Search for a Grand Tour of the Jupiter Galilean Moons

Dario Izzo
European Space Agency
dario.izzo@esa.int

Luís F. Simões
VU University Amsterdam
luis.simoes@vu.nl

Marcus Märtens
European Space Agency
marcus.maertens@esa.int

Guido C. H. E. de Croon
TU Delft
g.c.h.e.decroon@tudelft.nl

Aurelie Heritier
European Space Agency
relieheritier@gmail.com

Chit Hong Yam
Hong Kong University of Science and Technology
chithongyam@gmail.com

ABSTRACT

We make use of self-adaptation in a Differential Evolution algorithm and of the asynchronous island model to design a complex interplanetary trajectory touring the Galilean Jupiter moons (Io, Europa, Ganymede and Callisto) using the multiple gravity assist technique. Such a problem was recently the subject of an international competition organized by the Jet Propulsion Laboratory (NASA) and won by a trajectory designed by aerospace experts and reaching the final score of 311/324. We apply our method to the very same problem finding new surprising designs and orbital strategies and a score of up to 316/324.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Graph and tree search strategies, Heuristic methods*

General Terms

Algorithms

Keywords

Self-adaptation, Differential Evolution, multi-criteria tree-search, interplanetary trajectory optimization

1. INTRODUCTION

The design of interplanetary trajectories can be profitably approached as a global optimization problem. In the last decade, a number of researchers [1, 9, 5, 14, 13, 10] have successfully applied different automated search techniques to these type of problems, helping to define a new approach to interplanetary mission design. In this context, the international Global Trajectory Optimization Competition (GTOC) series[1], was born with the objective of fostering research in this area by letting different methods compete on one, difficult, well-defined, problem. During the competition, standard design methods based on a profound domain knowledge and on human-in-the-loop procedures, compete with more or less automated design techniques. Interestingly, evolutionary search techniques have been used by many of the competition participants, but have never obtained results that are competitive to those found by other types of optimization procedures. For an example of the complex and diverse solution strategies employed in the first edition of the competition one can refer to [7]. In this paper we describe an automated search procedure based on evolutionary techniques that is able to design multiple fly-by interplanetary trajectories and apply it to the latest problem released in the GTOC series, the sixth edition problem.

[1]See `http://sophia.estec.esa.int/gtoc_portal/`

1.1 Brief problem description

The sixth edition of the GTOC series was organized by the Jet Propulsion Laboratory (JPL), and the problem was released on the 10th September 2012. The 33 international teams who registered to the event were given one month time to find a solution to arguably one of the most difficult problems ever formalized in the context of these competitions: mapping globally the four Galilean Jupiter moons (Io, Ganymede, Europa and Callisto) using multiple fly-bys and an electric propulsion spacecraft. The formal mathematical description (problem statement) can be found in [12]. Here we repeat some of the details to facilitate the reader. The inertial Jupiter centric frame $\mathcal{F}_i$ is indicated with $\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}$. On each moon, a body frame $\mathcal{F}_b$ is defined and indicated with $\hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2, \hat{\mathbf{b}}_3$. In such a body frame, the moon surface is considered to be divided into 32 faces $F_i, i = 1..32$, similarly to the division used to stitch soccer balls (see Figure 1), defined by their cartesian coordinates in $\mathcal{F}_b$. We write $F_i := \{\hat{\mathbf{p}}_j, j = 1..n_i\}$ where n_i is the number of vertices of the i-th face (either 5 or 6), and $\hat{\mathbf{p}}_j$ are the j-th vertex coordinates in $\mathcal{F}_b$. Each face, according to its scientific interest, is assigned a score ranging from 1 to 3 points. Each face on Europa is assigned double scores because of the higher scientific interest of this particular Jupiter moon. A face F_i on a moon is visited, and thus its score added to the total trajectory score, when a fly-by around that moon is performed having its closest approach vector $\mathbf{r}_p$ passing through that particular face[2]. At each successive fly-by over

[2]More formally, this happens when $\mathbf{r}_p$ lies within the pyramid having the face F_i as a basis and the moon center as a vertex.

