Genetic algorithm for multi-gravity assist interplanetary trajectory optimization

D. Huterer Prats*, I. Diaz Cilleruelo†, and S. A. Altmeyer‡
Universitat Politecnica de Catalunya, EETAC, C. Esteve Terradas, 7. 08860 Castelldefels, Spain

Interplanetary and interstellar 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 huge numbers of 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 eventually may contain the optimal trajectory. In this paper, we present the development of a genetic algorithm that is capable of determining optimal interplanetary trajectories, where tradeoffs between minimizing Δv and an earlier arrival time 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.

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

I. Nomenclature

Δv	=	delta-v
$\vec{v}$	=	velocity
a	=	semi-major axis
e	=	eccentricity
I	=	inclination
L	=	mean longitude
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

II. Introduction

A gravity assist or flyby is a technique used by spacecrafts to change their velocity by doing a close approach to a planet or body within their trajectory. This technique can add or subtract momentum to the spacecraft without the use of its own propulsion system. It is normally used to reach faraway bodies where the required delta-v is too high. By progressively increasing the speed during the trajectory this problem is minimized. This technique can have virtually zero cost if the trajectory is properly designed. For this reason, it is extremely attractive from a mission design point of view to use gravity assists effectively. Reducing the delta-v (Δv) requirement can allow devoting more mass for other spacecraft components or subsystems. In addition, a less powerful launcher may be used. As a matter of fact, this technique allowed Voyager I and II missions to reach their destinations. Without the several flybys they performed

*Master student, daniel.huterer@estudiantat.upc.edu
†Software engineer, Circontrol S.A., ikerdiaz1312@gmail.com
‡Associate Professor, Department of Physics, sebastian.andreas.altmeyer@upc.edu.
I. Diaz Cilleruelo and D. Huterer Prats contributed equally to this work

during the trajectory, the launcher used (Titan III-E/Centaur D-IT) [1, 2] would not have been capable of providing the required Δv.

Designing the most optimal trajectory for a mission that involves multiple gravity assist maneuvers is complex and also requires taking other factors that have to be taken into account, such as the time of flight. 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 targets/goals of the mission. At the same time it also has an economical factor as there are hidden costs that are beyond fuel costs which are way more important. Doubtless time in space is expensive and as such, all the ground stuff (depending on the type of mission 100s of persons) has to be paid even when the spacecraft is still on it's way to the target. 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 depend on the profile of the mission. This time consideration further allows the targeting of a specific arrival date which is required by some mission to leverage the local geometry at the destination occurring only at a certain date. Moreover, there are a lot of hidden costs associated with a spacecraft traveling through space to its destination as their is a numerous ground control and science staff that is being paid, while providing little to no service during the travel phase. Furthermore, as the 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. Therefore, the most critical factor in the design for the majority of missions are Δv and arrival time. Over the course of this paper, it will be shown that the tradeoff at different prioritization levels of the two mission design factors creates a family of trajectories depending on the user's needs rather than finding a single minimum Δv trajectory.

This optimization problem is NP-Hard, i.e. 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.

There are some deterministic optimizers (optimization algorithms) [3] available, even though the most common approach is to use an evolutionary algorithm [4–6]. 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 offer an alternative that is much simpler yet can still provide optimal results [6].

In this paper, we present a new developed (really) simple genetic algorithm (GA), to showcase its effectiveness to optimize a multi-planetary gravity assist (MGA) interplanetary trajectory in terms of a trade-off/balance between arrival time t and total Δv of the trajectory. 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.

III. 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 perigee passage at each flyby, while MGA-DSM is an extension in which 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.

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. This objective can be converted to a multiple-objective function for the optimization problem.

$$\underset{\vec{X}}{\text{minimize}} \left(\sum_{i \in \vec{X}} \Delta v_i + \sum_{i \in \vec{X}} t_i \right) \tag{1}$$

where Δv_i is the Δv at each time event t_i. The input to the problem, i.e. the optimization variable is a time-dependent vector $\vec{X}$.

$$\vec{X} = [t_0, t_1, \ldots, t_n]. \tag{2}$$

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, i.e. pre-defined by the user. The sequence of planets could be as well optimized in the first step as in [6–8]. The so-obtained optimal flyby sequence could then be used for MGA optimization as presented in this paper. The two parts, i.e. 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 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 [4] 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 $f(x)$ and $t(x)$ (representing the Δv and time of the trajectory), the algorithm is able to assess the relative optimally 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, i.e. how good the output is given an input. This fitness is assessed based on a fitness function $c(\vec{X})$:

$$c(\vec{X}) = f(\vec{X}) + g(\vec{X}) + t(\vec{X}), \tag{3}$$

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 $t(\vec{X})$ is the time taken to the final destination. While typically a higher value is a better fitness, since this can be considered a cost function, the lower the value of $c(x)$, the fitter an individual is.

