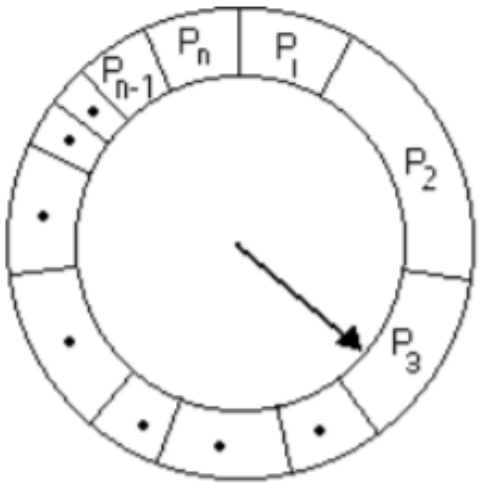

Figure 3.4 – Roulette wheel selection

The probability of each individual to be selected is the following:

$$P_i = \frac{f_i}{\sum_{j=1}^{n} f_j} \tag{3.1}$$

As it will be seen in later, this way of distribution of the probability may cause problems for the individuals whose fitness value is very low. In this line, this method can cause blocking as the population will get homogenised, hence, is not able to explore new and different solutions from the best ones.

3.4.3 Ranked selection

When among the population the fitness value differs a lot, or there is one that is much better than the other ones, the *wheel selection* may have problems, since the portion for each individual is proportional to their fitness value. This selection method takes care of this fact by sorting the different individuals from best to worst. Later, a probability is assigned to each of them according to their position. This is the key issue in this method, the probability is assigned according to the position and not to the fitness value. The method functions jus as the wheel selection but with this different detail. In comparison with the *wheel selection*, this method requires of an extra step, and therefore, the computational time is greater.

In figure 3.5 an example can be seen of a population of 4 individuals. It has to be noticed that the probability to be selected is not proportional to the fitness value but follows a linear law according to the position of the individual. The probability of choice can follow different laws such as exponential, lineal… however, in this case it corresponds to an easy law in order to clarify the concept.

$$P_i = 50 - Pos \cdot 10 \tag{3.2}$$

Where P_i is the probability of each individual to be selected, and Pos is the position in the sorted ranking.

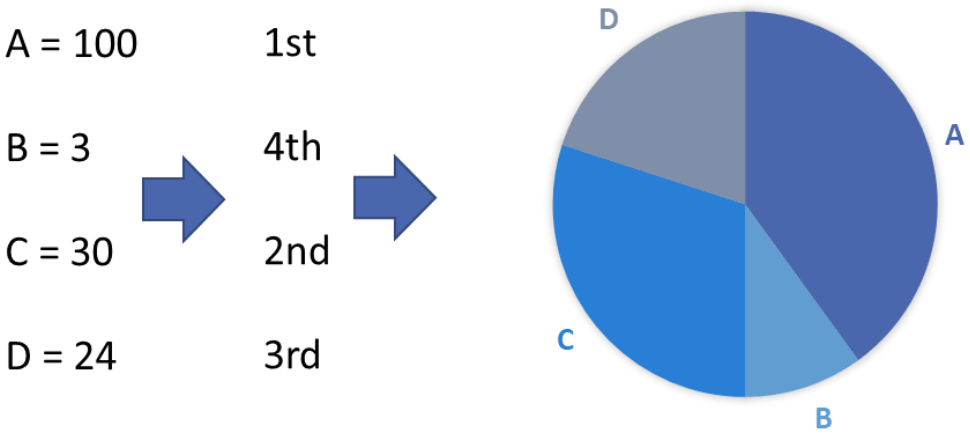

Figure 3.5 – Ranked selection

3.4.4 Truncation selection

The truncation selection is a very simple technique that firstly, orders the sample of individuals taken from the population according to their fitness values. Then, only a certain percentage of the fittest individuals is selected. The number of individuals that is selected from the sample is called selection pressure. In practise it is less used than other methods, except for very large population. This selection method is often used by breeders and in genetic population. This method can be considered a branch of the best only selection, where only the best individuals from the population are selected.

3.4.5 Reward-based selection

This technique is probably the most different and innovative among the ones presented. In this case the probability of being selected of each member of the population is proportional to a cumulative reward. This cumulative reward of each individual is obtained as the reward of itself plus the reward obtained from his parents. With this, this selection method also considers the parents of an individual so it could be useful for recombination. However, as the cumulative reward has to be obtained, and the parents cannot be deleted, the computational cost of this method is significantly greater than the predecessors.

3.5 Crossover

The crossover consists on taking two solutions as if they were parents and combining their genes to create a new *child*. However, crossover could also be done with a different number of parents, although being two the most common due to the analogue to sexual reproduction in biology. Furthermore. The solutions can be generated by cloning an existing solution, analogue to asexual reproduction. For a good efficiency, the crossover must allow to conserve the genetic material from the parents and mix their genes in order to obtain better solutions. The crossover is a recombination operator that proceeds in two basic steps:

1. The reproduction operator selects a pair of individual strings and takes them as parents.
2. Recombination is done according to a specific strategy.

As also happens with the mutation, the basic parameter is the crossover probability. If it is set up in 100%, all offspring are made by crossover, while if it is 0% there is no crossover at all. Crossover function is to obtain a new population with good parts of the eldest population, nevertheless, it is often recommendable to leave some part of old population survive to the next generation.

In the present section some crossover strategies are described, in particular, those ones that are more used in the Travelling Salesman Problem.

3.5.1 Single point crossover

It is the simplest of all the presented in this work. A crossover point is picked corresponding to a position in the strings. Until the crossover point the genes are those of the father, after this point, the genes are those of the mother. On the other hand, the same thing is done but permutating the order of the parents, so two children are obtained from a couple of parents.

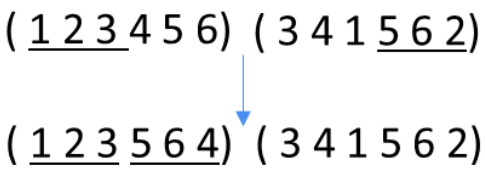

Figure 3.6 – Single point crossover

3.5.2 Double point crossover, k-point crossover

The two points crossover is the same as the previous one, but instead of one single crossover point this time there are two. So, the genes transferred are father-mother-father and mother-father-mother. When there are more than 2 crossover points, it is denominated k-point crossover where k is the number of crossovers.

3.5.3 City Centred crossover

This type of crossover is about picking a gen, not a position. Until this particular gen, the previous ones remain with the same order and are transmitted to the child. After the chosen gen, the child is completed with the genes of the other parent in the same order that they are in it. If a gen is already in the child, is omitted and repeat the process with the next one. As can be deduced, two children are obtained changing the roles of the parents. As an example, if gen 3 is the chosen gen in the following individuals:

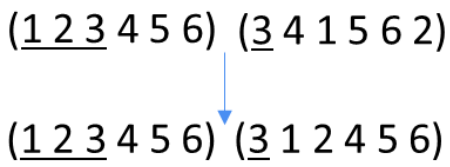

Figure 3.7 – City centred crossover

3.5.4 Ordered crossover

Firstly, some positions are selected from both parents. The genes between these positions are transmitted to the child in the same order. Then, from the other parent the rest of the genes are picked starting from the position just at right of the transmitted string. If the gen is already in the child, the gen is omitted and the next one is picked going from left to right. Then it is done permutating each parent's function, so two children are obtained. It is graphically explained in the example of figure 3.8, when selecting positions 3 and 5.

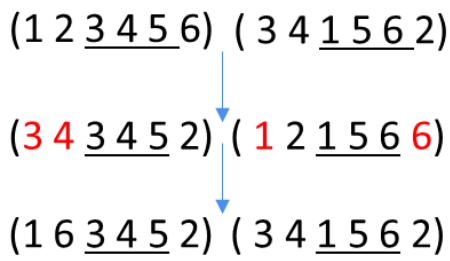

Figure 3.8 – Ordered crossover

This type of crossover is useful when wanting to transmit a sequence of genes to the next generation, due to the good result it provides.

3.5.5 Alternating-position crossover (AP)

In this case the child is obtained from a combination of the parents' genes, where they are selected alternatively between both parents. If the gen from the father or mother is already present in the output child, the gen is omitted.

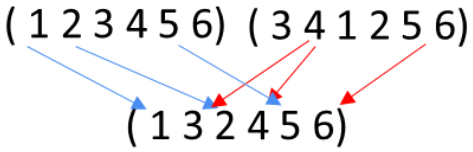

Figure 3.9 – Alternating position crossover

3.5.6 Partially-mapped crossover (PMX)

This crossover carries the most complex process among all the described this far. Notwithstanding having a much bigger time of execution, the results of this crossover are better than the ones above. Due to the complexity of the process, it will be explained with a written and graphical example.

Firstly, a string between two positions must be chosen from both father and mother. Keeping the same order, the strings are exchanged.

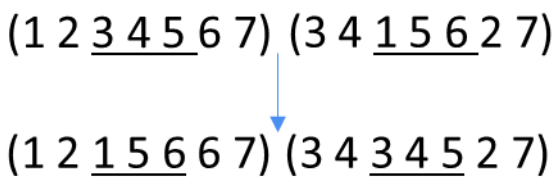

Figure 3.10 – PMX crossover (1)

Secondly, the relation, or map, between the previous genes is done. A gen from the father corresponds to a gen of the mother if they are in the same position. With this, the map obtained for this case is:

3 – 1

4 – 5 – 6

Now, the rest of the genes are changed according to the relations above, without repeating any gen in the same child. It must be highlighted that if the gen does not appear in the map there is no change in that gen.

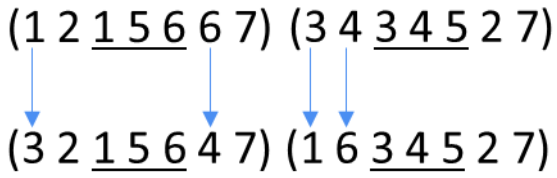

Figure 3.11 – PMX crossover (2)

The fact that increases the computational cost is the high number of steps that must be followed for the mixing of the parent's genes, where some relations must be considered.

3.6 Mutation

The mutation stage is the one that allows the algorithm to maintain genetic diversity from one generation of a population to the next one. This means, is in charge of changing the previous population in order to find better solutions if possible. The big difference with the process of crossover is that the mutation does not tend to get stacked in a homogenised population. With this, the mutation provides a much wider range of solutions after several iterations. For this reason, the mutation can provide fitness values very high or very low even though it is in advanced phases, so these solutions could dominate over the non-mutated ones.

However, it must be mentioned the fact that the efficiency of the genetic algorithm is strongly dependant of the mutation stage. If the mutation of the population is too high, the result would be a random search of solution, being useless the mutation strategy. In order to explore different

solutions correctly, diverse techniques are used, as the possibilities are numerous, just some of them are mentioned in this section.

3.6.1 Displacement mutation (DM)

In this first case, a gen from the father string is selected, and a position to me moved is also chosen. So, this mutation requires of two different numbers. The gen that is selected is moved to the indicated position, displacing all the string one position as follows:

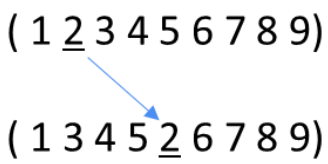

Figure 3.12 – Displacement mutation

3.6.2 Exchange mutation (EM)

Also referred as *swap mutation*, selects to different indexes and exchange their genes. The rest of the string remains without any change.

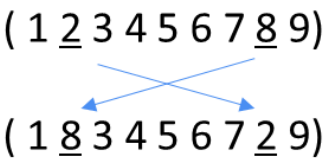

Figure 3.13 – Exchange mutation

3.6.3 Simple inversion mutation (SIM)

Two cut points in the string are selected, and the substring between these two cut points is reversed. As an example, if indexes 1 and 4 are chosen as cut points:

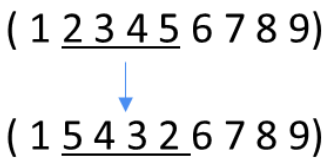

Figure 3.14 – Simple inversion mutation

3.6.4 Scramble mutation (SM)

Two indexes are chosen and scrambles the genes in them. Therefore, the genes which are contained in the interval of these positions are scrambled. For example, consider that the subtour chosen is (5 6 7 8):

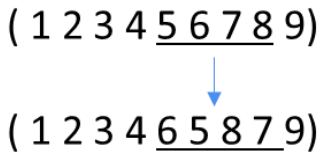

Figure 3.15 – Scramble mutation

3.6.5 Swap blocks (SB)

The genes are divided into different blocks, so different sections are done. For example, if the chromosome is divided in three different blocks the first and last block are swapped. The second block remains in the same position as it was before. Thanks to this type of mutation some groups of genes, which correspond to a good fitness value, are maintained. This fact allows the code to build better elements in the following iterations.

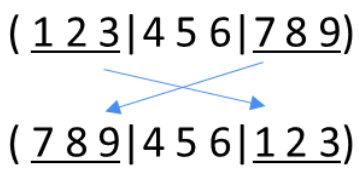

Figure 3.16 – Swap blocks mutation

3.7 The Code

The code used for the performance of this thesis has been written in MATLAB language. It consists on a genetic algorithm in order to obtain the most suitable solutions to the problems that were presented in chapter 2. The code has been developed with the intention of being as much versatile as possible. Therefore, it can be adapted to the different cases under study by introducing the required inputs for each case.