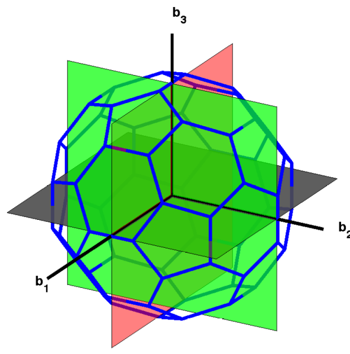

Figure 1: Visualization of the moon surface division into polygonal faces. Reproduced from the original problem description from JPL [12].

the same face, no further score is accounted. This scoring mechanism was used by the organizers in an attempt to represent numerically as an objective function the scientific value of imaging-experiments carried out on-board the spacecraft during each of its closest passages over the moons. This creates a trajectory design problem with a maximum score of 324 points, which corresponds to all faces of the four Galilean moons being visited. The spacecraft has a starting mass $M_0 = 2000$ kg of which half can be used as propellant for the spacecrafts electric engine, able to deliver continuously a maximum thrust $\tau_{max} = 0.1$ N. At each of its close Jupiter passages the available propellant mass decreases by a penalty value to account for added shielding necessary to protect the electronics from the strong Jupiter magnetic field. The functional form of this penalty is given in the JPL problem description [12]. Applying our technique, we find many different trajectories scoring up to 316/324 points and employing very different strategies from the one designed by the winners of this international competition.

The sixth edition of the GTOC series was won by a joint team from the University of Rome and Turin Polytechnic, who designed their winning trajectory scoring 311 out of the maximum 324 points. Their interplanetary trajectory made a clever use of orbital mechanics to first map Callisto partially, then all Ganymede, then all of Europa and to finish with a full mapping of Io.

The paper is organized as follows: in Section 2 and in Section 3 we introduce two global optimization problem ($\mathcal{P}_1$ and $\mathcal{P}_2$) which form the building blocks of our proposed method. In both cases, solutions represent interplanetary multiple gravity assist trajectories in the Jupiter system. In Section 4 we describe and comment upon the evolutionary strategy that we use for solving generic instances of both problems. We then show, in Section 5, how to build incrementally a tree $\mathcal{T}$ of solutions to the GTOC6 problem juxtaposing multiple $\mathcal{P}_1$ and $\mathcal{P}_2$ solutions. In the same section we discuss the tree search strategy that is employed to find the best possible tour. In Section 6 we report on the experimental set-up used and the results produced. Finally, in Section 7, we describe the best found trajectory that improves the previously known best.

2. PROBLEM $\mathcal{P}_1$: THE JUPITER CAPTURE

In this section the first part of the trajectory, i.e. the Jupiter capture, is described as a global optimization problem and is indicated with $\mathcal{P}_1(\mathbf{m})$. Formally, we define problem $\mathcal{P}_1$ as a global optimization problem:

$$\mathcal{P}_1(\mathbf{m}): \quad \begin{array}{rl} \text{find:} & \mathbf{x} \in R^{16} \\ \text{to minimize:} & \Delta V(\mathbf{x},\mathbf{m}) = \sum_{i=0}^{3} \Delta V_i \\ \text{subject to:} & \mathbf{lb} \le \mathbf{x} \le \mathbf{ub} \\ & \Delta V_i < cT_i a_{max} \end{array}$$

