

GENETIC ALGORITHM FOR MULTI-GRAVITY ASSIST INTERPLANETARY TRAJECTORY OPTIMIZATION

Daniel Huterer Prats¹, Iker Diaz Cilleruelo², Sebastian Andreas Altmeyer^{2,*}

¹Department of Aerospace Engineering Sciences, University of Colorado Boulder, 3775 Discovery Dr., Boulder, CO 80303, US.

²Universitat Politècnica de Catalunya EETAC - School of Telecom and Aerospace Engineering, 08860 Castelldefels, Spain

+These authors contributed equally to this work

*Correspondence Author :E-mail: sebastian.andreas.altmeyer@upc.edu (<https://orcid.org/0000-0001-5964-0203>)

Abstract

Interplanetary space missions require a combination of multiple gravity-assist and impulsive Δv maneuvers which are difficult to design with traditional mission analysis techniques. As these mission often demand the optimization of numerous variables in a highly nonlinear and discontinuous design space, a prepruning is required to determine potential trajectories. However, in doing so, one may also cut out non-intuitive solutions, which may contain the optimal trajectory. In this paper, we present the development of a genetic algorithm that is capable of determining optimal interplanetary MGA trajectories but not the sequence itself, where tradeoffs between minimizing Δv and time, in particular the time taken from the first possible launch date to the destination, can be made. We illustrate this as an improvement to just finding the minimum Δv trajectory that does not take time into account by analyzing the performance of the algorithm and providing comparisons with experimental data from real missions as well as other numerical codes. Moreover, it is also different from algorithms that aim to optimize time of flight, as in this paper the arrival date can be targeted, which provides a more industry oriented optimization approach.

Keywords: Multi-planetary gravity assist, Genetic algorithm, Global optimization, Interplanetary trajectory optimization, Deep Space Maneuver.

1. INTRODUCTION

Multi Gravity Assist (MGA) trajectories are commonly used by satellites in order to reduce the amount of Δv required for a mission. These MGAs or 'flybys' involve doing a close approach to a planet or body to change the momentum of the spacecraft without the use of its own propulsion system. Also, this can allow devoting more mass for other spacecraft components or subsystems. This technique can have virtually zero cost if the trajectory is properly designed and allow for missions that otherwise would have an unachievable Δv requirement. In fact, MGA trajectories enabled the Voyager I and II missions to reach their destinations. Without the several flybys they performed during the trajectory, the launcher used (Titan III-E/Centaur D-IT) [1, 2] would not have been capable of providing the required Δv .

The number of flybys, their sequence, and flyby date play a crucial role in the overall performance. Designing the most optimal trajectory for a mission that involves multiple gravity assist maneuvers is complex and also requires taking other factors into account, such as the time of flight (the time taken from launch to the final destination). This is an essential part of the overall time of the mission as it discounts from the remaining time at the target to fulfill the mission objectives. The cost associated with the satellite reaching its destination is not only the fuel but also all the ground stuff and many researchers associated with the mission (depending on the type of mission 100s of persons) that have to be paid even when the spacecraft is still on its way to the target. In this paper, not the time of flight but in fact the time taken from the first possible launch date is considered. This distinction allows the user to also take into account the different satellite manufacturing, testing and qualification phases as well as the launcher availability logistics. All these

phases compose the majority of the cost for most missions and delays in the launch, requiring the appropriate storage of the satellite, can cause significant financial pressure on the operators. Therefore, the primary goal of the mission cannot be reduced to simply finding the trajectory with the lowest required Δv . In fact, a balance between the Δv and the arrival time has to be found that is dependent on the profile of the mission, meaning the mission designers are able to prioritize between the two as needed from an economic and scientific perspective. The implementation of time in the model presented here further allows the targeting of a specific arrival date. This is required by some missions to leverage the local geometry at the destination occurring only at a certain date, which cannot be done as easily when 'time of flight' is used instead since the launch date is still variable (as compared to a fixed moment in time such as the first day of the possible launch window). For instance, the Cassini-Huygens mission [3, 4] a four-year orbital survey of Saturn, its rings, magnetosphere, and satellites, and the descent into Titan's atmosphere of the Huygens probe. Here both satellites had to arrive in tandem to Jupiter at a specific time when one of the moons was located in a suitable spot for one of the satellites to descend to its surface without comprising the other's mission significantly. Moreover, the importance of an optimal and specified arrival time has also been quoted by Matson et al. [5] saying, 'the SOI [Saturn Orbit Insertion] burn will occur earlier in time (than optimum for minimizing propellant usage) so that the burn will end near the closest approach to Saturn. This allows time for post-periapsis remote sensing of the rings.' A further manifestation of the importance of a specified arrival time is based on the need to pass through the rings of Saturn in a region known to be free of particles to prevent Δv -expensive maneuvers required to fly around hazardous regions. Without going into too much detail, another examples is the ROSETTA mission design [6] which required an arrival at a certain time in the orbit of the comet to ensure the right approach angle while also guaranteeing stable communication with the Earth (visibility).

Furthermore, a minimum Δv trajectory often entails a later arrival date as compared to a higher Δv trajectory, the amount of time at the final destination can be massively increased which provides significantly more amounts of useful data and a prolonged satellite lifespan at the destination. Hence, more data can be collected using the same spacecraft, further making more use of the scientific resources of the spacecraft itself as well as the researchers analyzing them on Earth. Over the course of this paper, it will be shown that the tradeoff at different prioritization levels of the two mission design parameter (Δv and the arrival time) creates a family of trajectories depending on the designer's needs. This optimization problem is NP-Hard, e.g. a computational problem that is at least as hard as the hardest problems in NP (Nondeterministic Polynomial time). These algorithms are those that can be verified by a polynomial-time algorithm, but the solution may not be found in polynomial time by a deterministic algorithm.

Aside the various available deterministic optimizers (optimization algorithms) [7], another more recent and today more common approach is to use an evolutionary algorithm [8–10]. Deterministic algorithms assure the convergence to the optimal value (global minima) as long as the problem is well-defined, however, they are complex. Evolutionary algorithms, e.g. genetic algorithms (GAs), offer an alternative that is much simpler yet can still provide optimal results [10]. This class of algorithms have been already proven to be very successful in improving interplanetary trajectories. The list is already very long and contineously increasing, to mention all of them here. Xu et al. [11] utilized a GA to optimize launch and deep-space maneuver parameters for an approach to two opportunistic asteroids on the way to their main target. Molenaar [12] used an improved GA tool (GALOMUSIT) that it allows for multimodal optimization in order to determine an optimized trajectory for an Uranus orbiter. Another study to mention is the work by Izzo et al. [13] that discuss a deterministic search space pruning algorithm for multiple gravity assist problem. Further preliminary design of multiple gravity assist maneuvers as a global optimization method based on a hybrid genetic algorithm methodology has been developed und studied by Vasile and Pascale [14]. A detailed performance study of several search algorithms is presented in the work by Vasile et al. [15] with the outcome of a new algorithm based on redefinition of some heuristics in form of a discrete dynamical system. Gad and Abdelkhalik [16] completely focused on implementation of genetic algorithms on free number of deep-space maneuvers. Therefore they rendered the problem as a variable-size design space optimization problem which they called a multi-objective hidden genes genetic algorithm (MOHGGA). A modulation as discontinuities in velocity arising from an instantaneous turning of the spacecraft is

discussed by Petropoulos and Longuski [17]. This study deals with multi-objective optimization of impulsive trajectories where the objectives are the total fuel consumption and the total flight time. With the number of flybys, flyby sequence, locations and values of the deep space maneuvers all to be unknowns, the problem is considered a variable-size design space optimization problem [18].

In this paper, we present a new developed simple genetic algorithm (GA), to showcase its effectiveness to optimize a multi-planetary gravity assist (MGA) interplanetary trajectory in terms of the arrival time t and total Δv of the trajectory. So far we do not address the optimization of flyby sequence, which will be done in a future work. While most of the literature only use single objective optimization functions, this combination with two objectives prioritizing Δv and time t , respectively, allows the end user to find a family of solutions beyond just the minimum Δv trajectory. This enables the option to target a specific arrival date and with therefore helps to minimize costs, e.g. leverages a lot more of the human resources that are being paid on the ground working on the mission.