Even though the code is presented in appendix A, in this section it will be explained to understand how it works and to focus in the most important aspects of it. However, the lines of the script will not be commented one by one, but in a general and conceptual way.

3.7.1 Mission data loading

First of all, these data are loaded into the code. Considering that all the information about the debris is contained in a text file called *kosmos.txt*, the only thing to be done to obtain the data from the mission is load the txt. This text file contains all the necessary information about the diverse stages of *Kosmos 3M*. The imported data is stored in diverse vectors in order to keep a simple coding language. Below a list of the data imported from each column is presented.

- 1st column: Departure ID (Not used in this code).
- 2nd column: Arrival ID (Not used in this code).
- 3rd column: Difference in velocity necessary to go to one object to another.
- 4th column: Time when the transfer can be done
- 5th column: Difference in RAAN

Considering that there are 155 objects to be removed, as explained in chapter 2, the number of possible combinations is of 24025. Therefore, all these data are imported into vectors with dimension of 155. However, if another text file is imported with a different number of objects, the code can function correctly. This is because the number of debris to be removed can be changed in the following lines, set for this thesis in 155.

Apart from that, in this section the number of debris to be removed is introduced by the user, having been changed in the present work between 4, 5 and 8 debris. Furthermore, depending of the case which is being studied, the service time and the maximum mission time are modified in

this part of the script. The service time is the time that the chaser needs to complete the installation of the deorbit tool on the debris. The maximum mission time is the bound which differ from acceptable solutions and unacceptable. It must be noticed that while the maximum mission time has only been dependant of the number of debris to be deleted, the service time could change for a fixed number of objects to delete too.

3.7.2 Constants & Adimensionalize variables

This part of the code is in charge of two very different things that must be done at the beginning of the code: setting the value of the constants involved in the problem and the adimensionalizing some of the variables previously introduced. However, these two are closely related as the adimensionalisation is made with these magnitudes.

The two constants are the radius of the planet and the gravitational parameter, corresponding both to the ones of Earth. As said, with these constants the reference values of the velocity, service time and maximum time are obtained. It has to be remembered that the data imported from the txt file is all without dimensions, therefore, this step is crucial for the correct functioning of the algorithm.

3.7.3 Algorithm configuration

The next step in the code is to insert some algorithm inputs. In the lines corresponding to this section, the population size is introduced. As it will be seen in the following chapter, this population has been changed several times, to study its effects on the GA efficiency. In addition, the number of iterations is set, which in fact, are changed in the present work in order to study its effect.

Moreover, another input that the user set is the number of times the genetic algorithm is run. Increasing this number has as consequence an increase in the computational time, and the obtaining of more independent solutions between each other.

Lastly, the number of initializations for each individual is also introduced. The role of this parameter will be explained deeper in the following sections. For this option it must be outlined that as the number of initializations rises the computational time does the same, while being a random reinitialization of the population.

3.7.4 Sanity checks & Creation of matrixes

After the algorithm's configuration is set, some sanity checks are carried out in order to a complete assurance of the correct functioning of the code. Some redundancy in the code is introduced, as is in the cases of the population size and the number of iterations. With it, the population is always multiple of 4, while the number of iterations is always an integer positive, even though the user made a mistake introducing wrong inputs.

Besides, some flags have been introduced into the structure *defaultConfig*, which control the correct introduction of the necessary inputs and that the population is compatible with the number of iterations.

Apart from that, the data which were stored into the different vectors is transferred to matrixes. So, now in these matrixes the number of column and line correspond to a pair out of 24045 possible pairs. Finally, before starting with the principal code, the dimensions of some other matrixes are defined, and filled of zeros. These are some very important matrixes whose names and characteristics are:

- ***newPop***: Matrix used as a tool in order to obtain new individuals, and later transcript it to the final population.

- ***totalDist***: Vector that contains the fuel consumption of every individual for each iteration.
- ***totalTime***: Vector that contains the mission time of every individual for each iteration.
- ***distHistory***: Matrix that contains in each line the fuel consumption of the solutions for each iteration. The number of lines is the number of times the code has been looped.
- ***timeHistory***: Matrix that contains in each line the mission time solutions for each iteration. The number of lines is the number of times the code has been looped.
- ***Result_History***: Matrix that contains the individuals that fulfilled the maximum mission time requirement. If there is no solution, this matrix is filled of zeros.
- ***resultMatrix***: Matrix that contains, for each loop, the best result, its time, its individual and the iteration where the best solution was found. All the matrix is adimensionalized.

With all this, the principal code is executed, code which is explained briefly in the next subsection.

3.7.5 Principal code

Firstly, the code is inside a loop so that it is run several times in order to achieve different results without having to run again the code.

Once this said, as was explained in previous sections, the genetic algorithm starts with the initialisation of the population. The population is fill up with random permutations of numbers for each individual. Nevertheless, if the number of initialisations, previously explained input, is set greater than zero an additional process must be considered. This is, each member of the initial population is evaluated, and if it does not fulfil the maximum mission time limitation, is reinitialized again the number of times before introduced. When it does fulfil the time limitation, that individual is fixed as initial population. While performing such a process the initialization keeps always the individual with the lowest mission time. So, unless the number of initialisations is reached, the new initial population starts with acceptable solutions for the problem. Even though in this reinitialization process a sequence of debris that fulfils the time requisite is not found, the new initial population has a lower mission time, so improves the initialization of the algorithm. However, this process needs computational time, fact that will be studied in the following chapter.

After the initialization, there is another new loop that lasts until the end of the genetic algorithm. This is the iterations loop, which englobe all the following steps, as all the following functions will be done for each iteration.

Firstly, the current population is evaluated with a subroutine of evaluation that will be detailed. Afterwards, the obtained results are stored in a matrix, which contains the best result for each probe. This matrix is rewritten for each iteration with new or the same values. Such a matrix contains the following values in its structure:

- 1st column: Propellant consumption.
- 2nd column: Time to perform the sequence.
- 3rd column: Best iteration.
- 4th column: Index of the best individual.
- 5th - … columns: Debris sequence to be deleted.

The next step is to change the initial and evaluated population into a new one in order to obtain improvements in the result, in the most efficient way. Therefore, several techniques are applied which will be explained later. These strategies take part of the new population as follows:

Strategy	Percentage of new population
PMX crossover	30 %
Bubble sorting	20 %
Mutation of best ones	10 %
Single point crossover	10 %
Double point crossover	10 %
Inversion	10 %
Random insertion	10 %

Table 3.1 – Percentage of strategies in the code

As said before, this new population changed to be the population and begin again the same process just described. However, before looping another prevention is taken, due to the possibility of the algorithm to get homogenised. The algorithm eliminates some part of the population when half and three quarter of the total number of iterations are reached.

Now out of the iterations loop, the outputs are returned and printed as a graph. It must be reminded that this is done for each probe, so after this, the loop of the probes ends. Hereunder, the description of the different subroutines that were mentioned and also those not mentioned are explained deeper.

In order to change the population, we can distinguish 6 different functions, each one corresponding to a strategy showed in table 3.1. Firstly, the PMX crossover function is responsible of changing the population just as explained in the previous sections. Two other crossovers subroutines are encoded, which were explained too in other subsections, the *SinglePointCrossover* and the *DoublePointCrossover*. Apart from them, there is a subroutine for a simple inversion of two elements called *InvertParent*. Furthermore, the function *Mutation* consists on the mutation of the first two genes for half of the population considered, and mutation of the last two genes for the other half. The changes in these positions is done randomly, with the possibility of repeating two genes in the same individual. In addition, the function called *BubbleSorting* consists on the evaluation of different combinations of the population, so they are sorted, and the best ones have greater probability of been selected.

Besides, the most important function of the algorithm is the one that evaluates the population each time and provides the fitness function value. This one is called *EvaluatePath*, and the same function but just for the initial population is referred as *EvaluatePath_Initial*. Such a function gets a population and evaluates each individual's fitness value, giving as outputs a vector with these values, of dimension equal to the dimension of the population. This evaluation considers that in the sequence of removal, the last object must have the greater time to be removed. If this is not fulfilled, the RAAN parameter is added until this happen, normally providing very high mission times. Furthermore, if the mission time of the sequence is greater than the maximum time allowed, a penalisation is introduced to the fitness function, adding the mission time to the fitness function. Therefore, the solutions with less mission time are better if the mission lasts more than the maximum mission time. However, if is less, the fitness is not affected, so the best solutions are those with less consumption of fuel.

This explained evaluation is the normal mode of the algorithm. Nevertheless, for the studies that will be presented, the algorithm was changed. When wanting to prioritise the mission time over the fuel consumption, the fitness function only involved the sequence time. Everything else is the same as said in the previous paragraph, although some other changes can be implemented as limiting the fuel consumption.

All this said, there is only one more function that is introduced all through the code, named *ChromosomeHealing*. This one takes an individual and check that all the chromosomes in it are different from each other, because there is no sense in deleting two times the same object. If a chromosome is repeated, it is changed randomly and checked that the new one is not repeated. This chromosome healing can be considered as a sanity function in order to the correct functioning of the code.

CHAPTER 4

Results

The present chapter will analyse the results provided by the algorithm previously explained in chapter 3. Moreover, the problem that is being solved is described in chapter 2, and will be focused in 3 different cases, which are: Removing 4 debris, 5 debris and 8 debris. As the number of debris varies, some other parameters will be changed, which will be mentioned in the following subsections.

Apart from that, the versatility and feasibility of the genetic algorithm will be studied for some of the cases, in order to achieve the best configuration of the algorithm so the problem is solved as fast and efficiently as possible. Must be noticed that the reason why this algorithm is being used is to avoid the inspection of all the possible combinations for each problem, which follow the next expression:

$$Sequences = \frac{m!}{(m-n)!} \tag{4.1}$$

where *m* is the number of total debris that can be removed, and *n* is the number of debris that will be deleted that depends on the different cases.

4.1 Case 1: 4 Debris

Firstly, the objective is to remove 4 debris out of 155 possible debris to be removed. So, applying equation 4.1, the possibilities are:

$$Sequences = \frac{155!}{(155-4)!} = 5.55 \cdot 10^8 \tag{4.2}$$

The parameters previously stablished of the problem are fixed, with independence of the case being studied, while for the following parameters the values are:

$$T_{service} = 20\ days \tag{4.3}$$
$$T_{mission,max} = 180\ days \tag{4.4}$$

4.1.1 Efficiency of the algorithm

The first thing to be done is to study the efficiency of the algorithm in order to determine which is the best combination of the parameters that provides relevant results in an assumable execution time. This tuning process leads to an optimal number of iterations as well as dimension of population to be explore in each probe. Furthermore, another configuration is also considered, which consists on reinitialising the initial population so that the initial population comprises individuals with a sufficient quality.

Must be said that the important thing is that there is convergence. This means, running the algorithm provides a result that fulfils the temporary limit. To be highlighted the fact that this does not mean that the best solution is reached, but a good sequence is found.

In this line, it is proceeded to the study of the number of iterations for each probe. The population dimension, which is fixed, is 2000. As the population is fixed the dimension of it is supposed to not affect the comparison of iterations. Moreover, results have also been compared with the same configuration but with the reinitialization of the population 100 times. As previously explained,

the population is reinitialised until the population fulfil the condition that the debris are removed in less time than the maximum mission time.

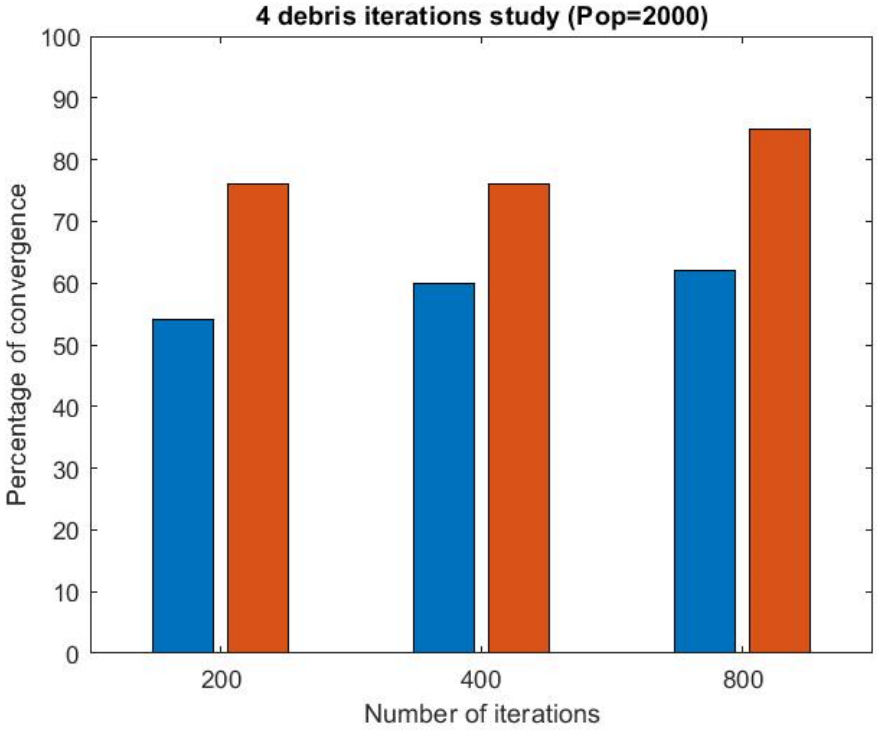

Figure 4.1 – Iterations study, 4 debris

As it can be seen above, there is a little improvement while increasing the number of iterations. However, the improvement of the percentage of convergence is not so great to be considered feasible, due to the increase of the execution time. With this, a better solution could be running more times the program with a lower number of iterations. Despite decreasing the number of iterations seems to be the best option, when the number of iterations is too small the algorithm does not function correctly, and therefore, the configuration chosen is of 200 iterations.