Consider the set $\mathcal{M} := \{i, e, g, c\}$ containing the four Galilean moons i=Io, e=Europa, g=Ganymede, c=Callisto. Given a sequence $\mathbf{m} \in \mathcal{M}^4$ we want to find a vector $\mathbf{x}$ encoding an interplanetary trajectory that visits, in sequence, the moons $\mathbf{m}$ making use of maximum one deep space maneuver between two successive moons and one at the very beginning (at 1000 Jupiter Radii (JR) where the spacecraft is prescribed to start its journey [12]). The cumulative strength of the four allowed deep space maneuvers $\Delta V = \sum_{i=0}^{3} \Delta V_i$ is minimized and a maximum acceleration constraint on each ΔV_i is considered to account for the maximum acceleration a_{max} defined as τ_{max}/M, where M is the spacecraft mass. A detailed description of the decision vector $\mathbf{x}$ (here also called trajectory encoding), of the objective function $\Delta V(\mathbf{x},\mathbf{m})$ and of the problem constraints follows.

2.1 Trajectory encoding

The capture trajectory is encoded into $\mathbf{x}$ using a modification of the MGA-1DSM encoding described in [8]. The modification is necessary to account for the fact that the spacecraft starts its journey from a point at $R_s = 1000$ Jupiter Radii (JR) from Jupiter center and not from a planet as assumed in the original MGA-1DSM encoding. Using the concatenation operator "$+$" to highlight the four separate trajectory legs[3], a capture trajectory is encoded as follows: $\mathbf{x} = [t_0, u, v, T_0] + [\beta_1, h_1, \eta_1, T_1] + [\beta_2, h_2, \eta_2, T_2] + [\beta_3, h_3, \eta_3, T_3]$, where all symbols are explained below.

2.1.1 1st leg

The launch date is encoded into t_0 as a continuous variable using the Modified Julian Date 2000 (MJD2000), defined as the number of days passed since 2000-01-01 at 00:00:00 hours. The starting position vector $\mathbf{r}_0$ of the spacecraft is encoded by the next two variables u, v as follow:

$$\mathbf{r}_0 = 1000R_s(\cos\theta\cos\phi\hat{\mathbf{i}} + \sin\theta\cos\phi\hat{\mathbf{j}} + \sin\phi\hat{\mathbf{k}})$$

where R_s is the Jupiter average radius and

$$\theta = 2\pi u, \phi = \cos^{-1}(2v-1) - \pi/2$$

The use of the variables, u and v, is motivated by the need to have the position vector $\mathbf{r}_0$ uniformly distributed over the starting sphere of radius R_s when the encoding variables are sampled at random. The distribution would not be uniform if we were using directly θ and ϕ. Finally the total duration of the first leg is encoded in T_0.

2.1.2 Following legs

For each of the following three legs, the starting velocity $\mathbf{v}_{out}$ (outgoing spacecraft velocity at the moon m in $\mathcal{F}_i$), is encoded by the first two variables β, h as follows:

$$\begin{array}{r} \mathbf{v}_{out} = \mathbf{v}_m + \\ |\mathbf{v}_{in}|(\cos(\delta)\hat{\mathbf{e}}_1 + \cos(\beta)\sin(\delta)\hat{\mathbf{e}}_2 + \sin(\beta)\sin(\delta)\hat{\mathbf{e}}_3) \end{array} \quad (1)$$

[3]A trajectory leg is a part of the whole trajectory between two consecutive moons.

Table 1: Lower and upper bounds, lb and ub, for the generic instance of $\mathcal{P}_1$.

	Lower	Upper
t_0 [MJD 2000]	7305	11323
u	0	1
v	0	1
T_0 [days]	190	210
β_i	-2π	2π
h_i [km]	50	2000
η_i	-2π	2π
T_1 [days]	0.1	5
T_2 [days]	5	100
T_3 [days]	20	55

where $\mathbf{v}_m$ is the velocity of the moon m at the beginning of the leg[4], $\mathbf{v}_{in}$ is the spacecraft velocity at the end of the previous leg (incoming velocity at the moon) and:

$$\begin{array}{l} \tilde{\mathbf{v}}_{in} = \mathbf{v}_{in} - \mathbf{v}_m \\ e = 1 + ((h + R_m)/\mu_m)\,\tilde{v}_{in}^2 \\ \delta = 2\sin^{-1}(1/e) \\ \hat{\mathbf{e}}_1 = \tilde{\mathbf{v}}_{in}/|\tilde{\mathbf{v}}_{in}| \\ \hat{\mathbf{e}}_2 = (\hat{\mathbf{e}}_1 \times \mathbf{v}_m)/|\hat{\mathbf{e}}_1 \times \mathbf{v}_m| \\ \hat{\mathbf{e}}_3 = \hat{\mathbf{e}}_1 \times \hat{\mathbf{e}}_2 \end{array} \tag{2}$$

where R_m is the moon radius and μ_m its non dimensional gravity parameter. Next, the variable η_i encodes the position of a deep space maneuver which occurs at $t = t_0 + \sum_{j=0}^{i-1} T_j + \eta_i T_i$ and T_i encodes the total leg duration.

2.2 Objective function

The objective function for problem $\mathcal{P}_1$ is computed as follows. A single revolution Lambert's problem [2] is solved between $\mathbf{r}_0$ and $\mathbf{r}_1$ with transfer time T_0. $\mathbf{r}_1$ is the position of the first encountered moon at $t_0 + T_1$. The solution to the Lambert's problem returns $\mathbf{v}_0$ and $\mathbf{v}_{in}^1$, i.e. the velocities at the beginning and at the end of the first trajectory leg. From these results, we compute $\Delta V_0 = abs(|\overline{\mathbf{v}}_0| - |\mathbf{v}_0|)$, where $\overline{\mathbf{v}}_0$ is the magnitude of the prescribed starting velocity (i.e. 3.5 [km/s]). For each successive leg we then compute the velocity increments ΔV_i corresponding to the deep-space maneuver location encoded in $\mathbf{x}$ using the standard procedure employed by the MGA-1DSM encoding [8]. The final objective function is the sum of all computed velocity increments, i.e., $\Delta V = \sum_{i=0}^{3} \Delta V_i$.

2.3 Constraints

The box bounds **lb** and **ub** used in problem $\mathcal{P}_1$ are reported in Table 1. The most important bounds are those on the transfer times T_i. They, alone, select the type of capture we are looking for. In our case, we want to be captured in the Jupiter system in the shortest possible time and we aim to acquire a short orbital period to increase the frequency we will be able to score points with. We achieve this goal by forcing two moons to be visited on the very first revolution in a rapid succession and we thus select a very small upper bound on T_1. The possibility to have these types of "multiple-satellite-aided Jupiter capture trajectories" is discussed thoroughly in [11]. We then force the following orbit to have a relatively small period with an upper bound of 100 days on T_2, thus allowing, in the second leg, a 13:1 resonance with the Ganymede orbit or a 5:1 with Callisto. The upper bound on T_3, is then selected to be lower, i.e. 55 days, to complete the spacecraft insertion in the Galilean moon system (allowing only 3:1 resonances with the outer moon Callisto[5]).

The non linear constraint on the spacecraft ΔV_i accounts for the maximum acceleration τ_{max} allowed by the on-board spacecraft electric propulsion system. A constant c, here set to be 0.1, is also introduced to account for orbital effects while biasing the search towards purely ballistic captures (corresponding to $\Delta V = 0$)[6].