NOMENCLATURE

Δv	delta-v
t	time
$\vec{v}$	velocity
$\vec{v}_{inf}$	relative velocity
$\vec{X}$	(time-dependent) trajectory vector
$c(\vec{X})$	fitness function
$g(\vec{X})$	penalty function
$f(\vec{X})$	injections
in, out	incoming, outcoming
a	semi-major axis
e	eccentricity
I	inclination
L	mean longitude
T_{eph}	ephemeris time
P_i	point/position of planet i
ω	longitude of perihelion
Ω	longitude of ascending node
μ	gravitational parameter
α	prioritization weight for delta-v
β	prioritization weight for time
δ	turning angle

2. PROBLEM FORMULATION

There are two variants for the type of trajectory concerning us; the MGA and the MGA-DSM (Deep Space Maneuver). MGA is the simplest variant, on it, a single Δv impulse is allowed during the periapsis passage at each flyby, while MGA-DSM is an extension in where other Δv impulses are allowed during the different legs of the trajectory. Both variants have real use cases and have been used by different missions. While MGA-DSM algorithms integrate deep space maneuvers into the trajectory design which offers enhanced flexibility and optimization potential as described by Englander et al.[10] and Schlueter et al.[19], this study utilizes the MGA model due to its simplicity and direct application to the mission scenario considered in this paper.

In this paper, we present a new algorithm to optimize MGA trajectories. While most commonly only the total Δv (the sum of the departure Δv and the impulses at each flyby) is considered as the minimization variable, here we consider a two-variable combination of Δv and arrival time t , that is the time taken to reach the final destination from the first day of the departure/launch window (which is driven by the satellite owner as well as the launch provider and environmental factors that are not taken into account in similar studies that optimize TOF in space. This objective can be converted to a multiple-objective function for the optimization problem.

$$\underset{\vec{X}}{\text{minimize}}(\alpha \sum_{i \in \vec{X}} \Delta v_i \beta \sum_{i \in \vec{X}} t_i), \quad (2.1)$$

where Δv_i is the Δv at each time event t_i and α and β are the prioritization weights for the Δv and time respectively (described in Eq. 2.5). The input to the problem, e.g. the optimization variable is a time-dependent vector $\vec{X}$.

$$\vec{X} = [t_0, t_1, \dots, t_n]. \quad (2.2)$$

Each element of $\vec{X}$ is a date representing an event in the trajectory. t_0 represents the departure date, t_i for $i \in [1, n-1]$ the dates in which a flyby is conducted, and t_n the arrival date at the destination point, most commonly a planet. From this, we can see that the sequence of planets is determined, e.g. pre-defined by the user. The sequence of planets could be as well optimized in the first step as in [10,20,21]. The so-obtained optimal flyby sequence could then be used for MGA optimization as presented in this paper. The two parts, e.g. the optimization of the flyby sequence as well as the optimization of the trajectory, are known as the outer and inner loop problem, where the first (outer loop) serves as the ground work for the second (inner loop). This approach is ideal when the trajectory is not relevant, but only the final destination is. A double optimization as such would avoid pruning out the optimal trajectory from the whole possible set of planet sequences. In the scope of this paper, we will focus solely on the trajectory optimization, that is the inner loop problem.

The vector $\vec{X}$ defines the interplanetary trajectory with the various time events t_i . As so, the Δv cost of it is also set as well as the arrival date (from t_n). The current model is discretized in time to define the trajectory. At each time event, the necessary Δv to match the continuous trajectory is computed. It must be noted that the optimal solution is inherently bounded by the corresponding considered model. For different models, the optimal solutions will usually be different. Therefore, when comparing the solution obtained to the real missions and other algorithms solutions, they should not be taken with scruples as the models used usually differ. However, this fact is not significant, as the aim is to demonstrate the use of a simple genetic algorithm to optimize an MGA trajectory. The algorithm will find the optimal trajectory for the model used.

The trajectory is computed and optimized by a genetic algorithm. Evolutionary algorithms [8] simulate nature's evolution principle to optimize a problem. In specific, genetic algorithms mimic the principle of natural selection and the evolution of populations. This kind of algorithm is ideal for optimizing problems where a solution can be easily obtained without a complex description of the problem. In other words, they do not require a strict mathematical description of the optimization problem like gradients, singularities, etc., as other deterministic approaches do.

Provided an input x , which in the current work is the vector of flight times $\vec{X}$, and an output $f(\vec{X})$ and $t(\vec{X})$ (representing the Δv and time of the trajectory), the algorithm is able to assess the relative optimality of that input for a set of possible inputs. By comparing the values, it tends them to the optimal one. The convergence of the algorithm is guaranteed by the use of genetic operators which generate new inputs weighted to an optimal value. These kinds of algorithms are simplistic at their core as they are similar to a random search, however, done in a controlled way.

In general, genetic algorithms use a set of inputs to the problems which is the so-called population. Each input is an individual (in the present case a trajectory) in the population. Each individual has a certain fitness value, e.g. how good the output is given an input. This fitness is assessed based on a fitness function $c(\vec{X})$:

$$c(\vec{X}) = \alpha (f(\vec{X}) + g(\vec{X})) + \beta (t(\vec{X})), \quad (2.3)$$

where $g(\vec{X})$ represents a penalty function to add soft constraints, $f(\vec{X})$ are the Δv injections during the flybys in addition

to the departure Δv and $t_n - t_0$ is the time taken to the final destination. The parameters α and β are the weights that the end user can tune depending on how they want to prioritize Δv vs arrival time. The function $f(\vec{X})$ is the sum of the departure and flyby Δv 's.

$$f(\vec{X}) = \Delta_{departure}(t_0) + \Delta_{fb}(t_1) + \dots + \Delta_{fb}(t_n). \quad (2.4)$$

For the main component of the penalty function, a piece-wise function is used where if $r_p < 1.1r_{pl}$, the penalty cost $g(\vec{X})$ is a large constant value:

$$g(\vec{X}) = \begin{cases} 0 & , r_p \geq 1.1r_{pl} \\ 1e10 & , r_p < 1.1r_{pl} \end{cases}$$

The function $t(\vec{X})$ is the time taken from the first date of the launch window, and then renormalized to comparative Δv value that can be used to relate it to the other variables in Eq. (2.3). This normalization entails running the algorithm first with $\alpha = 1$ (and $\beta = 1$), meaning only minimizing Δv , to get an initial arrival time. The obtained arrival time is then used to scale all other simulation that have α and $\beta \neq 0$, in conjunction with the minimum Δv taken. While typically a higher value is a better fitness, since this can be considered a cost function, the lower the value of $c(\vec{X})$, the fitter an individual is.

The objective function for the current investigation is provided by Eq. (2.1). The quality of the solution derived from that input can be ranked based on the associated fitness or cost of an individual. The algorithm effectively evolves the entire population in order to optimize the problem. It starts with assessing each person's level of fitness. Using a selection procedure, people are chosen from the population according to how fit they are. Using a series of genetic operators, the selected individuals - who are typically the fitter ones - are "crossed" to produce new and perhaps improved inputs made up of the advantageous genetic material of the parent generation. In order to maintain diversity, find new solutions, and keep the population from becoming centered around a local minima, mutation is also finally introduced into the population.

Following this process, a new population generation is produced, with new inputs thought to be superior because they are descended from fit individuals. Until a predetermined number of generations, N_{GEN} , is reached, this process is repeated. The mechanism is the same as what occurs in nature: the most successful individuals are more likely to pass on their genes to the following generation. Because of natural selection and the survival of the fittest, each generation will typically outperform the one before it. The operators of the genetic algorithm are used to numerically model this process.

Formally, in the MGA problem, each individual in a population of size N_{POP} is a vector of dates $\vec{X}$ (Eq. (2.2)) which is the input describing the initial trajectory. Subsequently, this trajectory is evaluated for each individual, and a cost/fitness is derived. Hereafter, the population is ranked, and individuals are selected for reproduction, which are then used to create new individuals for the next generation. After the process is repeated until the limit of generations is reached, the remaining fittest individual will be the optimal trajectory, e.g. the one with the smallest cost in terms of Δv and time calculated using 2.3. The present code allows to pre-define how the balance between those two terms is constructed, using the prioritization weights α and β for the Δv and time, respectively. This relationship between the weights is defined as:

$$1 = \alpha + \beta \quad (2.5)$$

All these are obtained under the assumption that the parameters of the genetic algorithm are properly selected. In subsection 4.3 these parameters are analyzed in more detail below.

3. ALGORITHM

The algorithm can be divided into two main parts; (A) the computation of the trajectory and (B) the genetic evolution. The computation of the trajectory is independent of the GA. In (A), given the provided sequence of planets and a vector of dates, the associated trajectory with the respective Δv cost, and some other orbital parameters from the multiple

flybys are computed. These results are then fed into the GA which uses them to evaluate the performance of multiple individuals, and assess which trajectory is better.

The complete algorithm has been developed in C++. This language has been chosen because it is compiled, hence, it is more optimal and fast during run-time. In addition, while being low-level, it contains a large built-in standard library. The language also allows the use of multi-threading to increase the speed of independent concurrent operations, contrary to other popular languages, such as Python, that do not support this option (due to Python's GIL). Using multiple threads can greatly reduce the computation time of the GA, most notably for complex trajectories.