On the other hand, the improvement of the results when applying a reinitialization of the initial population seems to be of great importance. Indeed, the execution time also rises up, but not so high as it happens when increasing iterations. Thus, the reinitialization of the initial population can be considered as a way to improving the algorithm.

The next step is to study the influence of the dimension of the population. For doing so, the iterations are fixed in 200 iterations and a variable population dimension is introduced.

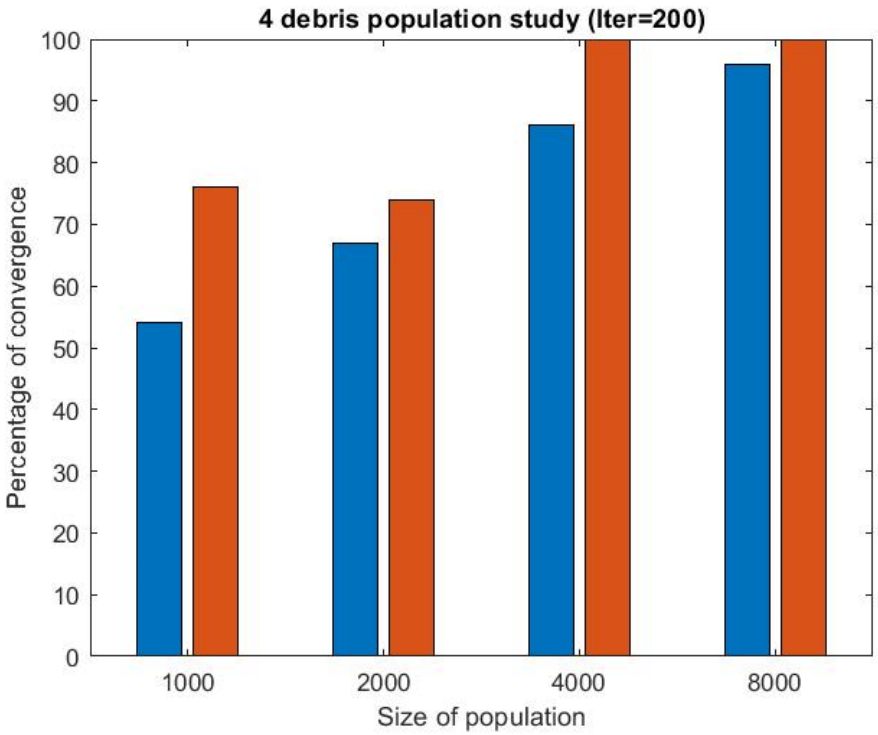

Figure 4.2 – Population study, 4 debris

Notwithstanding the increase in the execution time of the algorithm, the results are clearly improved in terms of convergence percentage, so increasing the population appears to be a good option. Apart from that, again, the reinitialization of the population improves the algorithm efficiency, reaching a hundred per cent of convergence for 4000 individuals. All this said, the configuration of 4000 individuals with reinitialization looks like the best option.

In figure 4.3 and figure 4.4 the influence of the number of times that the initial population is again initialised is analysed. For such a study the parameters of the algorithm are a population of 2000 and 400 iterations, which remain fixed in order to isolate the effects of the reinitialization. Must be reminded that except for this time, the other studies od the reinitialization are performed with a fixed number, 100 times.

As it can be easily seen, there is a small improvement in the results when individuals are initialized 100 times. From then on, there is a marked improvement, reaching an efficiency of 100% when redoing 10000 times. However, in figure 4.4 can be appreciated that the computational cost increases enormously. This is the reason why until 1000 times, the reinitialization is feasible while it is not worthy to increase this number too much.

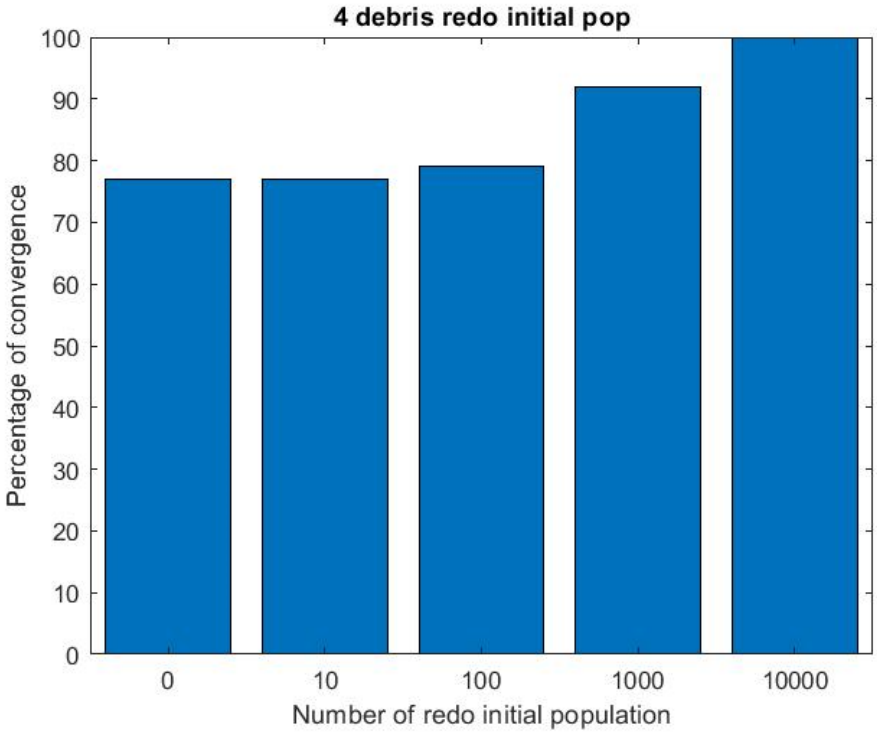

Figure 4.3 – Reinitialization of initial population study, 4 debris

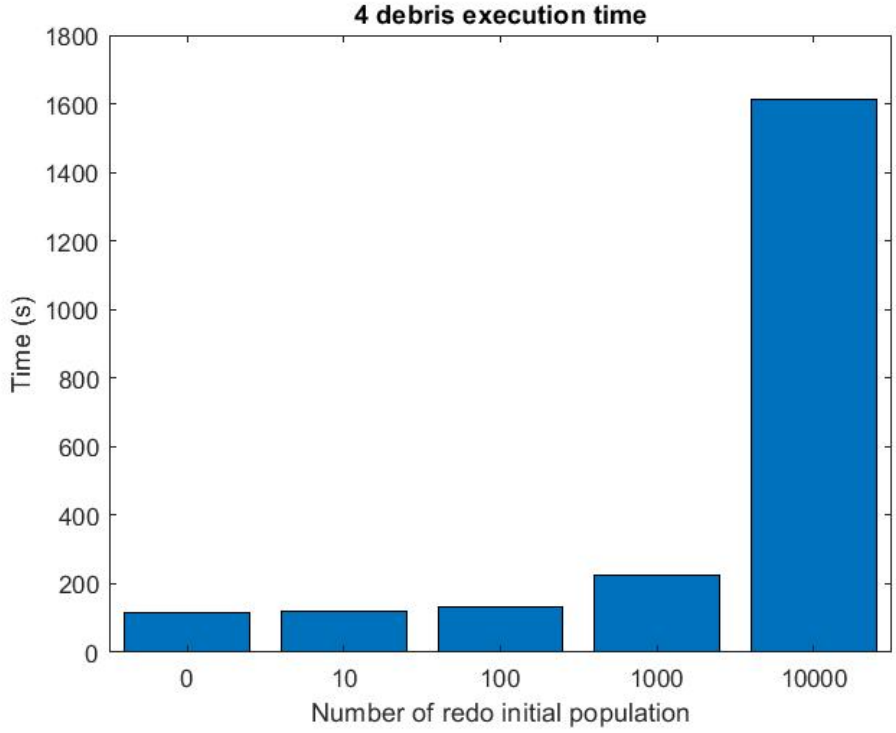

Figure 4.4 – Reinitialization of initial population time, 4 debris

A table with the results of the different combinations tried is presented above, in which 16 different configurations are studied.

Population	Iterations	Pop Reinitialized	Execution Time	Convergence %
1000	200	No	18	54
1000	200	100 times	20 (+11%)	76
1000	400	No	33	60
1000	400	100 times	36 (+9%)	75
1000	800	No	66	62
1000	800	100 times	70 (+6%)	86
2000	200	No	47	67
2000	200	100 times	61 (+30%)	74
2000	400	No	96	77
2000	400	100 times	117 (+22%)	79
2000	800	No	190	81
2000	800	100 times	223 (+17%)	95
4000	200	No	166	86
4000	200	100 times	185 (+11%)	100
8000	200	No	568	96
8000	200	100 times	581 (+2%)	100

Table 4.1 – Algorithm efficiency, 4 debris

Some conclusions can be drawn from the results presented for the case of removing 4 spatial objects. At first glance, it can be said that the population reinitialization is absolutely worthy, as the convergence increases quite a lot and the execution time is just a little bit higher. In addition, the increase of iterations does not give much better solutions while the execution times increases rapidly. Therefore, if all the previously said is taken into account, the configuration of 200 iterations and 4000 individuals provides a marked improvement in the algorithm studied.

4.1.2 Possible Sequences

With the limitation in time of 180 days, all the possibilities found are shown in figure 4.5. Each point represents a possible sequence of debris elimination.

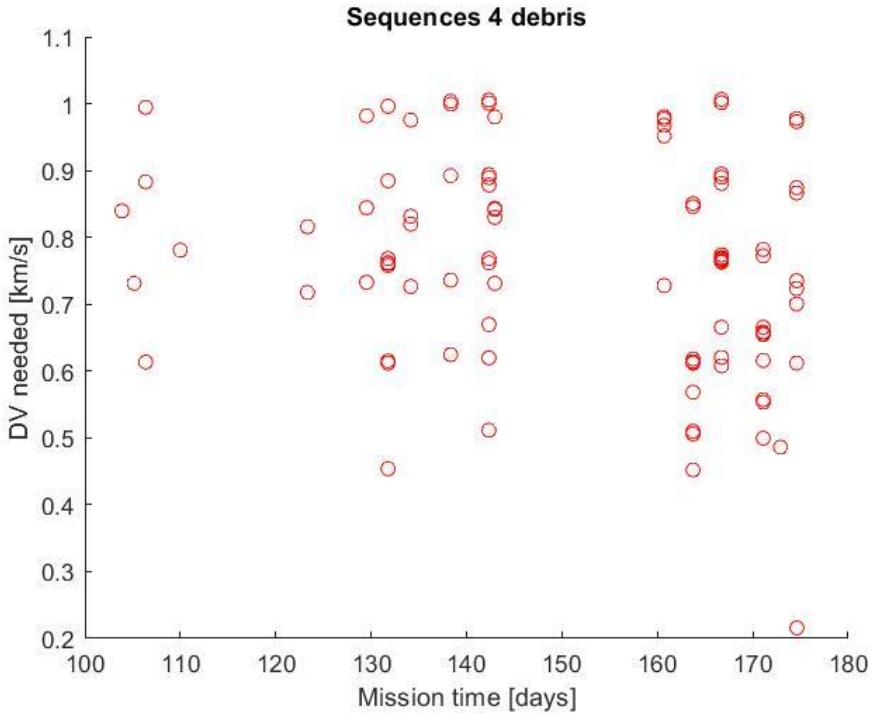

Figure 4.5 – Possible sequences, 4 debris

Also, it is important to say that some points could be overlapped because although they correspond to different sequences, both parameters that are represented are close to be the same. It depends on the criteria used for the selection of the best options that the best sequences could be on the left of the chart or at the bottom of it. Furthermore, it cannot be stated that these are all the possible solutions to the problem of removing 4 spatial debris, as all the spectral space is not explored.

Whereas for the results of this case, it can be noticed that there are no sequences that can perform a mission of removal of 4 debris in less than 100 days. Furthermore, the ΔV needed is always lower than 1.1 *km/s*, and almost everyone needs a ΔV lower than 1 *km/s*. Moreover, there is a marked difference between what seems to be the best point in terms of ΔV and the other ones. Due to the difficulty of exploring the spectral space, as it has been explained in previous chapters, some probes have been done initialization from the best solution to check that it was indeed the best one.