[4]The ephemerides of the moons are detailed in the original JPL problem statement [12]

3. PROBLEM $\mathcal{P}_2$: SINGLE TRANSFER BETWEEN MOONS

A fundamental building block to our overall strategy is a second global optimization problem we here indicate as $\mathcal{P}_2$. This represents the problem of finding a minimum ΔV transfer between two moons m_s and m_{next} fixing the starting conditions. The transfer starts at t_0 from m_s with a known absolute velocity $\mathbf{v}_{in}$ and makes a fly-by over the F_i face of m_s.

$$\mathcal{P}_2(F_i, m_{next}): \quad \begin{array}{rl} \text{find:} & \mathbf{x} \in R^4 \\ \text{to minimize:} & \Delta V(\mathbf{x}) \\ \text{subject to:} & \mathbf{lb_i}(F_i) \le \mathbf{x} \le \mathbf{ub_i}(F_i) \end{array}$$

where $\mathbf{x} = [\beta, h, \eta, T]$ and, most importantly, the box bounds $\mathbf{lb}_i$ and $\mathbf{ub}_i$ depend on the face F_i to be visited. While any particular instance of the problem $\mathcal{P}_2$ depends on F_i, m_{next}, m_s, $\mathbf{v}_{in}$ and t_0, we write $\mathcal{P}_2(F_i, m_{next})$, neglecting those variables $(m_s, \mathbf{v}_{in}, t_0)$ that are fixed when a previous part of the trajectory exists (e.g. a solution to $\mathcal{P}_1$ or to $\mathcal{P}_2$)

The objective function $\Delta V(\mathbf{x})$ is computed by first computing $\mathbf{v}_{out}$ from $\mathbf{v}_{in}$ using Eq.(1-2), and then by propagating those initial conditions with Keplerian dynamics for the time ηT. Finally, a single revolution Lambert's problem [2] is solved to get to the target moon position in $t_0 + T$.

3.1 Face-targeting

When instantiating $\mathcal{P}_2(F_i, m_{next})$ we need to specify the lower and upper bounds on the four variables β, h, η, T. The choice on the bounds on η, i.e. the position of the deep space maneuver, is rather simple: $\eta \in]0, 1[$ is set to allow small correction maneuvers at any points of the trajectory leg. For the transfer time, we use $T \in [0.1, min(4T_m, 40)]$ (days), where T_m is the largest period between the starting and final moon orbit. The maximum of 40 (days) on the upper bound avoids to consider long multiple revolution transfers when Callisto is either the starting or the targeted moon ($4T_m$ would then be roughly 64 days). The bounds on the remaining two variables, β and h are, instead, decided to have the resulting solution visit the chosen moon face F_i. The procedure to find these bounds is what we call face-targeting and it is based on the following results:

[5]The orbital period of Callisto is 16.7 days, while the orbital period of Ganymede is 7.15 days.

[6]The coefficient c is often referred to gravity loss and, coarsely, accounts for the difference between having to accumulate a given velocity difference in a flat space or in a central gravity field.

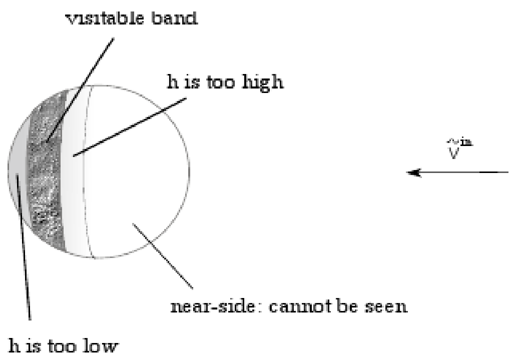

Figure 2: Visualization of the visitable band.

Definition 1. A point $\hat{\mathbf{p}}$ on the moon surface of radius R_m is visitable during a fly-by with incoming relative velocity $\tilde{\mathbf{v}}_{in}$ if and only if $\tilde{\mathbf{v}}_{in} \cdot \hat{\mathbf{p}} \geq 0$ and $h \in [h_{min}, h_{max}]$, where:

$$\begin{array}{l} h + R_m = \mu_m/(\tilde{\mathbf{v}}_{out} \cdot \tilde{\mathbf{v}}_{out})(1/\sin(\delta/2) - 1) \\ \tilde{\mathbf{v}}_{out} = \tilde{\mathbf{v}}_{in} - 2(\tilde{\mathbf{v}}_{in} \cdot \hat{\mathbf{p}})\hat{\mathbf{p}} \\ \delta = \text{acos}\left(\frac{\tilde{\mathbf{v}}_{out} \cdot \tilde{\mathbf{v}}_{in}}{\tilde{\mathbf{v}}_{out} \cdot \tilde{\mathbf{v}}_{out}}\right) \end{array}$$