The objective function for the current investigation is provided by Eq. (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 optimise 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 centred 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)), 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, i.e. the one with the smallest cost in terms of Δv and time calculated using 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:

$$1 = \alpha + \beta. \tag{4}$$

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

IV. 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 asses 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.

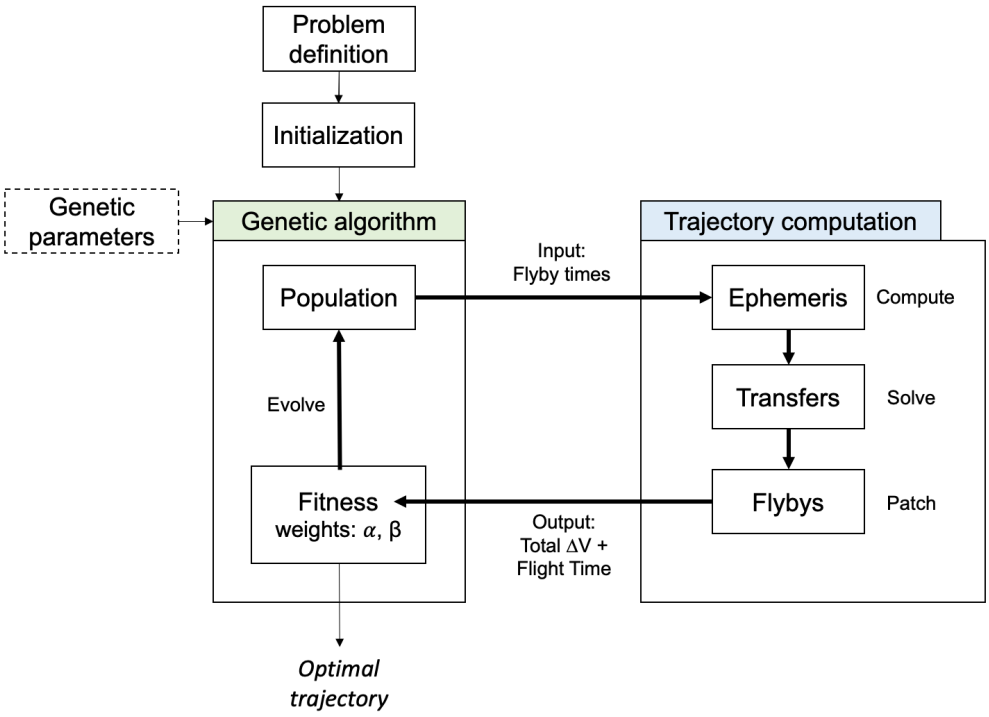

Fig. 1 Flowchart - Tentative Code diagram

A. 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.

1. Ephemeris

We recall that the trajectory is defined by a sequence of planets and a time vector $\vec{X}$. Thereby time vector defined 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 [9]. 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 [10].

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 [9] (web format by NASA at [10, 11]). With all provided data for a planet i.e. the Keplerian elements and their rates (a, e, I, L, ω, Ω), the orbital parameters are determined at a time T_{eph}. The $\vec{r}$ and $\vec{v}$ are computed using the following five steps:

(i) Compute the value of each of that planet's six elements.
(ii) Compute the argument of perihelion ω, and the mean anomaly M.
(iii) Modulus the mean anomaly. Obtain the eccentric anomaly E from the solution of Kepler's equation.
(iv) Compute the planet's heliocentric coordinates in its orbital plane r with the x'-axis aligned from the focus to the perihelion.
(v) Compute the coordinates r_{ecl} in the J2000 ecliptic plane, with the x-axis aligned toward the equinox.

The ephemeris puts down the first piece of the trajectory computation puzzle, which is to locate the planets at their corresponding moments in time.

2. 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.