Figure 1 illustrates a schematics of the overall code diagram which will be presented and discussed in detail in the following.

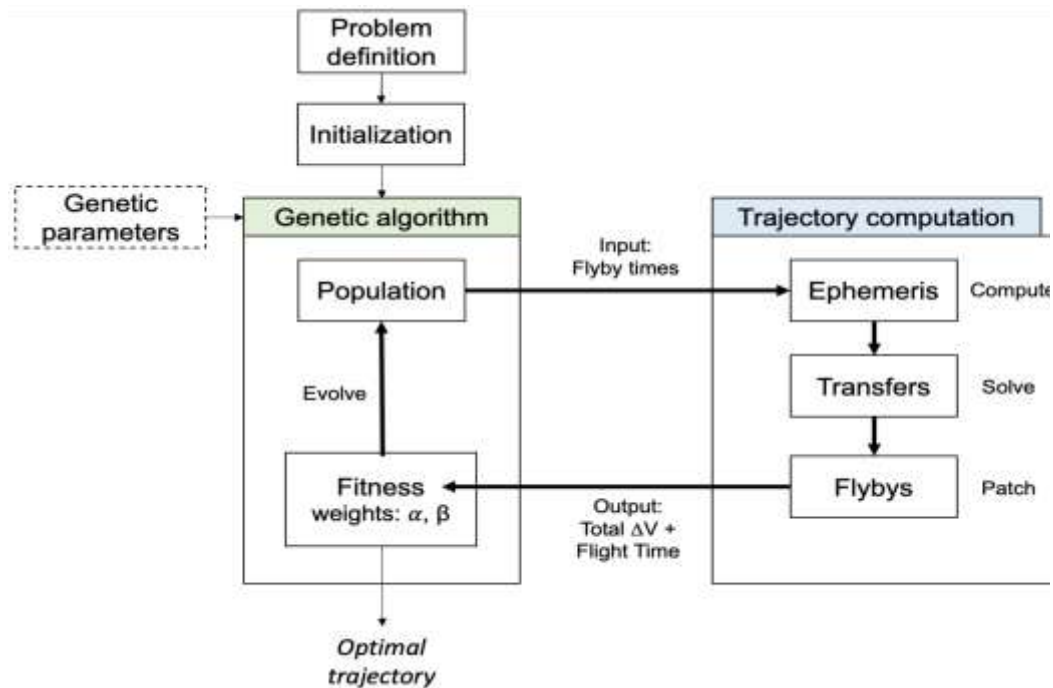


Figure 1: Flowchart - Tentative Code diagram

3.1 Trajectory computation

Computing an interplanetary trajectory can be very challenging, due to the interaction of numerous bodies that influence the spacecraft's trajectory. In order to simplify the problem, the interactions are simplified too. The trajectory is computed on a 2-body problem basis, rather than the N-body. Although this simplification is not ideal, it is accurate enough to estimate the optimal trajectory. All over the trajectory is broken down into three parts. (1) The ephemeris computation, (2) the Lambert problem, and (3) the patched conic approximation, which will be detailed described in the following.

3.1.1 Ephemeris

We recall that the trajectory is defined by a sequence of planets and a time vector $\vec{X}$. that moment in time at which a planet interacts during the trajectory. Hence, the first step in the computation is to determine the position and velocity

of the planet at the corresponding time. One option to do this, is to use the planet's ephemeris. The ephemeris is a set of numerical data that describes the positions and movements of celestial bodies over time [22]. Specifically, it provides the positions of planets in the solar system relative to a reference point. The data is typically generated using complex mathematical models that take into account the gravitational influences of other celestial bodies, as well as other factors that affect the movement of planets [23].

Following the ephemeris is used to obtain a position and velocity vector $\vec{r}$ and $\vec{v}$ of each planet. The computation is done following the algorithm present at [22] (web format by NASA at [23, 24]). With all provided data for a planet e.g. the Keplerian elements and their rates $(a, e, I, L, \omega, \Omega)$, the orbital parameters are determined at a time T_{eph} . The ephemeris puts down the first piece of the trajectory computation puzzle, which is to locate the planets at their corresponding moments in time.

3.1.1 Lambert problem (transfers)

With the former described computation of the planetary ephemeris, one obtains a static picture of the location of the planets and their velocities at their corresponding time. Next step is to compute the transfers between each planet. To accomplish this step one has to solve the Lambert problem [25]. Lambert's problem determine the trajectory between these two points P_1 and P_2 after the time of flight T between two points P_1 and P_2 is determined. The trajectory is obtained after finding the departure velocity from point P_1 because the position and velocity of any point in the orbit are determined by r_1 and v_1 [26]. The Lambert solver implemented in the algorithm for this work was developed by Dario Izzo and Gooding [27]. The solver is developed in C++ and forms part of the open-source ESA's Pykep[36] library, a scientific library providing basic tools for astrodynamics research.

3.1.1 Patched conic approximation

To this point, we have completed two out of the three steps necessary to fully define the interplanetary trajectory. We know the parameters as position, velocities, and trajectory between every pair of planets, but so far they are not connected yet. For example, the Earth-Mars-Jupiter (E-M-J) trajectory has two separate legs (E-M and M-J). To complete the trajectory, one must match the arrival velocities from the transfer orbits to the departure velocities provided by the Lambert solver. For E-M-J, one must match the arrival velocity at Mars from the Earth transfer with the required departure velocity from Mars to transfer to Jupiter.

To perform this operation, we will consider the patched conic approximation. Under the sphere of influence (SOI) ([26] Sect 8.4) of a planet, the spacecraft follows a hyperbolic planetocentric trajectory [28]. Otherwise, it follows a hyperbolic heliocentric trajectory. A Δv cost is associated to match the incoming and required outgoing velocity around a planet. This cost represents the Δv impulse associated with the flyby. Using the patched conic approximations allows us to only focus on the gravitational interaction between the planet and the spacecraft, as this last will be under its SOI. To match the incoming and outgoing heliocentric velocities, one needs the required turning angle and periapsis radius [29]. The difference in velocity will be the Δv impulse required. The velocities relative to the planet are obtain as follows

$$\vec{v}_{\infty-in} = \vec{V}_{in} - \vec{V}_p, \quad \vec{v}_{\infty-out} = \vec{V}_{out} - \vec{V}_p \quad (3.1)$$

where $\vec{v}_{\infty}$ are the relative velocities, $\vec{V}$ the heliocentric velocities (from Lambert solution) and $\vec{V}_p$ the planet's velocity. To determine the periapsis radius r_p we must determine the semi-major axis of both trajectories (at pre- and post-periapsis passage).

$$a_{in} = -\frac{\mu_p}{v_{\infty-in}^2}, \quad a_{out} = -\frac{\mu_p}{v_{\infty-out}^2}. \quad (3.2)$$

The turning angle δ between the asymptote of both trajectories at the bounds of the SOI [26] is computed as

$$\delta = \left(\frac{\vec{v}_{\infty-in} \vec{v}_{\infty-out}}{v_{\infty-in} v_{\infty-out}} \right). \quad (3.3)$$

The periapsis can be expressed as

$$r_p = a_{in}(1 - e_{in}) = a_{out}(1 - e_{out}), \quad (3.4)$$

where e_{in} and e_{out} are the eccentricities of the incoming and outgoing hyperbolic trajectories. The turning angle can be rewritten in terms of the incoming and outgoing eccentricities as

$$\delta = \sin^{-1}\left(\frac{1}{e_{in}}\right) + \sin^{-1}\left(\frac{1}{e_{out}}\right). \quad (3.5)$$

Then, Eq. (3.4) can be used to obtain a function with single dependence on e_{out} . Afterward, Eq. (3.4) could be solved for e_{out} using an iterative method like Newton-Raphson [29]. However, this approach is unnecessary, if we assume that the trajectory will be correctly joined with the necessary Δv maneuver at the periapsis. Under this assumption we can model the full trajectory as an flyby from the periapsis passage. In that case, we can express the turning angle as follows $\delta = 2\sin^{-1}(1/e)$ (example 8.6 in [26]) from which we isolate the eccentricity as

$$e = \sin^{-1}\left(\frac{\delta}{2}\right). \quad (3.6)$$

Using Eqs. (3.2) and (3.6) we can solve (3.4). Finally, the required Δv applied at periapsis is

$$\Delta v = \left| \sqrt{v_{\infty-in}^2 - \frac{2\mu_p}{r_p}} - \sqrt{v_{\infty-out}^2 - \frac{2\mu_p}{r_p}} \right|. \quad (3.7)$$

Using these calculations, we can patch the two trajectories by applying the Δv impulse. We determine the periapsis radius and turning angle of the trajectory. However, it is important to note that both values are unconstrained, and it is possible to end up with an impossible periapsis radius or turning angle when patching the two trajectories. For example, the periapsis radius may pass through the interior of the planet in some cases. To avoid these non-physical cases, a penalty function must be incorporated into the algorithm

Using the time vector $\vec{X}$ defining the trajectory dates, we can compute the full trajectory and obtain the total Δv required, which is the sum of the departure Δv and every flyby Δv impulse (Eq. (3.7)).