and h_{min}, h_{max} were prescribed by the JPL problem statement [12] as the minimum and maximum fly-by altitude on each moon. If $\tilde{\mathbf{v}}_{in} \cdot \hat{\mathbf{p}} < 0$ the vertex is said to be in the near side of the moon. For the cases in which $\tilde{\mathbf{v}}_{in} \cdot \hat{\mathbf{p}} \geq 0$, if $h < h_{min}$ the vertex is said to be above the visitable band, if $h > h_{max}$ it is said to be below.

If a point is visitable according to the above definition, then a moon-centric hyperbola exists and has its closest approach radius $\mathbf{r}_p$ aligned with $\hat{\mathbf{p}}$. Essentially, at each fly-by the moon surface can be represented as in Figure 2, where the area made by all visitable points is shown to form a band (the visitable band) that is symmetric with respect to the incoming relative velocity and that is placed in the far side of the moon.

Definition 2. A face $F := \{\hat{\mathbf{p}}_j, j = 1..n\}$ defined by its n vertices is visitable if and only if $a_j \geq 0, j = 1..n$ exist such that the point $\sum_{j=1}^{n} a_j \hat{\mathbf{p}}_j$ is visitable

In other words, a face is visitable if at least one of its points is visitable. In order to compute the set L containing all visitable faces, one can use the following algorithm:

Algorithm 1 Find visitable faces

```
$L := \emptyset$
for all $F_i$ in the moon $m$ do
    for all $\hat{\mathbf{p}}_j \in F_i$ do
        check if $\hat{\mathbf{p}}_j$ is in the near side or above, below or in the visitable band.
    end for
    if (at least one vertex is visitable) or (at least one, but not all vertices are above the band) then set $L = L \cup \{F_i\}$
    end if
end for
return $L$
```

Given a face $F_i \in L$, we may now find the lower and upper bounds β_m, β_M, h_m, h_M that bracket such a face and bias the solution of the problem $\mathcal{P}_2$ to fly over F_i and thus to score that face (if not scored already). We do so by using algorithm 2.

Algorithm 2 Face bracketing

```
Let $F_i \in L$
for all $\hat{\mathbf{p}}_j \in F_i$ do
    If $\hat{\mathbf{p}}_j$ is on the near side, project it to the far side.
    Compute $h_j = \mu_m/(\tilde{\mathbf{v}}_{out} \cdot \tilde{\mathbf{v}}_{out})(1/\sin(\delta/2) - 1) - R_m$
    Compute $\beta_j = \text{atan2}(\tilde{\mathbf{v}}_{out} \cdot \hat{\mathbf{e}}_3, \tilde{\mathbf{v}}_{out} \cdot \hat{\mathbf{e}}_2)$
end for
set $\beta_m = \min_j \beta_j$, $\beta_M = \max_j \beta_j$
set $h_m = \max(\min h_j, h_{min})$, $h_M = \min(\max_j h_j, h_{max})$
```

4. SOLVING $\mathcal{P}_1$ AND $\mathcal{P}_2$

All problems that derive from different instances of $\mathcal{P}_1(\mathbf{m})$ and $\mathcal{P}_2(F_i, m_{next})$ are single objective, continuous global optimization problems that we solve by exploiting one powerful idea rooted in evolutionary computations: self-adaptation. By using self-adaptation we are able to not care about tuning algorithmic parameters, while still obtaining solutions for all different problems with an acceptable efficiency. This is a fundamental advantage as for each instance of $\mathcal{P}_1(\mathbf{m})$ or $\mathcal{P}_2(F_i, m_{next})$ we are, essentially, creating a much different objective function landscape. Resonant transfers, multiple revolution transfers and very short transfers may or may not exist according to the particular instance, drastically changing the fitness landscape.