Lambert's problem [12] was posed by J. H. Lambert in the 18th century. In his theorem Lambert states: *"The transfer time of a body moving between two points on a conic trajectory is a function only of the sum, of the distances of the two points from the origin of force, the linear distances between the points and the semi-major axis of the conic"* [12]. Therefore, if the time of flight T between two points P_1 and P_2 is determined, then Lambert's problem is to determine the trajectory between these two points P_1 and P_2. The trajectory is determined 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 [13] Sect 5.3. In other words, the next point in the trajectory can be inferred from the previous one as a consequence of the following equations

$$\vec{r} = f\vec{r}_0 + g\vec{v}_0 \tag{5}$$

$$\vec{v} = \dot{f}\vec{r}_0 + \dot{g}\vec{v}_0 \tag{6}$$

These equations (2.125 and 2.126 in the book by Curtis [13]) describe the concept of obtaining the position and velocity at a moment in time only using the initial values. In (5) and (6) f and g are the Lagrange coefficients [13] Sect 2.11.

Using the times defined for the trajectory, the travel time T between two planets is defined ($t_{P_2} - t_{P_1}$). Then, using the $\vec{r}$ state vector from the ephemeris, planets points P_1 and P_2 are defined as well. The Lambert problem can then be solved. The solution is the required velocity $\vec{v}_0$ from planet P_1 ($\vec{r}_0$) at t_{P_1} in order reach planet P_2 at t_{P_2}.

The Lambert solver implemented in the algorithm for this work was developed by Dario Izzo and Gooding [14]. The solver is developed in C++ and forms part of the open-source ESA's Pykep* library, a scientific library providing basic tools for astrodynamics research.

3. 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

*https://esa.github.io/pykep/

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) ([13] Sect 8.4) of a planet, the spacecraft follows a hyperbolic planetocentric trajectory [15]. 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 perigee radius [16]. The difference in velocity will be the Δv impulse required. The velocities relative to the planet are obtain as follows

$$\vec{v}_{\infty-in} = \vec{V}_{in} - \vec{V}_p \tag{7}$$

$$\vec{v}_{\infty-out} = \vec{V}_{out} - \vec{V}_p \tag{8}$$

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 perigee radius r_p we must determine the semi-major axis of both trajectories (at pre- and post-perigee passage).

$$a_{in} = -\frac{\mu_p}{v_{\infty-in}^2} \tag{9}$$

$$a_{out} = -\frac{\mu_p}{v_{\infty-out}^2}. \tag{10}$$

The turning angle δ between the asymptote of both trajectories at the bounds of the SOI [13] is computed as

$$\delta = \left(\frac{\vec{v}_{\infty-in} \cdot \vec{v}_{\infty-out}}{v_{\infty-in} \cdot v_{\infty-out}}\right). \tag{11}$$

The perigee can be expressed as

$$r_p = a_{in}(1 - e_{in}) = a_{out}(1 - e_{out}), \tag{12}$$

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

$$\delta = \sin^{-1}\left(\frac{1}{e_{in}}\right) + \sin^{-1}\left(\frac{1}{e_{out}}\right). \tag{13}$$

Then, Eq. (12) can be used to obtain a function with single dependence on e_{out}. Afterward, Eq. (12) could be solved for e_{out} using an iterative method like Newton-Raphson [16]. However, this approach is unnecessary, if we assume that the trajectory will be correctly joined with the necessary Δv maneuver at the perigee. Under this assumption we can model the full trajectory as an un-powered flyby from the perigee passage. In that case, we can express the turning angle as follows (see book [13] example 8.6):

$$\delta = 2\sin^{-1}\left(\frac{1}{e}\right). \tag{14}$$

We isolate the eccentricity from (14),

$$e = \sin^{-1}\left(\frac{\delta}{2}\right). \tag{15}$$

Using (10) and (14) we can solve (12). Finally, the required Δv applied at perigee is

$$\Delta v = \left|\sqrt{v_{\infty-in}^2 \frac{2\mu_p}{r_p}} - \sqrt{v_{\infty-out}^2 \frac{2\mu_p}{r_p}}\right| \tag{16}$$

Using these calculations, we can patch the two trajectories by applying the Δv impulse. We determine the perigee 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 perigee radius or turning angle when patching the two trajectories. For example, the perigee 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. (16)).

B. 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 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', i.e. 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, genetic algorithms are particularly well suited as they

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.

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. 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.