3.1 Genetic algorithm

A genetic algorithm (GA) is a powerful stochastic optimization technique that models natural selection and reproduction to evolve optimized designs. In contrast to calculus-based methods, GAs operate using a population of designs and key genetic operators: (1) selection, (2) crossover, and (3) mutation. These emulate biological evolution, creating improved populations across generations.

An initial population of individuals is generated randomly, and their fitness values are assessed based on a multivariable cost function. Fitter individuals are selected as parents, similar to natural selection. Crossover, applied here in the form of uniform crossover (or other possible crossover types), combines two parents to create offspring with blended attributes. These offspring inherit genetic characteristics while introducing new patterns to explore the design space. Mutations are introduced to some offspring to maintain diversity in the population and to prevent convergence to local minima. This iterative process continues until convergence or the generation limit is reached. Since the original input of the optimization problem is a set of flyby times that compose the $\vec{X}$, coded in (shortened) Julian dates, it needs to be transcribed into a binary string that represents a 'chromosome', e.g. the entire information of the individual trajectory. In this application, the search space for the optimal arrival is composed of several large time intervals which shall not be shrunk prematurely in order to avoid pruning out the globally optimal solution in advance. Here we chose a genetic algorithm (GA) as these types are well suited for combinatorial problems, and are a straightforward method that is relatively easy to apply [30]. Moreover, a key advantage of GAs is that there are no initial guesses required as the initial population is randomly defined [31]. Additionally, without the need for gradient information, the genetic operators can effectively explore discrete, multimodal, and expansive design spaces. Furthermore, modifications to the simple GA can transform the algorithm into an effective multi-objective optimization scheme.

3.2.1 Selection operator

While there are multiple different selection algorithms that can be implemented, in this work the 'tournament selection'

was used. In this method, multiple individuals are selected and the fittest of those is eventually used for crossover. This introduces an intrinsic tendency to promote beneficial genes, meaning the flyby dates that produce more optimal solutions are more likely to be used for the subsequent evolutionary steps.

3.2.2 Crossover operator

Similarly, the crossover can be applied in various ways such as for example single or double-point crossover, single-gene or uniform crossover, etc. Here, different sections of the two parents are exchanged based on the method selected. Note, that here a gene implies a portion of the whole chromosome that is being switched between the two progenitors.

3.2.3 Mutation operator

The final step of the traditional evolutionary process observed in nature is mutation, which in the context of this work is represented by bit flips in the binary string of each new individual produced by the crossover. The effect of different mutation methods trialed has shown to be minimal, hence 'flip-bit' mutation was used throughout this work, where a randomly chosen bit in the string is flipped. Depending on the location of this mutation (and the nature of binary numbers), it can have a minor or a major effect on the flyby date it altered. This allows the code to be resilient to converging at local minima and ensures the creation of new individuals in the population.

4 CODE VALIDATION AND RESULTS

In order to assess the performance of our approach, the new algorithm has been tested with both, real missions (Voyager I [1] and Galileo [32], both targeted at reaching Jupiter) and other well-established algorithms, the Pykep and Pagmo libraries. Pykep[37,33] is an ESA library for astrodynamics research, while Pagmo [38,34] (an ESA library too) is used for massively parallel optimization. Pagmo optimized the MGA problem defined with Pykep.

Some factors have to be taken into account for comparison and understanding the results. For the real mission data, no Δv data are available to obtain the real mission Δv cost. The data used to represent the real mission are gathered by the dates of the different events (departure / flyby / arrival) and the periapsis at the flybys. The total amount of Δv has been computed in our algorithm basis, meaning, using the dates from the real mission or Pykep+Pagmo solution in our algorithm. The motivation is to have a solution based on the same model, as our model has some deviation from the one used in Pykep+Pagmo, and the one used to define the real mission. Even the precision (decimals used) may have a large impact on the solution as the values used are extremely large or small (astronomical distances). For instance, dates on our algorithm are accurate to 8 levels, while in Pykep+Pagmo uses floating precision (64-bit). The

8 levels of precision result from the 3 bits that were devoted to daily decimal precision. While 3 bits were chosen for all the runs of the algorithm, this parameter can be increased for a higher precision, at the downside of a longer execution time. This implies that the time resolution used is 3 hours for each event in the trajectory, e.g. departure, flybys and arrival dates. This alone can already result in large variations in the optimal solution.

While this fact is not ideal, it does not undermine the main goal of this paper, which is to demonstrate that MGA problems can be optimized with a simple GA. If the same model was to be used during the optimization, the results would match more accurately the results provided by Pykep+Pagmo, or other algorithms.

4.1 Voyager I

4.1.1 Set-up:

The code validation initiates by focusing on the Voyager mission, renowned for its simplicity in trajectory. Launched from Earth in 1977, this mission successfully executed a single flyby at Jupiter before charting a course toward Saturn. The algorithm's input parameters used in the GA are shown in Tab. 1. The population size was fixed at a large number of 15000, which would guarantee that the code would optimally converge to the global minimum. Although these values are purposefully low to ensure precision, it's important to remember that they might be lowered to improve computing

effectiveness. The ten most fit members of the population were labeled as "elite", while the other members were selected and multiplied using tournament and double point systems, respectively. The crossover (as opposed to reproduction) probability between two individuals was set at 0.9. Furthermore, a mutation process utilizing the flip-bit method is shown, in which every person has a 0.2 likelihood of undergoing a mutation.

The windows for the event dates were set quite large in order to have a sufficiently big search space, and thus increasing the chances of obtaining a more optimal solution. The departure window for the departure time (T_0) spans from January 1, 1977, and concludes three years later. Similarly, Jupiter and Saturn's flyby times (T_1 and T_2) are within an extensive window of 5.33 years each. It is imperative to align these window sizes/durations with the realistic speed capabilities for interplanetary travel. In particular, for large interplanetary distances, the start of the window must be adjusted accordingly; otherwise, the required velocity to reach the designated planets within the allotted time would become prohibitively high, and potentially unrealistic. Consequently, it is plausible to encounter scenarios where the values of T_1 and T_2 fall within the prescribed window, e.g. of 50 days which is possible from the inputs used. However, such trajectories would necessitate an exorbitant amount of Δv to be exerted within a comparatively short time frame, resulting in trajectories with significantly diminished fitness and correspondingly high costs.

Table 1 Genetic Algorithm Parameters Voyager-I

GA parameters		
Population		$N_{POP} = 15000$
Generations		$N_{GEN} = 50$
Elitism	10	
Selection	Tournament	
Crossover	Double Point ($P = 0.9$)	
Mutation	Flip Bit ($P = 0.2$)	

4.1.2 Results:

The precise outcomes yielded by the genetic algorithm are presented in Tab. 2 while a comparative analysis between the optimal trajectory and the real mission profile is depicted in Fig. 2. Notably, the optimal trajectory has a departure time which is nearly coincident with the real mission, differing by a single day. Subsequently, the optimal trajectory adopts a more gradual and optimized approach. The flyby at Jupiter transpires two years later, occurring within the same year as the actual mission but with a delay of seven months (Tab. 2). Meanwhile, this flyby is executed at a higher periapsis radius, characterized by a less abrupt turning angle, and thereby necessitates a Δv impulse of only 0.34m/s. The arrival date of the optimal trajectory is April 1982, differing by 1.5 years from the one of the real mission in November 1980. The total Δv requirement for the optimal solution amounts to 9.41km/s, which is smaller as the real mission necessitates 10.33km/s. Note to mention that this analysis is done using our model for the trajectory and therefore the values may vary compared to the mission's real one.