In the case of $\mathcal{P}_1$ (a more difficult problem with a higher dimensionality with respect to $\mathcal{P}_2$ and a much larger search space mainly determined by the wide bounds on the t_0, the launch window), we also make use of a second tool: the asynchronous island model. This allows us to achieve a coarse-grained parallelization while introducing a migration operator that further accelerates evolution.

We tested three algorithms employing self-adaptation as a principle, a) differential evolution with self adaptation strategy (jDE) [3], the Corana's Adaptive Neighbourhood Simulated Annealing (SA-AN) [4], and Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [6]. We do not have space here to discuss the performances of these algorithms in connection with our problems, but we did perform extensive test sessions and our final decision converged on jDE as for its clear superiority in the context of $\mathcal{P}_1$, where a good interaction with the asynchronous generalized island model is necessary to actually find any good solutions, and for its low run time in general.

It is worth anticipating here that, as explained in detail in a later section, to assemble one entire solution of the Jupiter's moon tour, up to 1,000,000 different instances of $\mathcal{P}_2$ need to be solved and one good solution to $\mathcal{P}_1$ has to be determined. To give an idea of the performances of our chosen set up, we report, preliminarily, some results on the resulting computational efficiency[7]. Figure 3 illustrates the results of 50 runs on three different instances of $\mathcal{P}_1(\mathbf{m})$, which consistently resulted in good final results: a) $\mathbf{m} = [g, c, c, c]$, b) $\mathbf{m} = [c, g, g, g]$, c) $\mathbf{m} = [g, i, i, i]$. Each run is made up to convergence, which we check every 40 generations and is defined by two criteria: fitness variability and chromosomic

[7]Our experiments are done on a Intel(R) Xeon(R) CPU - X5355 @ 2.66GHz

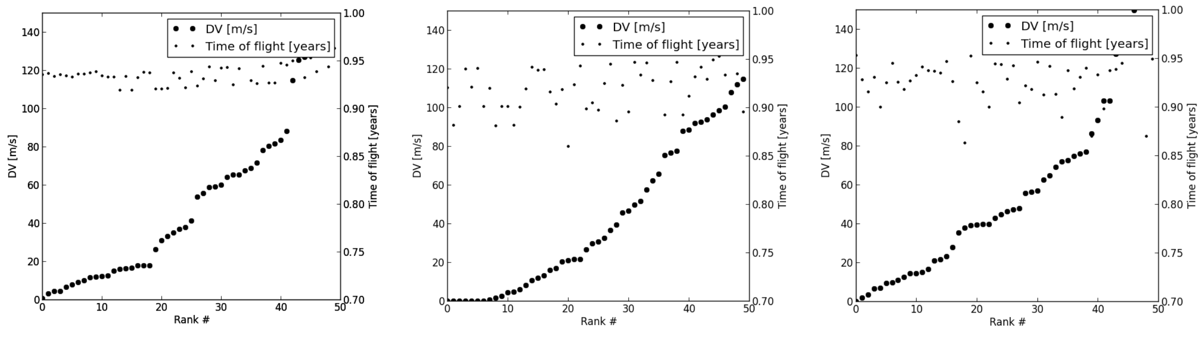

Figure 3: Performance of jDE in the asynchronous island model along 50 runs on different instances of problem $\mathcal{P}_1$. Left: $\mathbf{m} = [g, c, c, c]$. Center: $\mathbf{m} = [c, g, g, g]$. Right: $\mathbf{m} = [g, i, i, i]$. in the plots, the solutions are ranked with respect to their fitness.

variability $f_{tol} \leq 1e^{-2}$ and $x_{tol} \leq 1e^{-3}$. Eight islands are used and one individual is migrated along a ring topology each 100 generations. Each island evolves a population of 20 individuals. The asynchronous migration model implementation and the jDE implementation are made available via the open source PaGMO/PyGMO project[8]. In the case of $\mathcal{P}_2$, where one single island is used (i.e. no migration), we report that our set-up is able, on average, to solve 10 problem instances each second.