Last but not least, it must be highlighted that the number of sequences found are 100, and of those 94 are done with a ΔV lower than 1 *km/s*. Furthermore, just 44 of those sequences are able to remove the mentioned debris in less than 150 days, which means that most of the solutions are very near to the time limitation. The influence of this time limitation can be clearly seen when it is deleted. Without this condition there are several results from which the best one is performed with a ΔV of 0.0025 *km/s* in 170 years. Despite being unfeasible due to the durability of the mission this result shows the big influence of the time limitation.

4.1.3 Best Results

In this section tables the top 10 sequences are shown, according to two different criteria, minimisation of the time and minimisation of the fuel consumption.

ΔV [*km/s*]	ΔT [*days*]	1	2	3	4
0.2158	174.6206	60	4	133	84
0.2158	174.6211	109	37	130	146
0.4522	163.7354	4	60	151	93
0.4537	131.7702	7	151	60	3
0.4862	172.8901	60	4	133	1
0.4996	171.0915	4	60	151	3
0.5059	163.7354	103	2	151	93
0.5100	163.7354	7	2	151	93
0.5114	142.3484	7	151	2	93
0.5534	171.0915	103	2	151	3

Table 4.2 – Best sequences minimising ΔV, 4 debris

As it can be seen in order to minimise the fuel consumption the time of the mission is risen. To be noticed that almost every sequence of the top 10 implies a mission duration in the neighbourhood of the time limitation, 180 days. However, the forth best sequence in terms of propellant consumption, seems to be a great answer as it also a good sequence in terms of time of the mission just lasting 131 days. Thus, consuming 50% more of propellant the mission could be shortened by more than one month. As the time of the mission is shortened, saving money for the control operations is highly probable. It must be considered that all the possible best sequences involve almost the same debris, permutating them and introducing a couple of new ones. In this line, it could depend if there is a specific object as an objective that the best solution could change a lot.

The next table displays the best results referred to the mission time. To be reminded that the problem is the same, but in this case, there are several changes in the algorithm, so the objective function priority is the time of the mission.

ΔV [*km/s*]	ΔT [*days*]	1	2	3	4
0.8395	103.8730	7	133	60	151
0.7312	105.1597	60	4	133	26
0.6134	106.3745	151	7	2	60
0.8830	106.3745	133	7	2	60
0.8830	106.3745	151	133	60	2
0.9944	106.3745	7	133	60	2
0.7810	110.0075	133	7	2	151
0.7177	123.3291	60	4	133	50
0.8158	123.3291	26	1	50	133
0.7328	129.5228	60	4	133	84

Table 4.3 – Best sequences minimising ΔT, 4 debris

At first glance it can be said that the sequences have nothing to do with the best ones of propellant consumption. Just as before, the results are not very different between each other talking about their sequences, but they are a permutation between 6 objects except for a couple of them. Furthermore, some of them end with the same object but are not performed in the same time. This can occur due to the reasons which were explained in previous chapters. Besides, as happened before, the third best option saves 20% of the propellant and only lasts 3 days more, being the best one in the relation fuel/time.

With all this said, the best sequence depends of the requirements of the mission. This is, if the most important thing is to save fuel, the first option of table 4.2 would be the best choice. On the other hand, if the objective is to perform it as fast as possible the first option of table 4.3 would be the one that fits more the interests. Not to be overlooked that if there is a specific object that must be deleted, the sequences change. However, some good choices that at first glance could not seem the best ones are the third of table 4.3 and the fourth of table 4.4.

4.2 Case 2: 5 Debris

In the second case the objective is to remove 5 debris out of 155 possible debris to be removed. Again, applying equation 4.1, the possibilities are:

$$Sequences = \frac{155!}{(155-5)!} = 8.38 \cdot 10^{10} \tag{4.5}$$

The parameters are the same without any change, except for the maximum time of the mission, which changes from the previous problem of 4 objects.

$$T_{mission,max} = 300\ days \tag{4.6}$$

4.2.1 Efficiency of the algorithm

As done before, the efficiency of the algorithm but in this case with 5 objects is studied in order to determine which is the best combination of the parameters that provides relevant results in an assumable execution time. So, the number of iterations, dimension of the population and if it is better to reinitialise the population are determined again. It can be predicted that the behaviour of each characteristic will be similar to the 4-object case, although for sure the numerical results will vary largely.

It must be remembered that the important thing is the percentage of convergence. This means, how many times the algorithm can find at least a result within the temporary limit of 300 days. However, the percentage could be high, and the best result could not have been reached, as this algorithm does not provide evidence of achieving the best possible sequence.

In this sense, it is proceeded to the study of the number of iterations for each probe. The population dimension, which is fixed, is 2000. As the population is fixed the dimension of it is supposed not to be affecting the behaviour of the variance in the iterations. In addition, results have also been compared with the same configuration but with the reinitialization of the population 100 times. As previously explained, the population is reinitialised until the population fulfil the condition that the debris are removed in less time than the mission time max. If in 100 times this condition is not fulfilled, the algorithm will continue executing with the population that provides a lower mission time.

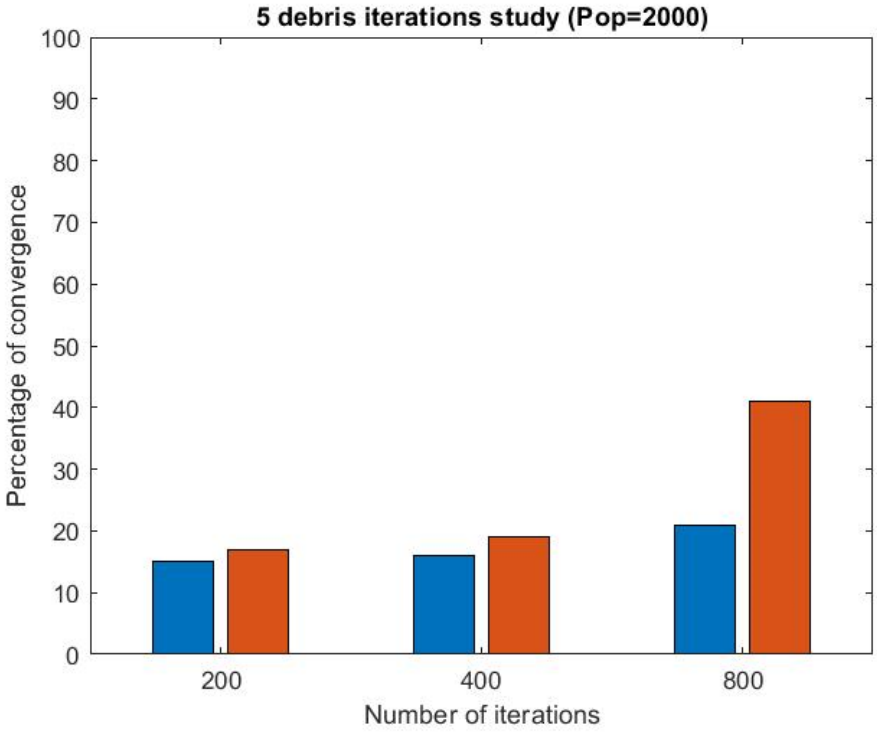

Figure 4.6 – Iterations study, 5 debris

In figure 4.6 it can be appreciated a very little improvement while increasing the number of iterations. However, the improvement of the percentage of convergence is not so great to be considered feasible, due to the increase of the execution time and because seems insignificant. Therefore, more solutions could be reached by running more times the algorithm with lower numbers of iterations, as the execution time would remain almost the same. Again, notwithstanding decreasing the number of iterations seems to be the best option, when the number of iterations is too small the algorithm does not function correctly, hence, the recommended configuration is of 200 iterations.

On the other hand, there is a marked improvement of the convergence percentage when applying a reinitialization to the population for the case of 800 iteration. This leads to an increase in the execution time, however, is not very high. Thus, the reinitialization of the initial population can be considered as a way to improving the algorithm.

Now the influence of the number of individuals is analysed. To do so, the iterations are fixed in 200 iterations and a variable population dimension is introduced.

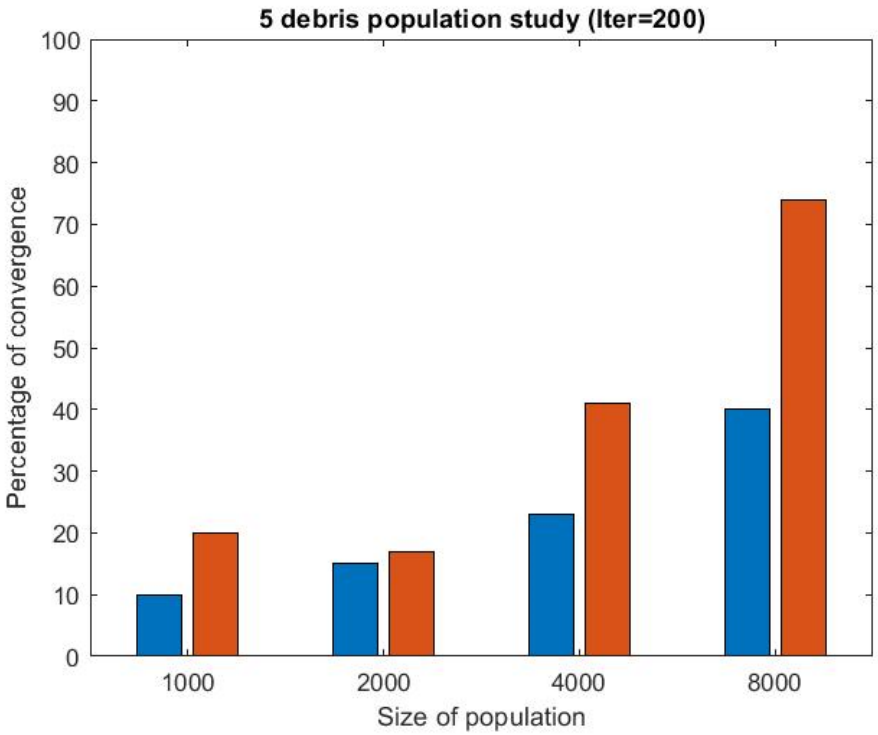

Figure 4.7 – Iterations study, 5 debris

At first glance, in figure 4.7 a more significant improvement can be observed when varying the number of individuals, in fact, the percentage reaches 40%. Thus, the behaviour of the convergence seems to be great with the increase of population. Besides, again, the reinitialization of the population improves the algorithm efficiency, reaching a 70 per cent of convergence for 8000 individuals. Nevertheless, as later in table 4.4 will be shown, the increase of execution time makes this configuration not to be as good as the one of 4000 individuals with reinitialization.

Figures 4.8 and 4.9 show the influence of the number of times that the initial population is reinitialised. To be highlighted that until now, the redo of the initial population was done just 100 times. For such a study the parameters of the algorithm are a population of 2000 and 400 iterations, which remain fixed in order to isolate the effects of the reinitialization.

Observable is the insignificant improvement of the percentage of convergence until 100 times that the population is redone. From then on, there is a marked improvement, reaching an efficiency of nearly 100% when redoing 10000 times. However, in figure 4.4 can be appreciated that due to the computational cost this option is not feasible. This is the reason why until 1000 times, the reinitialization is feasible while it is not worthy to increase this number too much. Another important point is that redoing 100 times the initial population do has a very positive effect when applying in bigger populations.