V. Code validation

In order to assess the performance of our approach, the new algorithm has been tested with both, real missions (Voyager I [1] and Galileo [17], both targeted at reaching Jupiter) and other well-established algorithms, the Pykep and Pagmo libraries. Pykep[†] [18] is an ESA library for astrodynamics research, while Pagmo[‡] [19] (an ESA library too) is used for massively parallel optimization. Pagmo optimized the MGA problem defined with Pykep.

[†] https://github.com/esa/pykep

[‡] https://github.com/esa/pagmo2

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 perigee 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 where devoted to daily decimal precision. While 3 bits where 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 even in the trajectory, i.e. 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.

A. Voyager I

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.

Both the algorithm's output and the inputs used in the GA are shown in Table 5. The population size was fixed at a large number of 15,000, 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 labelled 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 utilising 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.

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)

Table 1 Genetic Algorithm Parameters Voyager-I

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

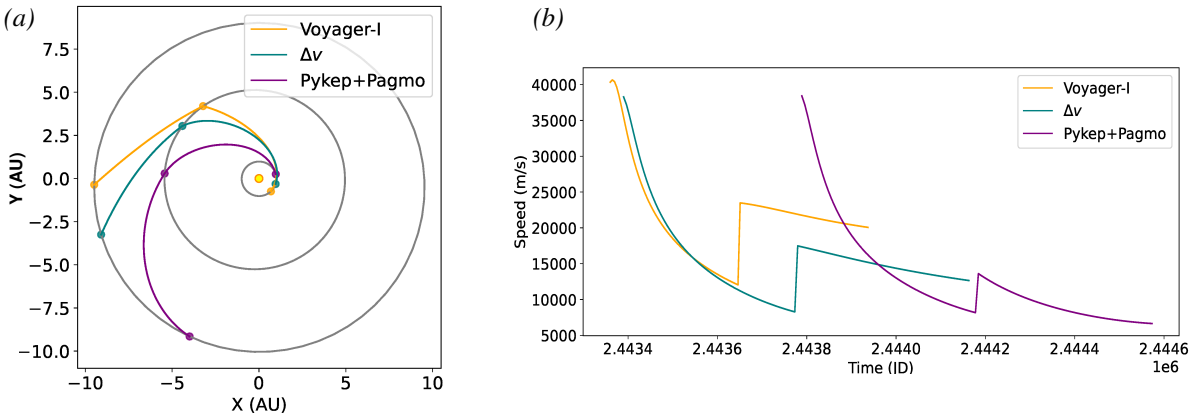

Fig. 2 (a) **Trajectory plot and** (b) **Velocity evolution for the Voyager-I mission in comparison with the real mission data and *Pykep+Pagmo* optimizer.**

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.

2. Results:

The precise outcomes yielded by the genetic algorithm are presented in Section VI in table 5 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. Meanwhile, this flyby is executed at a higher perigee radius, characterized by a less abrupt turning angle, and thereby necessitates a Δv impulse of only $0.34 m/s$. 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 $9.41 km/s$, which is smaller as the real mission necessitates $10.33 km/s$. 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.

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 $0.91 km/s$ during 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 $20.11 km/s$ at Saturn, compared to the optimal trajectory's $12.7 km/s$. This discrepancy can be attributed to the lower perigee radius attained by the real mission, imparting a greater velocity.

B. Voyager II

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

Solution	Real mission	Pykep/Pagmo	Ours
Earth departure	1977-Aug-05	1978-Oct-08	1977-Sep-04
Jupiter flyby	1979-Mar-05	1980-Nov-30	1979-Oct-17
Saturn arrival	1980-Nov-12	1986-May-23	1982-Apr-24
Duration (days)	1529	2784	1693
Total Δv *(using our model)*	10331.5	9568.16	9414.05

Table 2 Comparison of Voyager I mission solutions. Values of Δv are computed using the reference dates in our algorithm.

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 15,000 individuals to 50,000 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. 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 table 4.

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 3 Genetic Algorithm Parameters Voyager-II

2. Results:

The trajectories of the true mission flown and our two trajectories are shown in Fig. 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. Notably, the $\alpha = 0.7$ trajectory benefits from higher incoming velocities, reducing travel time. However, the real mission was able to transfer quickly from Uranus to Neptune but at the cost of a high delta-v maneuver. 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.

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 Table 3.