Table 2 Trajectory optimization for Voyager-I; inputs (left) and solution (right)

GA parameters		Trajectory Result	
Population	$N_{POP} = 15000$	Real	GA code
Generations	$N_{GEN} = 50$	Earth Departure	
Elitism	10	05-09-1977	04-09-1977
Selection	Tournament	Date T_0	2443391
Crossover	Double Point ($P = 0.9$)	Departure Δv [m/s]	9413.34

Mutation Flip Bit ($P = 0.2$)**Trajectory parameters**

$T_{0,min}$	01-01-1977
	2443145
$T_0(Earth)$	[0, 1095]
$T_1(Jupiter)$	[50, 2000]
$T_2(Saturn)$	[50, 2000]

Jupiter Flyby

	05-03-1979	17-10-1979
Date T_1	2443937.5	2444163.5
$v_{\infty-in}[km/s]$	10964.4	6558.52
$v_{\infty-out}[km/s]$	10960.7	6559.38
$\delta[deg]$	98.4863	93.0884
$r_p[m]$	3.376×10^8	1.1118×10^9
$\Delta v[m/s]$	1.1371	0.34356

Saturn Arrival

	12-11-1980	24-04-1982
Date T_2	2444555.5	2445084.125
Arrival $V_{in}[m/s]$	20115.7	12704.8

Result

Total cost Δv	10331.5	9414.05
---	----------------	----------------

The disparity between the two solutions primarily stems from the departure Δv . While both trajectories entail minimal Δv during the flyby, the real mission demands a higher departure speed to ensure timely arrival at Jupiter, owing to an earlier flyby date. Consequently, this incurs an additional cost of 0.91km/s at the departure phase of the actual mission. Nonetheless, the performed flyby endows the real mission with a much higher speed, resulting in an arrival velocity of 20.11km/s at Saturn, compared to the optimal trajectory's 12.7km/s. This discrepancy can be attributed to the lower periapsis radius attained by the real mission, imparting a greater velocity.

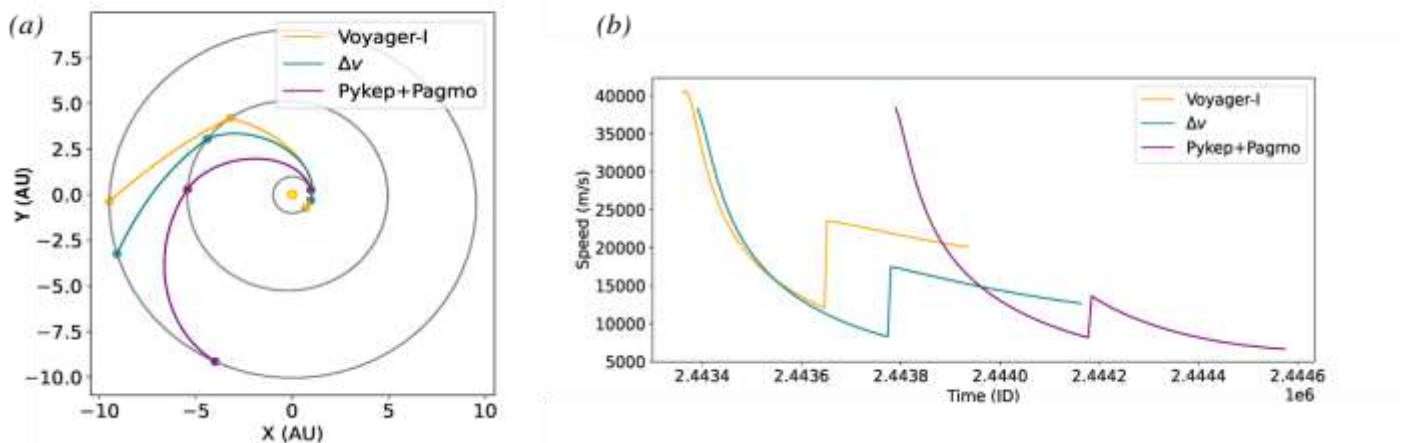


Figure 2 (a) Trajectory plot and (b) Velocity evolution for the Voyager-I mission in comparison with the real mission data and *Pykep+Pagmo* optimizer.

In the velocity plots of Fig. 2 it is observed that the real mission does a more energetic fly-by around Jupiter as the velocity increase is larger, providing it with a higher speed, which leads to an earlier arrival time. Table 2 shows that the fly-by is done at a lower periapsis in comparison with our solution, which provides the spacecraft with more speed, at the cost of a larger Δv injection cost during the maneuver. While our solution fly-by enters at a similar angle, the higher periapsis provides it with a lower speed gain, however, the required Δv is lower. Nonetheless, the main additional Δv cost incurred by the real Voyager 1 mission comes from the high departure velocity.

4.2 Voyager II

4.2.1 Set-up:

Similar to Voyager I, the Voyager II satellite completed the same initial trajectory as its twin satellite up until Jupiter. At this point, Voyager II went on to complete another set of gravity assist maneuvers while passing around every remaining planet of the outer solar system. In total, this mission included four gravity assist over 12 years starting in 1977 and reaching its final destination in 1992. (Such a mission was only achievable due to the favorable position of all the outer planets during that era. Today, this mission would be impossible in a similar amount of time as now the outer planets are misaligned.) Due to the increased complexity of the trajectory compared to the Voyager I mission, the respective search space is much larger and there is a much greater number of potential solutions. To account for this added intricacy the initial population size was increased from the aforementioned 15000 individuals to 50000 individuals. Moreover, while some GA parameters such as the number of elites, crossover and mutation probability have been left unchanged, the generational limit was also raised to 500 to allow more time to find the optimal solution. Table 3 provides the corresponding input parameters used in the GA for Voyager-II. Finally, since the aim of this paper is to demonstrate the utility of optimizing an interplanetary trajectory beyond just the minimization of the Δv spent, the prioritization parameters α and β were chosen at the level of 0.7 and 0.3 respectively. These values were chosen in order to obtain an arrival time similar to the flown mission. Higher levels of β where time is further prioritized have also been explored. Nonetheless, to benchmark the effectiveness of our model in purely minimizing the Δv required of the mission, the $\alpha = 1.0$ trajectory computed using our trajectory design algorithm is also added to the comparison. The results of the real mission, along with the other solvers and the two trajectories constructed using our code are shown in Tab. 4.

Table 3 Genetic Algorithm Parameters Voyager-II

GA parameters	
Population	$N_{POP} = 50000$
Generations	$N_{GEN} = 500$
Elitism	10
Selection	Tournament
Crossover	Double Point ($P = 0.9$)
Mutation	Flip Bit ($P = 0.2$)

Table 4 Voyager-II results comparison using parameter of Tab. 3

Trajectory Results				
Real mission minimum Δv and Time Tradeoff Pykep+Pagmo				
	$(\alpha = 1)$		$(\alpha = 0.7)$	
Earth Departure				
	20 – 08 – 1977	04 – 09 – 1977	01 – 09 – 1977	05 – 11 – 1979
Date T_0	2443375.5	2443391	2443388	2444182.5
Departure Δv [m/s]	10230.7	9413.34	9855.66	9994.88
Jupiter Flyby				
	09 – 07 – 1979	17 – 10 – 1979	03 – 05 – 1979	07 – 07 – 1981
Date T_1	2444063.5	2444163.5	2444000	2444792.5
$v_{\infty-in}$ [km/s]	7901.55	6563.52	9298.66	10088.1
$v_{\infty-out}$ [km/s]	7757.95	6559.38	9295.96	10087.6

$\delta[deg]$	96.9863	93.0884	97.7442	15.17
$r_p[m]$	7.058×10^8	1.1118×10^9	4.8026×10^8	8.18×10^9
$\Delta v[m/s]$	57.844	0.34356	1.01265	0.40
Saturn Flyby				
Date T_2	26 – 08 – 1981	24 – 04 – 1982	18 – 03 – 1981	27 – 11 – 1984
$v_{\infty-in}[km/s]$	2444842.5	2445084.125	2444680	2446031.5
$v_{\infty-out}[km/s]$	10790.6	8254.31	13088.7	5082.83
$\delta[deg]$	9052.45	8255.15	13090.4	5541.03
$r_p[m]$	85.656	83.3468	85.269	83.46
$\Delta v[m/s]$	2.18×10^8	2.805×10^8	1.05455×10^8	6.20×10^8
	815.66	0.378272	0.758372	198.385
Uranus Flyby				
Date T_3	24 – 08 – 1986	26 – 05 – 1987	19 – 02 – 1985	02 – 10 – 1991
$v_{\infty-in}[km/s]$	2446666.5	2446942.125	2446120	2448531.5
$v_{\infty-out}[km/s]$	12877.1	11900.1	17378.6	6641.43
$\delta[deg]$	17661.6	11900.2	17379.4	7303.52
$r_p[m]$	22.164	18.1143	26.0872	5.82
$\Delta v[m/s]$	9.21×10^7	2.583×10^8	7.76536×10^7	2.39×10^9
	3728.92	0.025	0.654346	626.252
Neptune Arrival				
Date T_4	25 – 08 – 1989	15 – 09 – 1991	25 – 04 – 1989	06 – 08 – 1998
Arrival $V_{in}[m/s]$	2447763.5	2448484.125	2447277	2451031.5
Results	21551.0	15349.0	21862.2	8277.72
Total cost Δv	14830.12	9414.075	9856.34	10819.9