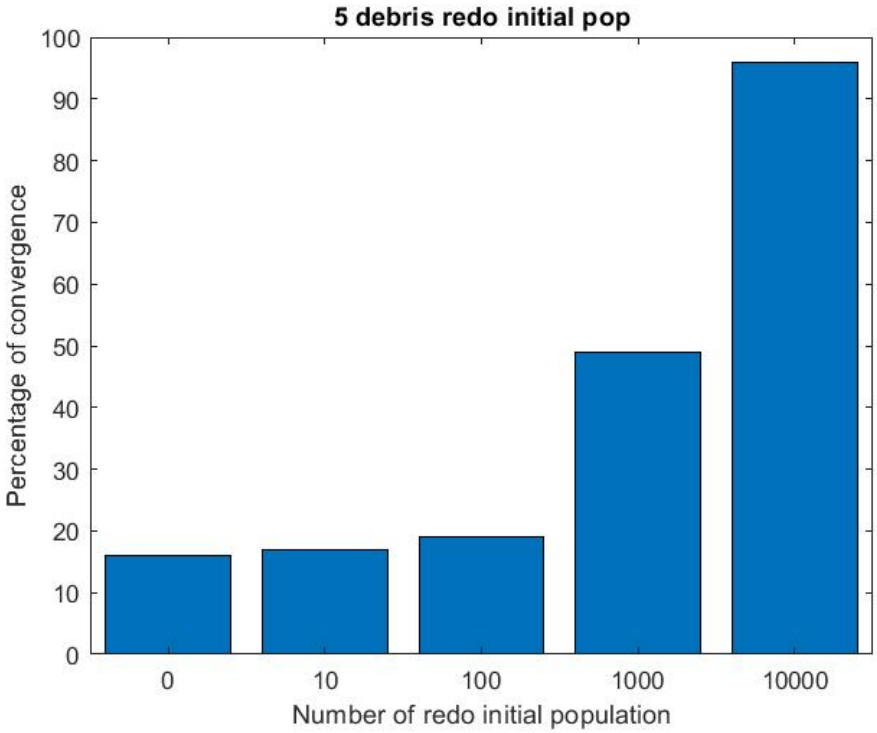

Figure 4.8 – Reinitialization of initial population study, 5 debris

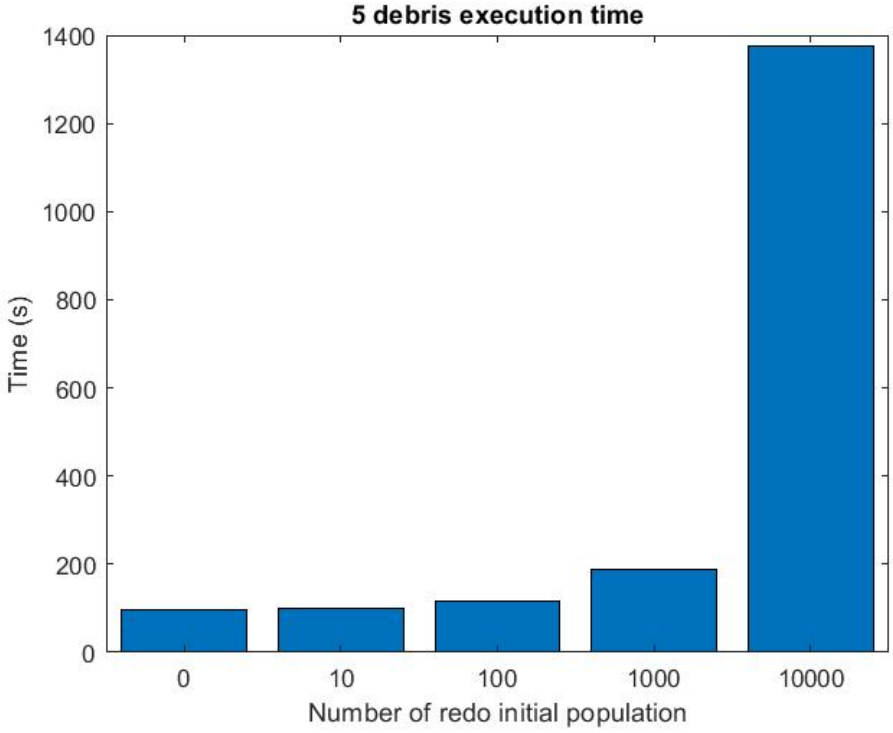

Figure 4.9 – Reinitialization of initial population time, 5 debris

So, table 4.4 presents all the different combinations tried, in which as done with the first case, 16 different configurations are studied.

Population	Iterations	Pop Reinitialized	Execution Time	Convergence %
1000	200	No	21	10
1000	200	100 times	27 (+29%)	20
1000	400	No	42	11
1000	400	100 times	47 (+12%)	17
1000	800	No	86	9
1000	800	100 times	90 (+5%)	19
2000	200	No	62	15
2000	200	100 times	73 (+18%)	17
2000	400	No	117	16
2000	400	100 times	132 (+13%)	19
2000	800	No	236	21
2000	800	100 times	260 (+10%)	41
4000	200	No	175	23
4000	200	100 times	220 (+26%)	41
8000	200	No	607	40
8000	200	100 times	621 (+2%)	74

Table 4.4 – Algorithm efficiency, 5 debris

The first conclusion that can be drawn is that in this case is more difficult for the code to find a sequence that fits the requirements. At first glance, it can be said that the population reinitialization is worthy when is applied in larger number of individuals, as the convergence increases quite a lot and the execution time is just a little bit higher. Moreover, whilst the number of iterations increases the execution time increases much more. Besides, despite of seeming a population of 8000 to be the best option, it is the configuration of 4000 individuals the one that presents the best ratio between convergence/execution time. All considered, the best choice is the one of a population of 4000, 200 iterations and reinitialization of the initial population.

4.2.2 Possible Sequences

In this case the time limitation is of a maximum mission time of 300 days. All the different sequences that the algorithm was able to find are printed in figure 4.10. In the graph each red point represents a sequence of objects that is removed, printing two parameters which are the fuel consumption, ΔV, and the time that lasts to do so.

It must be mentioned that there is great difficulty when trying to distinguish between some sequences that have very similar or identical parameters. Depending on the criteria used for the selection of the best options that the best sequences could be on the left of the chart or at the bottom of it. Besides, the algorithm does not assure that all the possible sequences are founded, however, it provides a wide group of good sequences to perform the mission.

In this case, 488 different sequences were found with a consumption of less than 1.3 *km/s*, and that fulfil the maximum time bound. Moreover, of all these sequences 191 have a ΔV lower than 1 *km/s* while only 52 sequences are performed in less than 200 days. This last thing implies that most of the sequences are close to the mission time limit.

So, obviously, the maximum time bound does have a great influence in the problem. By deleting the mission maximum time, the only parameter to be minimised is the propellant consumption, so other new sequences are found. Under these new conditions 5 objects could be removed in 384 years with a ΔV lower than 0.004 *km/s*. Nevertheless, must be outlined that, as previously happened in the first case, it is not a feasible option as the duration of the mission is too long to a real application.

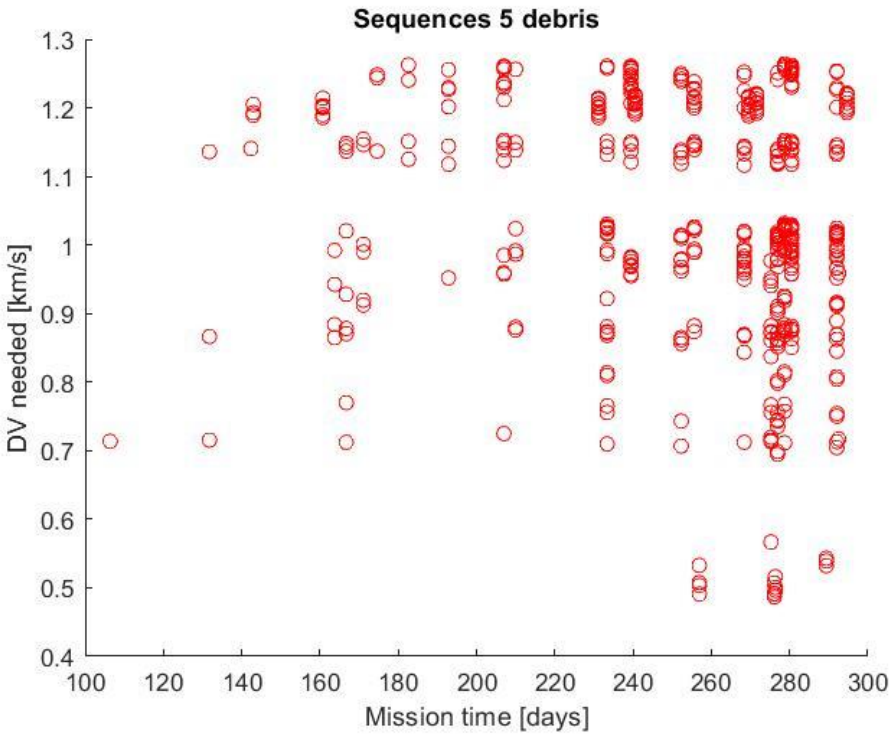

Figure 4.10 – Possible sequences, 5 debris

Graphically can be easily seen that there is no a sequence of objects that results in a mission with a lower mission time of 100 days. It can also be appreciated a concentration on the top right part of the graph, which are amongst the possible solutions the worst ones. Due to the difficulty of exploring the spectral space, as it has been explained in previous chapters, some probes have been done initialization from points that were of interest, such as the best sequences in terms of propellant consumption and in terms of time.

4.2.3 Best Results

The following section consists on representing in tables the top 10 sequences, according to two different criteria, minimisation of the mission time and minimisation of the propellant consumption.

ΔV [*km/s*]	**ΔT [*days*]**	**1**	**2**	**3**	**4**	**5**
0.4870	276.0855	94	32	71	42	9
0.4909	276.0855	32	94	102	42	9
0.4909	256.9440	32	94	102	42	132
0.4917	276.0855	32	94	71	42	9
0.4964	276.3678	94	32	71	42	48
0.5004	276.3670	32	94	102	42	48
0.5035	256.9446	94	32	71	42	132
0.5067	276.0855	71	94	102	42	9
0.5075	256.9446	32	94	102	42	132
0.5162	276.3670	71	94	102	42	48

Table 4.5 – Best sequences minimising ΔV, 5 debris

Firstly, the sequences that need less propellant consumption are all in the neighbourhood of the mission maximum lifetime, 300 days. There is just a difference of at most 20 days between all of them. Apart from that, the ΔV is very similar in all of the different sequences, being much higher

than the previous case as there is a need of performing an extra manoeuvre. It must be considered that all the possible best sequences involve almost the same debris, permutating them and introducing a couple of new ones. So, in this case all the best solutions are very similar, so the choice depends on the criteria used or the objective or constraints.

ΔV [*km/s*]	ΔT [*days*]	1	2	3	4	5
0.7138	106.3745	103	151	7	2	60
0.7154	131.7702	103	151	7	3	60
0.8664	131.7702	151	7	2	60	3
1.1359	131.7702	133	7	2	60	3
1.1406	142.3484	151	133	60	2	93
1.2522	142.3484	7	133	60	2	93
1.2142	160.7062	26	1	84	133	49
1.1858	160.7062	133	84	4	89	2
1.1897	160.7062	133	7	2	93	3
1.2016	160.7062	151	133	26	2	89

Table 4.6 – Best sequences minimising ΔT, 5 debris

Comparing the sequences of table 4.6 with those of table 4.5, it can be said that they do not have nothing to do between each other. When prioritising the mission time, much more higher values of propellant consumption are achieved. In particular, it is appreciated that the sequence of objects that minimise the mission time also is the one that has the lower ΔV among the top10. Just as before, the results are not very different between each other talking about their sequences, but they are a permutation between 6 different objects except for a couple of them. Furthermore, some of them end with the same object but are not performed in the same time. This can occur due to the reasons which were explained in previous chapters.

Finally, to be highlighted the great increase referring to ΔV that takes place when changing from 4 objects to 5, while there is just a slight increase in the lifetime of the mission, just 3 days, when trying to minimising it.

4.3 Case 3: 8 Debris

In this case the objective is to remove 8 debris out of 155 possible debris to delete. One last time, applying equation 4.1, the possibilities are:

$$Sequences = \frac{155!}{(155-8)!} = 2.77 \cdot 10^{17} \tag{4.7}$$

For the removal of 8 objects, two different cases are studied with different times of service. The other parameters of the problem remain with the same value that the precedent cases, except for the maximum mission time.

$$T_{mission,max} = 365\ days \tag{4.8}$$
$$T_{service} = 5\ days \tag{4.9}$$
$$T_{service} = 10\ days \tag{4.10}$$

4.1.1 Efficiency of the algorithm

As the number of objects increases, the complexity of the problem becomes greater, hence, the computational cost is greater while trying the different sequences and in the process of optimisation. Apart from that, taking into account that the increase of the maximum mission time is just of 65 days when increasing the number of objects in 3, the possibility to find a sequence that fulfils all the constraints is reduced significantly.

This fact was clearly noted when the algorithm was run, and the algorithm was unable to find a suitable solution with most of the combinations in population and iterations that were studied in precedent sections. Therefore, it can be said that the algorithm does not function as good as wished for this specific case. However, a sequence of objects was found which was acceptable.

Unlike the cases of 4 and 5 objects, this time the study of the efficiency of the algorithm was unfeasible, as the time to carry it out would be enormous. The percentage of convergence is so low that it is strongly dependant of the randomness in each time the code is run. This is the reason why different configurations have been tried, and although some results were found, it is not showed the study of the algorithm.

It can be said that the algorithm is not very efficient in this case, nevertheless, this problem involves a lot of difficulty with lots of different possible sequences. For all this, a different strategy has been implemented in order to achieve better results. Such a strategy will be explained in the following chapter.

4.1.2 Possible Sequences

As previously mentioned, the algorithm was able to find a couple of solutions, but the execution time was enormous. Therefore, the strategy used to find more solutions was to initialise the population from previously found solution. This is, the code was run and after plenty of execution time a solution was found which fulfilled the time limitation, less than 365 days. Afterwards, the code was run again but this time with the initial population equal to the genes of the just mentioned solution. So, as was mentioned in chapter 3, the algorithm will change this population searching for better solutions with most of them similar to the initial population. Thanks to this strategy it was possible to find much more sequences and improve the mentioned result.