Figure $4(a)$ graphically portrays the correlation between total Δv and arrival dates across various α values. In this

Solution	Real mission	Pykep/Pagmo	minimum Δv ($\alpha = 1$)	Δv and Time Tradeoff ($\alpha = 0.7$)
Earth departure	1977-Aug-20	1979-Nov-05	1977-Sep-04	1977-Sep-01
Jupiter flyby	1979-Jul-09	1981-Jul-07	1979-Oct-17	1979-May-03
Saturn flyby	1981-Aug-26	1984-Nov-27	1982-Apr-24	1981-Mar-18
Uranus flyby	1986-Aug-24	1991-Oct-02	1987-May-26	1985-Feb-19
Neptune arrival	1989-Aug-25	1998-Aug-06	1991-Sep-15	1989-Apr-25
Duration (days)	4388	6849	5124	4255
Total Δv *(using our model)*	14380.12	10819.9*	9414.08	9856.34

Table 4 Comparison of Voyager II mission solutions. Values of Δv are computed using the reference dates in our algorithm.
* **Pykep/Pagmo model**

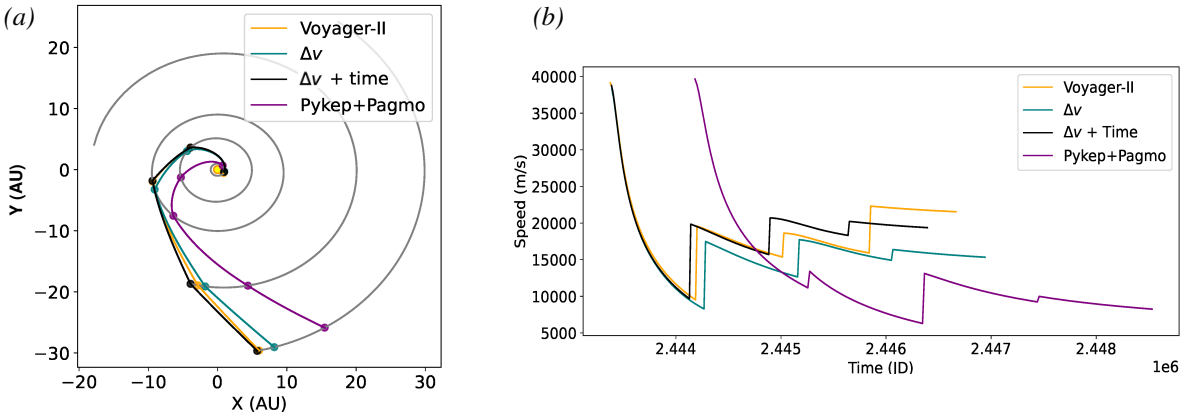

Fig. 3 (a) **Trajectory plot and** (b) **Velocity evolution for the Voyager-II mission in comparison with the real mission data and *Pykep+Pagmo* optimizer.**

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 1997 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 VI. 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.

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.

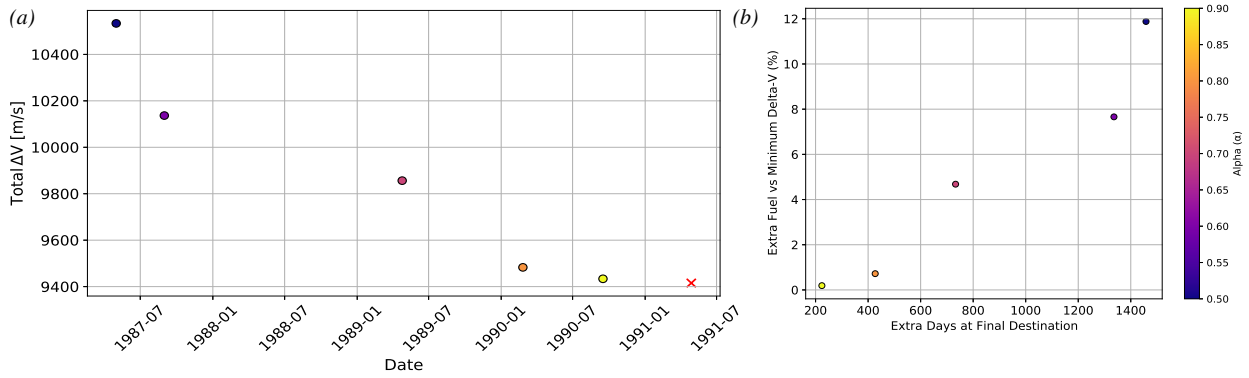

Fig. 4 (a) **Total** Δv **vs Arrival Date and** (b) **Trade-off between Extra Days and Fuel Cost.**