4.2.2 Results:

Figure 3 provides a comparison between the data from the true Voyager-II mission flow, the Pykep+Pagmo optimizer data and our new code. The $\alpha = 1$ trajectory minimizes delta-v but requires the longest travel time as it aims to use the least amount of thrust. Differences in arrival times become apparent during flybys of Jupiter and Saturn, with the $\alpha = 0.7$ trajectory outperforming the real mission substantially as seen in Fig. 3(b). Notably, the $\alpha = 0.7$ trajectory benefits from higher incoming velocities, reducing travel time and arriving at the final destination 4 months before the real Voyager. However, the real mission was able to transfer quickly from Uranus to Neptune but at the cost of a high delta-v maneuver, which is seen as the big spike at the end of Fig. 3(b) (orange curve). Moreover, the inefficiency of only focusing on delta-v in terms of overall mission planning is highlighted by ESA's Pykep/Pagmo model shown in purple in Fig. 3. ESA's model provided with the same parameters would have reached Neptune in August of 1998, almost a decade after the real mission. Moreover, the total delta-v required for the purple trajectory, using ESA's Pykep/Pagmo model, was still found to be higher than the $\alpha = 1$ or even the $\alpha = 0.7$ solution. For further confirmation of the performance of our new code a comparison between the here used GA code and the real Galileo mission are provided in the Appendix.

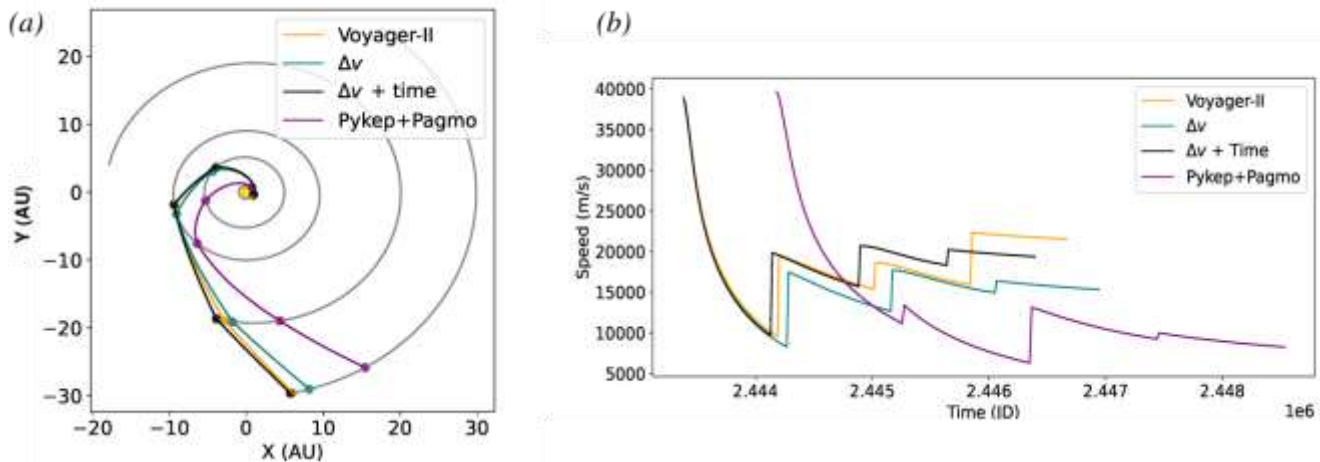


Figure 3 (a) Trajectory plot and (b) Velocity evolution for the Voyager-II mission in comparison with the real mission data and *Pykep+Pagmo* optimizer.

The dependence of the arrival date as a function of α is also analyzed by creating trajectories with continuously increasing prioritization of time in the objective function. Figure 4(a) illustrates six trajectories with levels of α in the range of [0.5, 1.0] that have been constructed using the GA parameters detailed in Tab. 3.

Figure 4(a) graphically portrays the correlation between total Δv and arrival dates across various α values. In this representation, the color map intuitively signifies distinct α values, with darker shades denoting smaller α values. As evidenced in the graph, a lower α value signifies a reduced emphasis on minimizing Δv (with a higher priority on reaching the destination sooner), consequently leading to an augmented total Δv requirement. Using a range for α from [0.5, 1.0], arrival dates spanning across 4 years, from May 1991 to May 1887, can be targeted at will. Of course further increases of β will inevitably lead to even earlier arrival dates at increasing costs of Δv . The limits of tuning the prioritization parameters are discussed in section 5. Nonetheless, thanks to the appropriate scaling of the terms in the fitness evaluation of each individual, allowing for the comparison of a quantity measured in m/s to a calendar date in a meaningful way, the tuning of the prioritization parameters α and β is of order unity. Utilizing the same trajectories, the additional time at the final destination as a function of the extra percentage of Δv required compared to the minimum Δv solution is constructed in Fig. 4(b). This graphic underscores the capacity of marginal increases in fuel usage to substantially expedite arrival dates at the final destination.

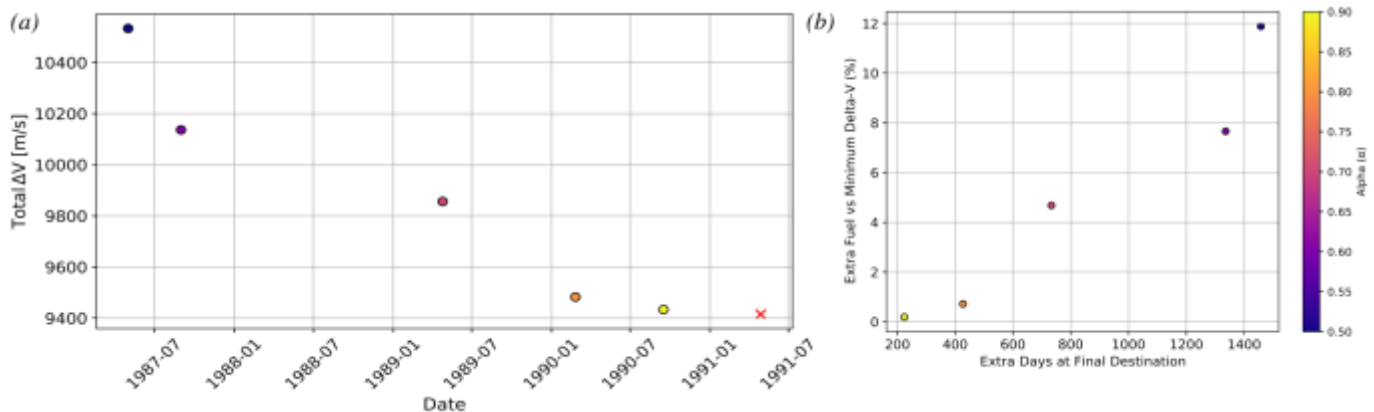


Figure 4 (a) Total Δv vs Arrival Date and (b) Trade-off between Extra Days and Fuel Cost.

All over Fig. 4 effectively depicts the delicate balance between the time of arrival and the associated Δv costs in the realm of space mission planning. Lower values of α allow the spacecraft the opportunity to place greater importance on achieving an earlier arrival, albeit at the cost of heightened fuel consumption. This pivotal trade-off, underscored by our data, demonstrates the critical importance of strategic decision-making for mission planners who must navigate the intricate interplay between scientific objectives and resource constraints.

4.3 Performance

The evolution of the generations and its fittest individual's fitness in the Voyager I and II test case is shown in Fig. 5. Following, Fig. 6 show the fitness evolution on some generations for the Voyager I mission, on it, we observe how the overall fitness is largely improved through early generations and slowed through the final ones. As long as the crossover is high (closer to 1) and the mutation is reasonably minor (between 0.05 and 0.3), the precise values selected for the crossover and mutation operators have no discernible effect on performance. High crossover is necessary to produce new genetic material, but low mutation prevents excellent genetic material from being altered after crossover. It should not, however, be zero in order to prevent the population from becoming trapped in local minima, which is a common drawback of many gradient-based techniques [35], and to enable fresh material to emerge when the population tends to stagnate.