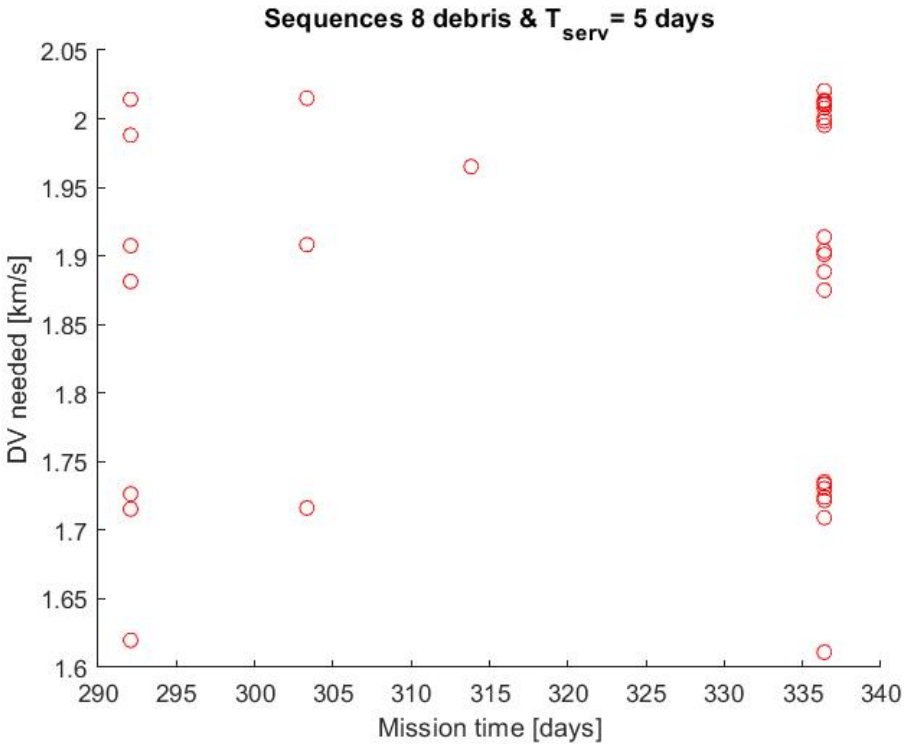

Figure 4.11 – Possible sequences, 8 debris

The sequences obtained for a service time of 5 days are 34. From those, 25 need a ΔV lower than 2 km/s and just 8 sequences need a mission time of less than 200 days. Of course, both parameters are greater in this case that in the studied before in this same chapter. Also, remarkable the fact

that can be appreciated that all the sequences would be performed in just 4 different times. This happens because all of them are permutations of very few genes and implies that the sequences only finish at most in 4 different objects, one per each mission time.

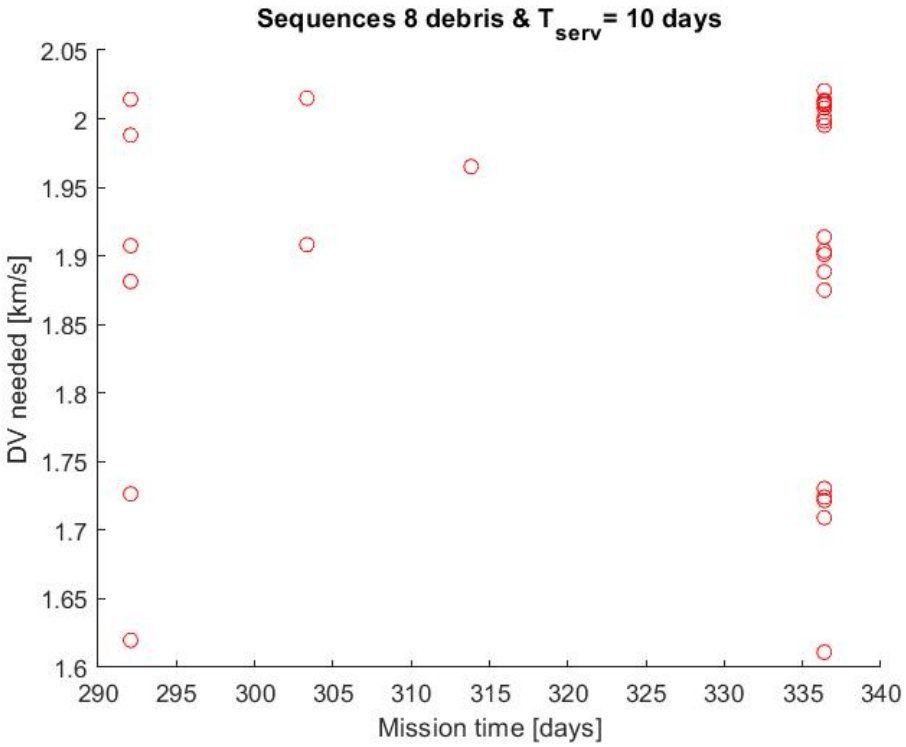

Figure 4.12 – Possible sequences, 8 debris

As can be seen in figure 4.12, the results are almost the same when the service time is doubled. This happens because for these sequences obtained, the increase of service has no influence, but as the service time increases less points would be seen in the graph. Thus, 31 results are achieved, 3 less than before. 22 of those need less than ΔV, which means that the three sequences that cannot be done belong to this group.

For both service times some conclusions can be drawn. As could be easily predicted, the sequences which can be done for 10 days are also acceptable results for a service time of 5 days. Apart from that, it can be observed that the propellant consumption doubles its precedents. Another noticeable point is that in both cases under study, there is a concentration of possibilities close to the time constraint.

4.1.3 Best Results

Now, the best results in terms of propellant consumption are presented. Firstly, with a service time of 10 days which is more restrictive, and afterwards with a service time of 5 days. Indeed, it is supposed to find all the best solutions of one of them in the other.

ΔV [*km/s*]	ΔT [*days*]	1	2	3	4	5	6	7	8
1.6110	336.3860	151	133	4	50	2	76	3	49
1.6197	292.0726	151	133	4	26	2	50	3	76
1.7091	336.3860	26	1	50	4	76	2	49	3
1.7217	336.3860	26	1	50	4	76	2	89	3
1.7217	336.3860	26	1	50	4	89	2	49	3
1.7241	336.3860	50	1	76	4	89	2	49	3
1.7264	292.0726	60	4	133	26	2	50	3	76
1.7264	292.0726	60	133	4	26	2	50	3	76
1.7304	336.3860	60	4	133	26	2	50	3	49
1.7304	336.3860	60	133	4	26	2	50	3	49

Table 4.7 – Best sequences minimising ΔV, 8 debris, Tserv=10

ΔV [*km/s*]	ΔT [*days*]	1	2	3	4	5	6	7	8
1.6110	336.3860	151	133	4	50	2	76	3	49
1.61973	292.0726	151	133	4	26	2	50	3	76
1.7091	336.3860	26	1	50	4	76	2	49	3
1.7154	292.0726	26	1	50	4	84	2	76	3
1.7162	303.3364	26	1	50	4	76	2	89	3
1.7217	336.3860	26	1	50	4	89	2	49	3
1.7241	336.3860	50	1	76	4	89	2	49	3
1.7265	292.0726	60	4	133	26	2	50	3	76
1.7265	292.0726	60	133	4	26	2	50	3	76
1.7304	336.3860	60	4	133	26	2	50	3	49

Table 4.8 – Best sequences minimising ΔV, 8 debris, Tserv=5

To be outlined that the best sequence in both cases is the same. Furthermore, the sequence that provides the best ratio between time and consumption, is also the same, the one located in second place. In fact, in this top only the one placed in forth position is just possible when the service time is 5 days. As previously said, all the sequences look similar as they are nearly a permutation between 8 genes. Also, as mentioned before, it is shown how the different sequences only finish in 3 different objects, with their correspondent mission time.

The next tables represent the best sequences of objects to be removed minimising the mission time, first for a service time of 10 days and afterwards of 5 days.

ΔV [*km/s*]	ΔT [*days*]	1	2	3	4	5	6	7	8
1.6197	292.0726	151	133	4	26	2	50	3	76
1.7264	292.0726	60	4	133	26	2	50	3	76
1.7264	292.0726	60	133	4	26	2	50	3	76
1.8814	292.0726	151	133	26	4	50	2	76	3
1.9075	292.0726	151	133	26	4	84	2	76	3
1.9881	292.0726	60	133	26	4	50	2	76	3
2.0141	292.0726	60	133	26	4	84	2	76	3
1.9083	303.3364	151	133	26	4	76	2	89	3
2.0150	303.3364	60	133	26	4	76	2	89	3
1.9652	313.8257	1	26	133	50	4	76	2	49

Table 4.9 – Best sequences minimising ΔT, 8 debris, Tserv=10

ΔV [*km/s*]	ΔT [*days*]	1	2	3	4	5	6	7	8
1.6197	292.0726	151	133	4	26	2	50	3	76
1.7265	292.0726	60	4	133	26	2	50	3	76
1.7265	292.0726	60	133	4	26	2	50	3	76
1.8814	292.0726	151	133	26	4	50	2	76	3
1.9075	292.0726	151	133	26	4	84	2	76	3
1.9881	292.0726	60	133	26	4	50	2	76	3
2.0142	292.0726	60	133	26	4	84	2	76	3
1.7154	292.0726	26	1	50	4	84	2	76	3
1.7162	303.3364	26	1	50	4	76	2	89	3
1.9083	303.3364	151	133	26	4	76	2	89	3

Table 4.10 – Best sequences minimising ΔT, 8 debris, Tserv=5

Not a lot of comments that have been done before can be said. With a service time of 5 days there is a new sequence with respect to the first table, which is in eight position. Apart from that, just as before said, the sequences are mostly a permutation between 8 different genes, and they only finish in 3 different genes. Finally, must be outlined that there is not great difference in time nor in propellant consumption, hence, all these solutions could be a good choice for the mission of removal of debris

CHAPTER 5

Conclusions & Future work

5.1 Conclusions

The thesis presented has consisted on the development and study of a genetic algorithm in the programming language *Matlab*, in order to obtain the best sequences to be deleted with a mission of active removal of space debris. The immediate conclusion that can be drawn is that the code has proven to be effective as it worked correctly while achieves the goal. Indeed, some comments must be outlined which are stemmed from the results shown in chapter 5.

In the first place, the evolutive algorithm has shown its capacity to adapt to different conditions of the problem. The algorithm has probed to work correctly for the removal of 4 and 5 space debris, changing the maximum mission time. Due to the difficulty to find solutions in the case of 8 debris, it was needed another approach when facing the problem, nevertheless, it can be said that the algorithm behaviour has been acceptable. Apart from that, it has also probed to reach solutions of the problem when changing the service time. In addition, it has also demonstrated that it reaches solutions when trying to minimise the mission time. Another important aspect of the code is that for the cases of 4 and 5 debris, it has been able to obtain high percentages of convergence, being even of 100% for the first case.

On the other hand, the algorithm has shown a great improvement with the reinitialization of the initial population. However, the more the number of reinitializations increases the more the computational time rises, and hence, this number cannot be very big. Instead, reinitialising each member of the population 100 times, it has shown a noticeable improvement while the computational time did not increase a lot. Moreover, in the case of 8 space debris, the implementation of the initialisation from a known valid sequence has been demonstrated to be absolutely efficient for the search of other valid solutions.

Finally, it must be highlighted that the algorithm does not provide assurance of having reached the best solution whatsoever. It has proved itself to be useful in finding good solutions for active debris removal missions but not the best one. Therefore, each time the algorithm is run it may give a different result, but as its objective is to provide a wide range of good sequences, the algorithm seems to be suitable for its function. For all this, the choice of an evolutive algorithm in order to find the sequence of debris to be followed in ADR missions seems to be feasible and effective.

5.2 Future work

Firstly, although the algorithm showed good functioning under different conditions, more tests are to be done in order to validate the versatility of the code. As it was explained, depending on the numbers of debris to be deleted, the algorithm was able to reach solutions easily or not. Thus, the number of debris until the genetic algorithm is feasible could be studied. It must be reminded that while for the cases of 4 and 5 debris it reached acceptable solutions, when considering 8 debris it is not able to find a sequence rapidly. Apart from that, new solutions can be obtained, as this work does not assure that all the sequences were obtained, nor the best sequence for the removal of the space debris

Secondly, the parameters of the genetic algorithm could be modified, so the genetic algorithm is improved for this type of problem. This is, change the parameters such as the contribution of mutation, crossover, strategies used… Some important aspects that could be analysed with the new configuration are the computational time, the variety in the solutions and if it is able to reach the best solution, and in the best a better one. What is more, the fitness function could be changed as well as the reinitialization could be improved or changed of criteria.

Moreover, the code is prepared for the implementation of other input files with a different number of possible debris to be deleted. Therefore, other targets could be studied if the input files are implemented correctly, as it should not be forgotten that the objective of this algorithm is to be as versatile as possible. In addition, this thesis only considered the variation of the ascending node, so the code could also be adapted to other problems in which the debris are not just dependant of this variation. However, if a propagator is introduced within the algorithm, the code would have to be drastically changed, hence, this is not the line of work recommendable for the developed genetic algorithm.

Finally, if the active debris removal mission is intended to be done, there are several studies that should be carried out. Despite this thesis has obtained the impulses, a study of the most suitable propellant should be done with the consequent contribution to the mass budget. Moreover, the deorbiting technology and strategy should be defined as it could change some details such as the service time. Furthermore, the data imported from the debris catalogue could be propagate accurately until the date of launching, in order to minimise the possible failure of the mission. Another important aspect is to decide which debris are to be deleted, as it could be possible that the best solution in terms of propellant is not the selected sequence. These are some of the most important aspects to care about, as well as, of course, the pertinent actions and studies corresponding to the development of a space mission.

Bibliography

[1] Casalino, L. and A.Vavrina, M. Optimal power partitioning for electric thrusters. Advances in the Astronautical Sciences 2018, Stevenson, WA, USA: AAS/AIAA Astrodynamics Specialist Conference.