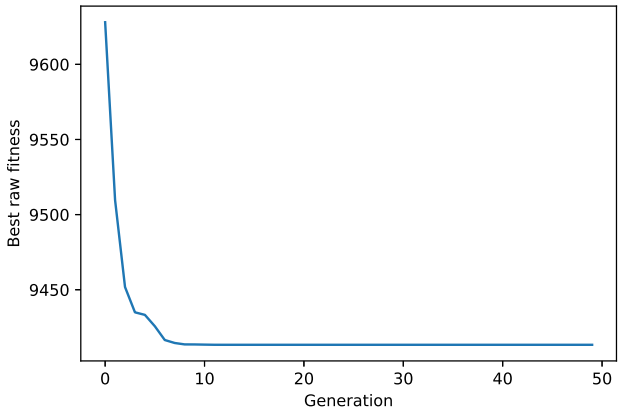

Fig. 5 Fittest individual evolution for Voyager-I mission (cf. Fig. 2).

C. Performance

The evolution of the fittest individual's fitness throughout all generations in the Voyager I test case is shown in Figure 5. 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 [20], and to enable fresh material to emerge when the population tends to stagnate.

VI. Conclusion

Although many interplanetary trajectory optimisation algorithms just minimise Δv without taking time into account [3, 21], our method takes time into account as an important component of the objective function. This is an important factor to take into account because optimising Δv on its own has been shown to increase travel time to the final destination by several years.

Our approach performs better in terms of Δv efficiency and arrival time than both the Pykep/Pagmo solution and the actual Voyager II mission, as shown in Table 4. Notably, our method shows a large advantage for $\alpha = 0.7$, emphasising 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. This is especially important for deep space missions, when launch costs can be significantly reduced by using less satellite fuel. Other than the satellite's mass, however, there are a lot of other essential components of a

satellite mission that are related to the mission's duration and travel time that can add even more to the overall financial costs. 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 ($\alpha = 0.7$).

It is crucial to recognise that although a mission's fuel cost has historically been directly related to its overall Δv requirements, this may not necessarily be the primary financial element determining a mission's viability. Our method optimises mission planning and resource allocation in space exploration to reflect this financial complexity. The obvious drawback of concentrating solely on Δv is that it might unduly penalise modest thrust increases that would otherwise optimise the mission profile. Beyond fuel costs, the main cost driver is frequently present in extensive satellite missions carrying a plethora of instruments. It is actually the costs associated with paying the full time staff of satellite operations and hiring researchers, whose participation might not be as great while the satellite is in the transit phase between its destination and its dormant state.

Despite being a mission parameter that is both scientifically and commercially significant, the total satellite mass at the destination was not selected as the second parameter in the objective function since it is dependent on the particular launcher that was selected [22]. It was decided to concentrate on the arrival time at the destination, which is an objective and physical element of a mission, because different rockets supplied by governments and commercial enterprises alike have distinct non-physical limits.

Mission designers can explore a family of trajectories with varying degrees of α by adjusting the priority between Δv and arrival time. This optimisation strategy using α and β is similarly limited in the amount of tuning that can be done, even though our algorithm can produce many distinct trajectories as opposed to simply the unique minimal Δv trajectory. Regardless of the excessive amount of Δv required, the algorithm will attempt to nearly travel in a straight line from planet to planet for values of β close to 1. However, this will not adhere to the spacecraft's structural limitations, such as its maximum allowable loads. Furthermore, flying directly to the final destination would take less time than flying as quickly as possible to each planet in the sequence if the only objective is to reach it and not really care about the flyby planets. The here presented evolutionary optimization approach can also be used to find the optimal flyby sequence, therefore having both the inner and outer loop optimized using genetic algorithms [5] Therefore, finishing the outer loop will be the next task and future work. After finishing, the algorithm finds the optimal sequence and then refines it even further. 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.

Appendix

Table 5 provides an overview of the input parameters used in the GA as well as the output of the algorithm, the Trajectory optimization for Voyager-I. For the Voyager-II mission, a more detailed comparison for different parameters, real data mission together with Pykep+Pagmo data are listed in Tab. 6 (Extended version of Tab. 4).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] Kohlhase, C., and Penzo, P. A., "Voyager mission description," *Space Science Reviews*, Vol. 21, 1977, pp. 77–101. https://doi.org/10.13140/RG.2.2.28422.09289.