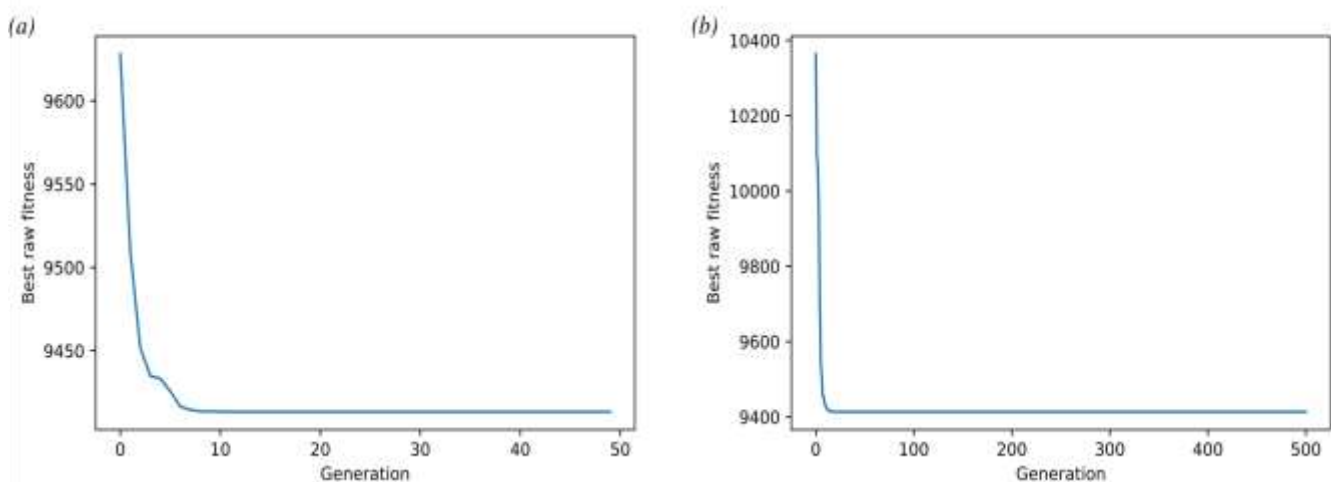
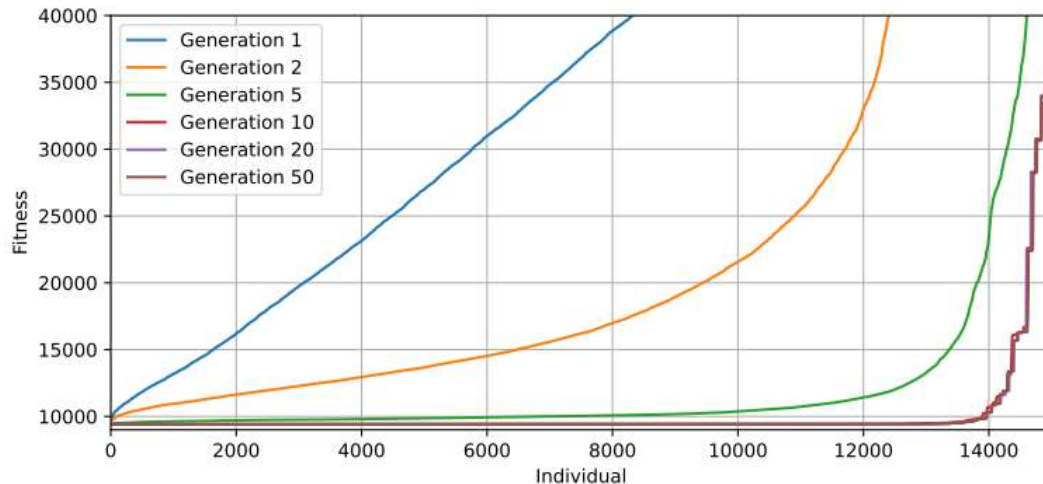


Figure 5 Fittest individual evolution for (a) Voyager-I and (b) Voyager-II mission (cf. Fig. 2).**Figure 6** Ordered population by fitness evolution along generations for Voyager-I mission (cf. Fig. 2).

5 CONCLUSION

In this paper, we presented a multi-objective optimization approach considering arrival time and Δv for MGA interplanetary trajectories. It is shown that simply minimizing Δv , the travel time to the destination is prolonged to the point where it does not represent a realistic mission trajectory as other mission cost drivers, including time and design constraints, are penalized heavily. Our approach performs better in terms of Δv efficiency and arrival time than both the Pykep/Pagmo solution and the actual Voyager II mission (amongst others), as shown in Tab. 4. Notably, our method shows a large advantage for $\alpha = 0.7$, emphasizing potential massive economic savings by obtaining an earlier arrival with significantly less fuel usage. All told, more than one-third of the fuel for the entire trajectory might have been saved, and the final destination could have been reached months sooner. For the Voyager II example, a just 4.7% increase in delta-v from the minimal delta-v trajectory can result in a flight duration reduction of more than 2 years ($\alpha = 0.7$).

Using the algorithm developed in this paper, a family of trajectories can be explored by tuning α to effectively target a desired arrival date. Moreover, the implementation of time in the cost function describe in Eq. (2.3), starts penalizing the trajectory starting from the first possible launch date. This is in contrast to many algorithms discussed before that include time in space as the optimizing parameter, which fails to encapsulate costs related to the launch (and especially delays), ground staff, scientist, etc. In a future work, it is planned to test more complex case studies and finishing the outer loop and address in addition the optimization of flyby sequence. The construction of long space mission trajectories would then be made easier by this tool, which would require only a final destination and user-specified priorities between Δv and time, which might be determined using the parameters α and β .

Acknowledgments

S. A. is a Serra H nter Fellow. This work was supported by the Spanish Government under grants PID2023-150014OB-C21 and PID2023-150029NB-I00.

Conflicts of Interest

The authors have no conflicts of interest.

Data Availability Statement

The data that support the findings of this study will be made available by the corresponding author on reasonable request.

APPENDIX

Galileo mission

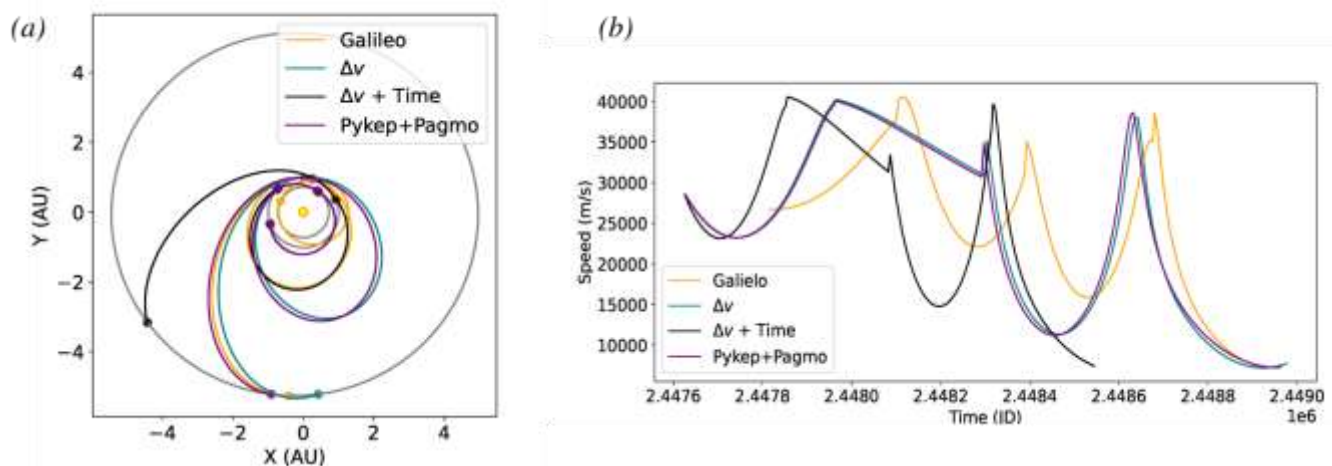


Figure 7 (a) Trajectory plot and (b) Velocity evolution for the Galileo mission (sequence Earth-Venus-Earth-Earth-Jupiter (EVEEJ)) in comparison with the real mission data and *Pykep+Pagmo* optimizer.

Following, the algorithm presented in this paper is shown with the example of the Galileo mission which had a flyby order of Venus-Earth-Earth before finally arriving at Jupiter (EVEEJ).

Figure 7 provides a comparison between the data from the true Galileo mission flow, the *Pykep+Pagmo* optimizer data and our new code. The corresponding data are listed in Table 5 using the same parameter of Tab. 3 as before.

Table 5 Galileo results comparison using parameter of Tab. 3, cf. Fig. 7.