[2] Shen, H. X., Zhang, T. J., Casalino, L., & Pastrone, D. (2017). Optimization of Active Debris Removal Missions with Multiple Targets. *Journal of Spacecraft and Rockets*, *55*(1), 181-189.

[3] Carolin Früh. (2011). Identification of Space Debris. *Astronomisches Institut der Universität Bern*

[4] Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, *1*(1), 53-66.

[5] Sentinella, M. R., & Casalino, L. (2009). Hybrid evolutionary algorithm for the optimization of interplanetary trajectories. *Journal of Spacecraft and Rockets*, *46*(2), 365-372.

[6] Casalino, L., Letizia, F., & Pastrone, D. (2014). Optimization of hybrid upper-stage motor with coupled evolutionary/indirect procedure. *Journal of Propulsion and Power*, *30*(5), 1390-1398.

[7] Pastrone, D., & Sentinella, M. R. (2009). Multi-objective optimization of rocket-based combined-cycle engine performance using a hybrid evolutionary algorithm. *Journal of Propulsion and Power*, *25*(5), 1140-1145.

[8] Razali, N. M., & Geraghty, J. (2011, July). Genetic algorithm performance with different selection strategies in solving TSP. In *Proceedings of the world congress on engineering* (Vol. 2, No. 1, pp. 1-6). Hong Kong: International Association of Engineers.

[9] Orbitaldebris.jsc.nasa.gov. (2019). *ARES: Orbital Debris Program Office*. [online] Available at: https://orbitaldebris.jsc.nasa.gov/. [Accessed Dec. 2018].

[10] Stuart, J., Howell, K., & Wilson, R. (2016). Application of multi-agent coordination methods to the design of space debris mitigation tours. *Advances in Space Research*, *57*(8), 1680-1697.

[11] Shen, H. X., Zhang, T. J., Li, Z., & Li, H. N. (2017). Multiple-hopping trajectories near a rotating asteroid. *Astrophysics and Space Science*, *362*(3), 45.

[12] Marin, A. (2018). *Ottimizzazione di missioni spaziali ADR mediante algoritmi genetici* (Doctoral dissertation, Politecnico di Torino).

[13] Marin, A. (2018). *Ottimizzazione di missioni spaziali ADR mediante algoritmi genetici* (Doctoral dissertation, Politecnico di Torino).

[14] European Space Agency. (2019). *About space debris*. [online] Available at: https://www.esa.int/Our_Activities/Operations/Space_Debris/ [Accessed Jan. 2019].

[15] Klinkrad, H. (2006). *Space debris: models and risk analysis*. Springer Science & Business Media.

[16] Larranaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., & Dizdarevic, S. (1999). Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, *13*(2), 129-170.

[17] Sivanandam, S. N., & Deepa, S. N. (2007). *Introduction to genetic algorithms*. Springer Science & Business Media.

[18] Blickle, T., & Thiele, L. (1996). A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, *4*(4), 361-394.

[19] Celestrak.com. (2019). *CelesTrak*. [online] Available at: https://www.celestrak.com/ [Accessed 2019].

[20] Larson, W. J., & Wertz, J. R. (1992). *Space mission analysis and design* (No. DOE/NE/32145-T1). Torrance, CA (United States); Microcosm, Inc.

[21] Fragmentation.esoc.esa.int. (2019). *ESA's fragmentation database*. [online] Available at: https://fragmentation.esoc.esa.int/home/statistics [Accessed Jan. 2019].

[22] Pavarin, Daniele & Francesconi, Alessandro & Branz, Francesco & Chiesa, Sergio & Viola, Nicole & Bonnal, Christophe & Trushlyakov, Valery & Belokonov, Ivan. (2012). *Active Space Debris Removal by Hybrid Engine Module.*

[23] Flury, W., & Contant, J. M. (2001, October). The updated IAA position paper on orbital debris. In *Space Debris* (Vol. 473, pp. 841-849).

[24] Zimper, Dirk & Göge, Dennis & Eckel, Hans-Albert. (2016). *Laser-Based Space Debris Removal - An Approach for Protecting the Critical Infrastructure Space*. The Journal of the JAPCC (Vol. 22, pp. 75-84).

[25] Fabacher, E., Lizy-Destrez, S., Alazard, D., Ankersen, F., & Profizi, A. (2017). Guidance of magnetic space tug. *Advances in Space Research*, *60*(1), 14-27.

[26] Jankovic, M., & Kirchner, F. (2018). Taxonomy of LEO Space Debris Population for ADR Capture Methods Selection. In *Stardust Final Conference* (pp. 129-144). Springer, Cham.

[27] Ecss.nl. (2019). *European Cooperation for Space Standardization*. [online] Available at: http://www.ecss.nl/ [Accessed Feb. 2019].

[28] Discosweb.esoc.esa.int. (2019). *Statistics - discos*. [online] Available at: https://discosweb.esoc.esa.int/web/guest/statistics [Accessed Jan. 2019].

[29] Ansdell, M. (2010). *Active Space Debris Removal: Needs, implications, and recommendations for today's geopolitical environment*. Journal of Public & International Affairs, 21.

[30] Klinkrad, H. (2006). *Space debris: models and risk analysis*. Springer Science & Business Media.

Appendix A: Matlab Code

```
function varargout = tspo_ga(varargin)

tic
```

Loading data

```
load kosmos.txt


departure_ID = kosmos(:,1);

arrival_ID   = kosmos(:,2);

dv           = kosmos(:,3);

dt           = kosmos(:,4);

d_omega      = kosmos(:,5);


total_debris     = 155;

Debris_to_remove = 5;   % Parameter to be changed (4 5 or 8)


time_service    = 20*86400;  % Parameter to be changed (5 10 or 20)

mission_timemax = 300*86400; % Parameter to be changed (180 300 or 365)


g  = 1;
```

Constants and times of references

```
mu = 398600.4415;

r  = 6378.1363;

v_ref              = sqrt(mu/r);

time_ref           = r/v_ref;

time_servref       = time_service/time_ref;

mission_timemaxref = mission_timemax/time_ref;
```

Configuration of the Algorithm

```
defaultConfig.popSize     = 1000;

defaultConfig.numIter     = 10;

defaultConfig. showProg   = true;
```

```
defaultConfig.showResult  = true;
defaultConfig.showWaitbar = false;


num_redo         = 1; % If there is no reinitialization of pop, set to 0.
numeroProve      = 2;


% Interpret user configuration inputs
if ~nargin
    userConfig = struct();
elseif isstruct(varargin{1})
    userConfig = varargin{1};
else
    try
        userConfig = struct(varargin{:});
    catch
        error('Expected inputs are either a structure or parameter/value
pairs');
    end
end


% Override default configuration with user inputs
configStruct = get_config(defaultConfig,userConfig);


% Extract configuration
popSize     = configStruct.popSize;
numIter     = configStruct.numIter;
showProg    = configStruct.showProg;
showResult  = configStruct.showResult;
showWaitbar = configStruct.showWaitbar;
```

Creation of matrixes & Sanity checks

```
% Sanity Checks
popSize     = 4*ceil(popSize/4);
numIter     = max(1,round(real(numIter(1))));
showProg    = logical(showProg(1));
```

```
showResult  = logical(showResult(1));
showWaitbar = logical(showWaitbar(1));
% Matrixes dmat, tmat, omegamat
dmat     = zeros(total_debris,total_debris);
tmat     = zeros(total_debris,total_debris);
omegamat = zeros(total_debris,total_debris);
row       = 1;
rowt      = 1;
row_omega = 1;

for fileCounter = 0:total_debris:length(departure_ID)-total_debris
    dmat(row,:) = dv(fileCounter+1:fileCounter+total_debris);
    row = row + 1;
end
for fileCounter_time = 0:total_debris:length(departure_ID)-total_debris

    tmat(rowt,:) = dt(fileCounter_time+1:fileCounter_time+total_debris);
    rowt = rowt + 1;
end
for fileCounter_omega = 0:total_debris:length(departure_ID)-total_debris

    omegamat(row_omega,:) =
d_omega(fileCounter_omega+1:fileCounter_omega+total_debris);
    row_omega = row_omega + 1;
end

% Creation of Matrixes
resultMatrix   = zeros(numeroProve,4+Debris_to_remove);
totalDist      = zeros(1,popSize);
totalTime      = zeros(1,popSize);
distHistory    = zeros(numeroProve,numIter);
timeHistory    = zeros(numeroProve,numIter);
Result_History = zeros(numIter*numeroProve,2+Debris_to_remove);
newPop         = zeros(popSize,Debris_to_remove);
```

Start with the GA

```
for prova=1:numeroProve
    prova
    bestIter = inf;
    % Initialize the Population
    pop = zeros(popSize,Debris_to_remove);

    for k = 1:popSize
        b = randperm(total_debris);
        c = b(1,1:Debris_to_remove);
        pop(k,:) = c(:);

        % Reinitialization of each member of population
        if num_redo>0
            [d_pop_initial] =
EvaluatePath_Initial(c,time_servref,mission_timemaxref,tmat,dmat,omegamat,Debr
is_to_remove,total_debris);
            counter_initial=0;
            while d_pop_initial>=1000 & counter_initial<num_redo
                d_pop_initial_vec(counter_initial+1)=d_pop_initial;
                counter_initial=counter_initial+1;
                b = randperm(total_debris);
                c = b(1,1:Debris_to_remove);
                [d_pop_initial] =
EvaluatePath_Initial(c,time_servref,mission_timemaxref,tmat,dmat,omegamat,Debr
is_to_remove,total_debris);
                d_pop_initial_min=min(d_pop_initial_vec);
                if d_pop_initial<=d_pop_initial_min
                    pop(k,:) = c(:);
                end
            end
        end
    end
```

Start iteration loop & Evaluate Each Population Member

```
    for iter = 1:numIter


[totalDist,minDist,index,distHistory,globalMin,bestIter,optRoute,totalTime,tim
eHistory,route_time,Result_History,g] =
EvaluatePath(pop,popSize,dmat,tmat,omegamat,numIter,iter,totalDist,distHistory
,totalTime,timeHistory,time_servref,mission_timemaxref,Debris_to_remove,prova,
total_debris,g,Result_History);
```

Write the results in a matrix

```
        format long

        resultMatrix(prova,1) = globalMin;

        resultMatrix(prova,2) = route_time;

        resultMatrix(prova,3) = bestIter;

        resultMatrix(prova,4) = index;

        resultMatrix(prova,5:4+Debris_to_remove) = optRoute;
```

Bubble Sorting

```
        [popSorted4Fitness,choice,roulette_it,totalTime] =
BubbleSorting(popSize,pop,totalDist,totalTime,Debris_to_remove);
```

APX crossover (PMX)

```
        [child1,child2] =
APX_Crossover(choice,popSorted4Fitness,roulette_it,Debris_to_remove,total_debr
is);


        newPop(1:0.3*popSize,:)             = [child1;child2];

        newPop(0.3*popSize+1:0.5*popSize,:) =
popSorted4Fitness(0.8*popSize+1:1:popSize,:);
```

Mutation of a part of the best vectors

```
        [child1M,child2M] =
Mutation(popSorted4Fitness,Debris_to_remove,total_debris,popSize);


        newPop(0.5*popSize+1:0.6*popSize,:) = [child1M;child2M];
```

Single point crossover

```
        [child1SP,child2SP] =
SinglePointCrossover(popSorted4Fitness,Debris_to_remove,total_debris,popSize);


        newPop(0.6*popSize+1:0.7*popSize,:) = [child1SP;child2SP];
```

Double Point Crossover

```
        [child1DP,child2DP] =
DoublePointCrossover(popSorted4Fitness,Debris_to_remove,total_debris,popSize);


        newPop(0.7*popSize+1:0.8*popSize,:) = [child1DP;child2DP];
```

Simple inversion of 2 elemnts over the 40 best vectors

```
        invertParent =
InvertParent(popSorted4Fitness,Debris_to_remove,popSize);


        newPop(0.8*popSize+1:0.9*popSize,:) = invertParent;
```

Introduction of random elements in the new population

```
        for t=0.9*popSize+1:popSize
            newPop(t,:) = randperm(total_debris,Debris_to_remove);
        end
```

Creation of new population

```
        pop = newPop;
```

Elimination of part of population for iter=0.5*numIter

```
        regCounter = 0;
        if iter == 0.5*numIter
            for regCounter=11:popSize
                pop(regCounter,:) = randperm(total_debris,Debris_to_remove);


            end
        end
```