[2] PDS, "The Planetary Data System - Mission Information," , 2017. URL https://web.archive.org/web/20170220172041/https://starbrite.jpl.nasa.gov/ds-view/pds/viewMissionProfile.jsp?MISSION_NAME=VOYAGER.

[3] Izzo, D., Becerra, V. M., Myatt, D. R., Nasuto, S. J., and Bishop, J. M., "Search Space Pruning and Global Optimisation of Multiple Gravity Assist Spacecraft Trajectories," *J Glob Optim*, Vol. 38, 2006, p. 283–296. https://doi.org/10.1007/s10898-006-9106-0.

[4] Goldberg, D. E., "Genetic Algorithms in Search Optimization and Machine Learning," Adison-Wesley Publishing Company, 1988.

GA parameters	
Population	15000
Generations	50
Elitism	10
Selection	Tournament
Crossover	Double Point ($P = 0.9$)
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]

Trajectory Result		
	Real	Result
Earth Departure		
Date T_0	05-09-1977 2443392.5	04-09-1977 2443391
Departure Δv [m/s]	10330.2	9413.34
Jupiter Flyby		
Date T_1	05-03-1979 2443937.5	17-10-1979 2444163.5
$v_{\infty-in}$ [km/s]	10964.4	6558.52
$v_{\infty-out}$ [km/s]	10960.7	6559.38
δ [deg]	98.4863	93.0884
r_p [m]	3.376×10^8	1.1118×10^9
Δv [m/s]	1.1371	0.34356
Saturn Arrival		
Date T_2	12-11-1980 2444555.5	24-04-1982 2445084.125
Arrival V_{in} [m/s]	20115.7	12704.8
Result		
Total cost [Δv]	**10331.5**	**9414.05**

Table 5 Trajectory optimization for Voyager-I; inputs and solution.

Trajectory Result				
	Real	minimum Δv ($\alpha = 1$)	Δv and Time Tradeoff ($\alpha = 0.7$)	Pykep+Pagmo
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
δ[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
Δv[m/s]	57.844	0.34356	1.01265	0.40
Saturn Flyby				
	26 – 08 – 1981	24 – 04 – 1982	18 – 03 – 1981	27 – 11 – 1984
Date T_2	2444842.5	2445084.125	2444680	2446031.5
$v_{\infty - in}$[km/s]	10790.6	8254.31	13088.7	5082.83
$v_{\infty -}$ out [km/s]	9052.45	8255.15	13090.4	5541.03
δ[deg]	85.656	83.3468	85.269	83.46
r_p[m]	2.18×10^8	2.805×10^8	1.05455×10^8	6.20×10^8
Δv[m/s]	815.66	0.378272	0.758372	198.385
Uranus Flyby				
	24 – 08 – 1986	26 – 05 – 1987	19 – 02 – 1985	02 – 10 – 1991
Date T_3	2446666.5	2446942.125	2446120	2448531.5
$v_{\infty -}$ in [km/s]	12877.1	11900.2	17378.6	6641.43
v_{∞}-out [km/s]	17661.6	11900.2	17379.4	7303.52
δ[deg]	22.164	18.1143	26.0872	5.82
r_p[m]	9.21×10^7	2.583×10^8	7.76536×10^7	2.39×10^9
Δv[m/s]	3728.92	0.025	0.654346	626.252
Neptune Arrival				
	25 – 08 – 1989	15 – 09 – 1991	25 – 04 – 1989	06 – 08 – 1998
Date T_4	2447763.5	2448484.125	2447277	2451031.5
Arrival V_{in}[m/s]	21551.0	15349.0	21862.2	8277.72
Result				
Total cost [Δv]	**14830.12**	**9414.075**	**9856.34**	**10819.9**

Table 6 Voyager-II results comparison. Extended version of Tab. 4.

[5] Englander, J., Conway, B., and Williams, T., "Optimal Autonomous Mission Planning via Evolutionary Algorithms," 21st AAS/AIAA Space Flight Mechanics Meeting, American Astronomical Soc." 2011, pp. 833–852. URL https://api.semanticscholar.org/CorpusID:67110722.