Trajectory Results				
Real mission minimum Δv and Time Tradeoff <i>Pykep+Pagmo</i> for EVEEJ				
		($\alpha = 1$)	($\alpha = 0.5$)	
Earth Departure				
Date T_0	18-10-1989	10-04-1989	08-04-1989	10-04-1989
	2447817.5	2447627.25	2447625	2447626.5
Departure Δv [m/s]	3956.08	7329.11	7418.27	7301.78
Venus Flyby				
Date T_1	10-02-1990	12-12-1989	10-12-1989	12-12-1989
	2447932.5	2447872.5	2447870	2447872.5

$v_{\infty-in}[km/s]$	6211.31	6165.29	6312.31	6009.25
$v_{\infty-out}[km/s]$	6222.16	6178.85	6317.08	6140.34
$\delta[deg]$	32.0243	68.328	63.9678	68.16
$r_p[m]$	2.202×10^7	6.643×10^6	7.22837×10^6	6.758×10^6
$\Delta v[m/s]$	8.166	7.182	2.64229	69.03
Earth Flyby (1)				
Date T_2	08-12-1990	07-02-1990	04-02-1990	08-02-1990
$v_{\infty-in}[km/s]$	2448233.5	2447930.125	2447927	2447930.5
$v_{\infty-out}[km/s]$	8960.85	9576.54	10279.9	9345.45
$\delta[deg]$	5012.07	9534.7	10288.9	9324.78
$r_p[m]$	100.89	43.95	22.0235	46.347
$\Delta v[m/s]$	4.712×10^6	7.332×10^6	1.59474×10^7	7.065×10^6
Earth Flyby (2)				
Date T_3	08-12-1992	20-12-1992	15-10-1991	06-12-1992
$v_{\infty-in}[km/s]$	2448964.5	2448976.5	2448544.5	2448962.5
$v_{\infty-out}[km/s]$	5012.93	9533.31	10300	9320.84
$\delta[deg]$	8904.16	9541.97	10822.8	9193.31
$r_p[m]$	11.83	23.9571	37.96	44.2541
$\Delta v[m/s]$	4.37×10^7	1.671×10^7	7.06008×10^6	7.8048×10^6
Jupite Arrival				
Date T_4	07-12-1995	27-03-1996	06-03-1994	30-09-1995
Arrival $V_{in}[m/s]$	2450058.5	2450169.5	2449417.5	2449990.5
Results	7330.42	7610.06	7541.67	7242.09
Total cost Δv	9110.45	7371.56	7791.31	7498.13

As can be seen in Tab. 5, the $\alpha = 0.5$ trajectory would have allowed Galileo to study Jupiter and its moons for over two years longer compared to the minimum Δv trajectory (20 months before the real Galileo mission). Moreover, this could have been achieved with 1, 319.14 m/s less Δv than the actual trajectory flown by the satellite. The additional time at the destination in combination with the fuel saved using the ($\alpha = 0.5$) trajectory could have allowed for more science being done by the mission at a reduced overall cost (including the various sources of cost discussed before).

REFERENCES

- [1] Kohlhasse, C., and Penzo, P. A. (1977) Voyager mission description. Space Science Reviews, 21, 77–101.
- [2] PDS (2017). The planetary data system - mission information.
- [3] NASA (2011) Cassini–huygens: Quick facts. Tech. rep., Jet Propulsion Lab.
- [4] NASA (2009) Cassini/huygens probe entry, descent, and landing (edl) at titan independent technical assessment. Tech. rep., Jet Propulsion Lab.
- [5] Matson, D. L., J., S. L., and Lebreton, J. (2002). The cassini/huygens mission to the saturnian system. Space Science Reviews, 104, 1–58.
- [6] Hechler, M. (1997) Rosetta mission design. Advances in Space Research, 19(1), 127–136. Proceedings of the E1.1, B1.5, and C2.4 Symposia of COSPAR Scientific Commissions E, B and C.
- [7] Izzo, D., Becerra, V. M., Myatt, D. R., Nasuto, S. J., and Bishop, J. M. (2006) Search space pruning and global

optimisation of multiple gravity assist spacecraft trajectories. *J Glob Optim*, 38, 283–296.

[8] Goldberg, D. E. (1988) *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley Publishing Company.

[9] Englander, J., Conway, B., and Williams, T. (2011) Optimal autonomous mission planning via evolutionary algorithms, 21st aas/aiaa space flight mechanics meeting, american astronomical soc. 140, 833–852.

[10] Englander, J., Conway, B., and Williams, T. (2012) Automated Mission Planning via Evolutionary Algorithms. *Journal of Guidance, Control, and Dynamic*, 35(6), 1878–1887.

[11] Xu, R., Cui, P., Qiao, D., and Luan, E. (2007) Design and optimization of trajectory to near-earth asteroid for sample return mission using gravity assists. *Advances in Space Research - ADV SPACE RES*, 40, 12, 220–225.

[12] Molenaar, S. (2009) Optimization of interplanetary trajectories with deep space maneuvers - model development and application to a uranus orbiter mission.

[13] Izzo, D., Becerra, V. M., Myatt, D. R., Nasuto, S. J., and Bishop, J. M. (2007) Search space pruning and global optimisation of multiple gravity assist spacecraft trajectories. *J Glob Optim*, 283–296.

[14] Vasile, M., and De Pascale, P. (2006). “Preliminary design of multiple gravity-assist trajectories”. *Journal of Spacecraft and Rockets*, 43(4), 794–805.

[15] Vasile, M., Minisci, E., and Locatelli, M. (2010) Analysis of some global optimization algorithms for space trajectory design. *Journal of Spacecraft and Rockets*, 47(2), 334–344.

[16] Gad, A., and Abdelkhalik, O. (2011) Hidden genes genetic algorithm for multi-gravity-assist trajectories optimization. *Journal of Spacecraft and Rockets*, 48(4), 629–641.

[17] Petropoulos, A. E., and Longuski, J. M. (2004) Shape-based algorithm for the automated design of low-thrust, gravity assist trajectories. *Journal of Spacecraft and Rockets*, 41(5), 787–796.

[18] Abdelkhalik, O. (2021). *Algorithms for Variable-Size Optimization - Applications in Space Systems and Renewable Energy*, Vol. 1. CRC Press, Taylor & Francis Group.

[19] Englander, J., Conway, B., and Williams, T. (2013) MIDACO on MINLP space applications. *Advances in Space Research*, 51, 1116–1131.

[20] Zhu, K., Li, J., and Baoyin, H. (2009) Multi-swingby optimization of mission to Saturn using global optimization algorithms. *Acta Mechanica Sinica*, 25(6), 839–845.

[21] Ceriotti, M. (2010) Global optimisation of multiple gravity assist trajectories.

[22] Standish, M., and Williams, J. (2006) *Orbital Ephemerides of the Sun, Moon, and Planets*. 01.

[23] Folkner, W. M., Williams, J. G., Boggs, D. H., Park, R. S., and Kuchynka, P. (2014) *The Planetary and Lunar Ephemerides DE430 and DE431*. Interplanetary Network Progress Report, 42-196, 02, 1–81.

[24] NASA (1964) Approximate positions of the planets. Tech. rep., Jet Propulsion Lab.

[25] Jordon, J. F. (1964) The application of lambert’s theorem to the solution of interplanetary transfer problems. Tech. Rep. 32-521, NASA - Jet Propulsion Lab.

[26] Curtis, H. (2015). *Orbital Mechanics: For Engineering Students*. Aerospace Engineering. Elsevier Science.

[27] Izzo, D. (2014) Revisiting lambert’s problem. *Celestial Mechanics and Dynamical Astronomy*, 03.

[28] Bond, V. R., 1969. Matched-conic solutions to round-trip interplanetary trajectory problems that insure state-vector continuity at all boundaries, Vol. 4342. National Aeronautics and Space Administration.

[29] Wagner, S., Kaplinger, B., and Wie, B. (2012) Gpu accelerated genetic algorithm for multiple gravity-assist and impulsive v maneuvers.

[30] Mosher, T. (1998) Spacecraft Design Using A Genetic Algorithm Optimization Approach. 1998 IEEE Aerospace Conference Proceedings, 3, 123–134.

[31] M. A. Vavrina, J. A. Englander, S. M. P., and Hughes, K. M. (2017) Global, Multi-Objective Trajectory Optimization with Parametric Spreading. *Advances in the Astronautical Sciences AAS/AIAA Astrodynamics Conference*, 162.

[32] O’Neil, W., Ausman, N., Gleason, J., Landano, M., Marr, J., Mitchell, R., Reichert, R., and Smith, M., (1997) Project galileo at jupiter. *Acta Astronautica*, 40(2), 477–509. Enlarging The Scope of Space Applications.

- [33] Izzo, D. (2012) Pygmo and pykep: Open source tools for massively parallel optimization in astrodynamics (the case of interplanetary trajectory optimization).
- [34] Biscani, F., Izzo, D., and Yam, C. H. (2010). A global optimisation toolbox for massively parallel engineering optimisation.
- [35] Vasile, M., Minisci, E., and Locatelli, M. (2008). On Testing Global Optimization Algorithms for Space Trajectory Design - AIAA/AAS Astrodynamics Specialist Conference and Exhibit.
- [36] <https://esa.github.io/pykep/>
- [37] <https://github.com/esa/pykep>
- [38] <https://github.com/esa/pagmo2>