Elimination of part of population for iter=0.75*numIter

```
        if iter == 0.75*numIter
            for regCounter=6:popSize
                pop(regCounter,:) = randperm(total_debris,Debris_to_remove);

            end
        end
    end
end
if showWaitbar
    close(hWait);
end

if showResult
    plot(distHistory.','x','LineWidth',2);
    title('Best Solution History');
end

% Return Output
if nargout
    resultStruct = struct( ...
        'popSize',     popSize, ...
        'numIter',     numIter, ...
        'showProg',    showProg, ...
        'showResult',  showResult, ...
        'showWaitbar', showWaitbar, ...
        'optRoute',    optRoute, ...
        'minDist',     minDist);

    varargout = {resultStruct};
end
toc
end
```

```
% Subfunction to override the default configuration with user inputs
function config = get_config(defaultConfig,userConfig)

% Initialize the configuration structure as the default
config = defaultConfig;

% Extract the field names of the default configuration structure
defaultFields = fieldnames(defaultConfig);

% Extract the field names of the user configuration structure
userFields = fieldnames(userConfig);
nUserFields = length(userFields);

% Override any default configuration fields with user values
for i = 1:nUserFields
    userField = userFields{i};
    isField = strcmpi(defaultFields,userField);
    if nnz(isField) == 1
        thisField = defaultFields{isField};
        config.(thisField) = userConfig.(userField);
    end
end

end
```

Evaluation of the distance

```
function
[totalDist,minDist,index,distHistory,globalMin,bestIter,optRoute,totalTime,tim
eHistory,route_time,Result_History,g] =
EvaluatePath(pop,popSize,dmat,tmat,omegamat,numIter,iter,totalDist,distHistory
,totalTime,timeHistory,time_servref,mission_timemaxref,Debris_to_remove,prova,
total_debris,g,Result_History)

globalMin = Inf;
totalDist = zeros(1,popSize);
totalTime = zeros(1,popSize);
```

```
for p = 1:popSize
    d = 0; % Open Path
    t = 0;
    for k = 2:Debris_to_remove
        t = t + time_servref;
        t_old = t;
        if length(unique(pop(p,:)))<Debris_to_remove
            pop(p,:) = ChromosomeHealing(pop,
p,Debris_to_remove,total_debris);
        end
        t = tmat(pop(p,k-1),pop(p,k));
        omij = abs(omegamat(pop(p,k-1),pop(p,k)));
        if omij == inf
            omij = 1e12;
        end
        if t == inf
            t = 1e12;
            t_old = 1e12;
        end
        d = d + dmat(pop(p,k-1),pop(p,k));
        if t < t_old
            while t < t_old
                t = t + omij;
            end
        end
        if t > mission_timemaxref
            d = 1000+t/107.088;
        end
    end

    totalDist(p) = d;
    totalTime(p) = t;
end

% Find the Best Route in the Population
```

```
[minDist,index] = min(totalDist);
distHistory(prova,iter) = minDist;
timeHistory(prova,iter) = totalTime(index);
if minDist<1000
    Result_History(g,:) = [minDist totalTime(index) pop(index,:)];
    g=g+1;
end
if minDist < globalMin
    globalMin = minDist;
    bestIter = iter;
    route_time = timeHistory(prova,iter);
    optRoute = pop(index,:);
end


end
```

Evaluation of the distance for the initial population

```
function [d_pop_initial] =
EvaluatePath_Initial(c,time_servref,mission_timemaxref,tmat,dmat,omegamat,Debr
is_to_remove,total_debris);
d_pop_initial = 0; % Open Path


for k = 2:Debris_to_remove
    t = 0;
    t = t + time_servref;
    t_old = t;
    if length(unique(c(:)))<Debris_to_remove
        c(:) = ChromosomeHealing(c, 1,Debris_to_remove,total_debris);
    end
    t = tmat(c(k-1),c(k));
    omij = abs(omegamat(c(k-1),c(k)));
    if omij == inf
        omij = 1e12;
    end
```

```
    if t == inf
        t = 1e12;
        t_old = 1e12;
    end
    d_pop_initial = d_pop_initial + dmat(c(k-1),c(k));

    if t < t_old
        while t < t_old
            t = t + omij;
        end
    end
    if t > mission_timemaxref
        d_pop_initial = 1000+t/107.088;
    end
end
end
```

Sorting population function

```
function [popSorted4Fitness,choice,roulette_it,totalTime] =
BubbleSorting(popSize,pop,totalDist,totalTime,Debris_to_remove)


popSorted4Fitness = pop;
temp1 = zeros(1,Debris_to_remove);
temp  = 0;
temp2 = 0;
fitness_rank = 1:1:popSize;


for indice=1:popSize
    for indice1=1:popSize-1
        if totalDist(indice1) < totalDist(indice1+1)
            temp = totalDist(indice1);
            temp1 = popSorted4Fitness(indice1,:);
            temp2 = totalTime(indice1);
            totalDist(indice1) = totalDist(indice1+1);
            popSorted4Fitness(indice1,:) = popSorted4Fitness(indice1+1,:);
```

```
            totalTime(indice1) = totalTime(indice1+1);
            totalDist(indice1+1) = temp;
            popSorted4Fitness(indice1+1,:) = temp1;
            totalTime(indice1+1) = temp2;
        else if totalDist(indice1) == totalDist(indice1+1) &&
totalTime(indice1) < totalTime(indice1+1)
                temp = totalDist(indice1);
                temp1 = popSorted4Fitness(indice1,:);
                temp2 = totalTime(indice1);
                totalDist(indice1) = totalDist(indice1+1);
                popSorted4Fitness(indice1,:) = popSorted4Fitness(indice1+1,:);
                totalTime(indice1) = totalTime(indice1+1);
                totalDist(indice1+1) = temp;
                popSorted4Fitness(indice1+1,:) = temp1;
                totalTime(indice1+1) = temp2;
            end
        end
    end
end
roulette_it = 0.3*popSize;
choice = zeros(roulette_it,1);
accumulation = cumsum(fitness_rank);
for i = 1:roulette_it
    p = rand() * accumulation(end);
    chosen_index = -1;
    for index2 = 1 : length(accumulation)
        if (accumulation(index2) > p)
            chosen_index = index2;
            break;
        end
    end
    choice(i) = chosen_index;
end
end
```

Healing of Chromosomes function

```
function healedChromosome = ChromosomeHealing(chromosome,
counter,Debris_to_remove,total_debris)


n = zeros(1,Debris_to_remove);

[C, ia, ic] = unique(chromosome(counter,:),'sorted');

for e=1:length(ia)

    n(ia(e)) = chromosome(counter,ia(e));

end

repetition = chromosome(counter,:)-n;

q = find(repetition);

for u = 1:length(q)

    chromosome(counter,q(u)) =
randsample(setdiff(1:total_debris,chromosome(counter,:)), 1);


end

healedChromosome = chromosome(counter,:);

end
```

APX Crossover function (PMX)

```
function [child1,child2] =
APX_Crossover(choice,popSorted4Fitness,roulette_it,Debris_to_remove,total_debr
is)


parent1_index = choice(1:2:roulette_it);

parent2_index = choice(2:2:roulette_it);


parent1Var = popSorted4Fitness(parent1_index,:);

parent2Var = popSorted4Fitness(parent2_index,:);


child1 = zeros(roulette_it/2,Debris_to_remove);

child2 = zeros(roulette_it/2,Debris_to_remove);


for counter=1:length(parent1Var)

    parent1 = parent1Var(counter,:);
```

```
    parent2 = parent2Var(counter,:);
    nVar = numel(parent1);
    c  = randsample(nVar,2);
    c1 = min(c);
    c2 = max(c);

    L1 = parent1(1:c1);
    L2 = parent2(1:c1);

    k1 = parent2(c1+1:c2);
    k2 = parent1(c1+1:c2);

    nVar1 = numel(k1);
    nVar2 = numel(k2);

    R1 = parent1(c2+1:end);
    R2 = parent2(c2+1:end);

    for j=1:nVar
        for i=1:nVar1
            [a,loc1] = ismember(k1(i),L1);
            if a==1
                L1(loc1) = k2(i);
                break;
            end
        end
        for i = 1:nVar1
            [b,loc2] = ismember(k1(i),R1);
            if b==1
                R1(loc2) = k2(i);
            end
        end
    end
    child1(counter,:) = [L1,k1,R1];
```

```
    for n=1:nVar
        for i=1:nVar2
            [c,loc3] = ismember(k2(i),L2);
            if c==1
                L2(loc3) = k1(i);
            end
        end
        for i=1:nVar2
            [d,loc4] = ismember(k2(i),R2);
            if d==1
                R2(loc4) = k1(i);
            end
        end
    end
    child2(counter,:) = [L2,k2,R2];
    unique(child1(counter,:));
    unique(child2(counter,:));

end

% Healing of chromosomes
if length(unique(child1(counter,:)))<Debris_to_remove
    child1(counter,:) = ChromosomeHealing(child1,
counter,Debris_to_remove,total_debris);
end

if length(unique(child2(counter,:)))<Debris_to_remove
    child2(counter,:) = ChromosomeHealing(child2,
counter,Debris_to_remove,total_debris);
end

end
```

Mutation function

```
function [child1M,child2M] =
Mutation(popSorted4Fitness,Debris_to_remove,total_debris,popSize)


mutationParents = popSorted4Fitness(0.9*popSize+1:popSize,:);

mutationParent1 = mutationParents(1:2:0.1*popSize,:);

mutationParent2 = mutationParents(2:2:0.1*popSize,:);

child1M = zeros(0.05*popSize,Debris_to_remove);

child2M = zeros(0.05*popSize,Debris_to_remove);


for counterM = 1:length(mutationParent1)


    child1M(counterM,:) = mutationParent1(counterM,:);

    child2M(counterM,:) = mutationParent2(counterM,:);

    child1M(counterM,1) = randperm(total_debris,1);

    child1M(counterM,2) = randperm(total_debris,1);

    child2M(counterM, Debris_to_remove-1) = randperm(total_debris,1);

    child2M(counterM, Debris_to_remove)   = randperm(total_debris,1);


    if length(unique(child1M(counterM,:)))<Debris_to_remove

        child1M(counterM,:) = ChromosomeHealing(child1M,
counterM,Debris_to_remove,total_debris);

    end


    if length(unique(child2M(counterM,:)))<Debris_to_remove

        child2M(counterM,:) = ChromosomeHealing(child2M,
counterM,Debris_to_remove,total_debris);

    end

end

end
```

Single Point Crossover function

```
function [child1SP,child2SP] =
SinglePointCrossover(popSorted4Fitness,Debris_to_remove,total_debris,popSize)


singlePointParents    = popSorted4Fitness(0.9*popSize+1:popSize,:);

singlePointParent1Var = singlePointParents(1:2:0.1*popSize,:);

singlePointParent2Var = singlePointParents(2:2:0.1*popSize,:);

child1SP = zeros(0.05*popSize,Debris_to_remove);

child2SP = zeros(0.05*popSize,Debris_to_remove);


for counterSP=1:length(singlePointParent1Var)

    p1 = singlePointParent1Var(counterSP,:);

    p2 = singlePointParent2Var(counterSP,:);


    nVar = numel(p1);


    c = randi([1 nVar-1]);


    child1SP(counterSP,:) = [p1(1:c) p2(c+1:end)];

    child2SP(counterSP,:) = [p2(1:c) p1(c+1:end)];


    if length(unique(child1SP(counterSP,:)))<Debris_to_remove

        child1SP(counterSP,:) = ChromosomeHealing(child1SP,
counterSP,Debris_to_remove,total_debris);

    end

    if length(unique(child2SP(counterSP,:)))<Debris_to_remove

        child2SP(counterSP,:) = ChromosomeHealing(child2SP,
counterSP,Debris_to_remove,total_debris);

    end


end

end
```

Double Point Crossover function

```
function [child1DP,child2DP] =
DoublePointCrossover(popSorted4Fitness,Debris_to_remove,total_debris,popSize)

doublePointParents    = popSorted4Fitness(0.9*popSize+1:popSize,:);

doublePointParent1Var = doublePointParents(1:2:0.1*popSize,:);

doublePointParent2Var = doublePointParents(2:2:0.1*popSize,:);

child1DP = zeros(0.05*popSize,Debris_to_remove);

child2DP = zeros(0.05*popSize,Debris_to_remove);


for counterDP=1:length(doublePointParent1Var)

    p1   = doublePointParent1Var(counterDP,:);

    p2   = doublePointParent2Var(counterDP,:);

    nVar = numel(p1);


    c  = randsample(nVar-1,2);

    c1 = min(c);

    c2 = max(c);


    child1DP(counterDP,:) = [p1(1:c1) p2(c1+1:c2) p1(c2+1:end)];

    child2DP(counterDP,:) = [p2(1:c1) p1(c1+1:c2) p2(c2+1:end)];



    if length(unique(child1DP(counterDP,:)))<Debris_to_remove

        child1DP(counterDP,:) = ChromosomeHealing(child1DP,
counterDP,Debris_to_remove,total_debris);

    end


    if length(unique(child2DP(counterDP,:)))<Debris_to_remove

        child2DP(counterDP,:) = ChromosomeHealing(child2DP,
counterDP,Debris_to_remove,total_debris);

    end


end

end
```

Inversion function

```
function invertParent =
InvertParent(popSorted4Fitness,Debris_to_remove,popSize)


invertParent = popSorted4Fitness(0.9*popSize+1:popSize,:);

tempInvert = 0;


for counterInvert=1:0.1*popSize

    numbers = randperm(Debris_to_remove,2);

    tempInvert = invertParent(counterInvert,numbers(1,1));

    invertParent(counterInvert,numbers(1,1)) =
invertParent(counterInvert,numbers(1,2));

    invertParent(counterInvert,numbers(1,2)) = tempInvert;

end

end
```