[6] J. Englander and B. Conway and T. Williams, "Automated Mission Planning via Evolutionary Algorithms," *Journal of Guidance, Control, and Dynamic*, Vol. 35, No. 6, 2012, pp. 1878–1887. https://doi.org/\textcolor{red}{10.2514/1.54101}.

[7] Zhu, K., Li, J., and Baoyin, H., "Multi-swingby optimization of mission to Saturn using global optimization algorithms," *Acta Mechanica Sinica*, Vol. 25, No. 6, 2009, pp. 839–845. https://doi.org/10.1007/s10409-009-0299-6.

[8] Ceriotti, M., "Global optimisation of multiple gravity assist trajectories," 2010.

[9] Standish, M., and Williams, J., *Orbital Ephemerides of the Sun, Moon, and Planets*, 2006.

[10] Folkner, W. M., Williams, J. G., Boggs, D. H., Park, R. S., and Kuchynka, P., "The Planetary and Lunar Ephemerides DE430 and DE431," *Interplanetary Network Progress Report*, Vol. 42-196, 2014, pp. 1–81. URL https://ipnpr.jpl.nasa.gov/progress_report/42-196/196C.pdf.

[11] NASA, "Approximate Positions of the Planets," Tech. rep., Jet Propulsion Lab, 1964. URL https://ssd.jpl.nasa.gov/planets/approx_pos.html.

[12] Jordon, J. F., "The Application of Lambert's Theorem to the Solution of Interplanetary Transfer Problems," Tech. Rep. 32-521, NASA - Jet Propulsion Lab, 1964. URL http://www.gravityassist.com/IAF1/Ref.%201-77.pdf.

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

[14] Izzo, D., "Revisiting Lambert's Problem," *Celestial Mechanics and Dynamical Astronomy*, 2014. https://doi.org/10.1007/s10569-014-9587-y.

[15] Bond, V. R., *Matched-conic solutions to round-trip interplanetary trajectory problems that insure state-vector continuity at all boundaries*, Vol. 4342, National Aeronautics and Space Administration, 1969. URL https://ntrs.nasa.gov/api/citations/19690005529/downloads/19690005529.pdf.

[16] Wagner, S., Kaplinger, B., and Wie, B., "GPU Accelerated Genetic Algorithm for Multiple Gravity-Assist and Impulsive V Maneuvers," 2012. https://doi.org/10.2514/6.2012-4592.

[17] O'Neil, W., Ausman, N., Gleason, J., Landano, M., Marr, J., Mitchell, R., Reichert, R., and Smith, M., "Project Galileo at Jupiter," *Acta Astronautica*, Vol. 40, No. 2, 1997, pp. 477–509. https://doi.org/https://doi.org/10.1016/S0094-5765(97)00114-8, URL https://www.sciencedirect.com/science/article/pii/S0094576597001148, enlarging The Scope of Space Applications.

[18] Izzo, D., "PyGMO and PyKEP: Open Source Tools for Massively Parallel Optimization in Astrodynamics (the case of interplanetary trajectory optimization)," 2012. URL https://www.esa.int/gsp/ACT/doc/MAD/pub/ACT-RPR-MAD-2012-(ICATT)PyKEP-PaGMO-SOCIS.pdf.

[19] Biscani, F., Izzo, D., and Yam, C. H., "A Global Optimisation Toolbox for Massively Parallel Engineering Optimisation," , 2010. https://doi.org/10.48550/arXiv.1004.3824.

[20] Vasile, M., Minisci, E., and Locatelli, M., *On Testing Global Optimization Algorithms for Space Trajectory Design - AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 2008. https://doi.org/10.2514/6.2008-6277, URL https://arc.aiaa.org/doi/abs/10.2514/6.2008-6277.

[21] Gage, P. J., Braun, R. D., and Kroo, I. M., "Interplanetary trajectory optimization using a genetic algorithm," *Journal of the Astronautical Sciences*, Vol. 43, No. 1, 1995, pp. 59–75. https://doi.org/10.2514/6.1994-3773.

[22] Yam, C. H., Lorenzo, and Izzo, D., "Low-Thrust Trajectory Design as a Constrained Global Optimization Problem," *PROCEEDINGS OF THE INSTITUTION OF MECHANICAL ENGINEERS. PART G, JOURNAL OF AEROSPACE ENGINEERING*, Vol. 225, 2011, pp. 1243–1251. https://doi.org/10.1177/0954410011401686.