5. ASSEMBLING ONE MOON TOUR

Our overall strategy to solve the GTOC6 problem consists of assembling in cascade multiple solutions to problems $\mathcal{P}_1$ and $\mathcal{P}_2$. We start with a trajectory that captures our spacecraft in the Jupiter system (i.e. one solution to $\mathcal{P}_1(\mathbf{m})$), and we append to it, one by one, solutions to $\mathcal{P}_2(F_i, m_{next})$. As there are four moons and, on average, 15 visitable faces, the average branching factor is 60. This means that at the n^{th} step a full search tree contains 60^n trajectories. This creates a tree $\mathcal{T}$ of possibilities that soon becomes intractable.

In our notation, the generic tree node is defined by $\mathcal{N} := \{m, t, \mathbf{v}_{in}, \Delta V, \mathbf{f}\}$ and represents an interplanetary trajectory (solution to the GTOC6 problem), that at the epoch t visits the moon m with an incoming velocity $\mathbf{v}_{in}$, having cumulated a velocity change of ΔV and having previously visited all moon faces in $\mathbf{f}$. Each node can thus be assigned a score that is computed as a function of $\mathbf{f}$. The root node of the tree, i.e. a Jupiter capture trajectory, is determined using Algorithm 3.

Algorithm 3 Find a capture trajectory $\mathcal{N}_{root}$

```
while $\Delta V \geq 100$ (m/s) and $\Delta T = t - t_0 \geq 0.9$ (years) do
    pick a random $\mathbf{m} \in \mathcal{M}^4$
    solve $\mathcal{P}_1(\mathbf{m})$
end while
    let $m$ be the last moon in $\mathbf{m}$,
    let $t$ be the epoch at the last moon,
    let $\mathbf{v}_{in}$ be the spacecraft velocity at $t$,
    let $\Delta V$ be the cumulative velocity increment,
    let $\mathbf{f}$ the set of all moon faces visited,
    let $\mathcal{N}_{root} := \{m, t, \mathbf{v}_{in}, \Delta V, \mathbf{f}\}$
```

5.1 The lazy race tree search

As explained above, starting from a root node $\mathcal{N}_{root}$ that is an outcome of Algorithm 3, we define a tree $\mathcal{T}$ where each node $\mathcal{N}$ represents a solution (i.e. a trajectory) to our problem. The tree depth, defined as d, represents the number of fly-bys performed (after capture). The tree dimension make a complete search intractable. We thus need to deploy a different strategy. A first attempt could be to search the tree implementing a depth first search by adding pruning criteria to make sure to trim uninteresting branches and to complete the search in a reasonable time. The advantage would be to prioritize an early exploration of the tree leaves. The problem with this approach is that it is impossible to estimate how much of the tree would be pruned out by a certain criterium, and thus each search could easily be too short or too long. A breadth first search strategy, carrying over to the next depth a constant number of nodes that is selected by a ranking criteria, solves this problem having a time complexity that grows linearly with the tree depth, which is a characteristic that is highly desirable in our case. In such an algorithm, nodes that have equal depth (i.e. number of fly-bys) need to be ranked and we would then be forced to compare trajectories that possess different times of flight, cumulative ΔV and score using an aggregated objective function, which introduces a lot of greediness in the search[9]. We thus introduce a novel algorithm that we call Lazy Race Tree Search (LRTS) able to overcome the problems mentioned above. The basic idea, outlined in Algorithm 4, is to only rank (and eventually select and prune) nodes that have similar time of flight, not equal tree depth. Introducing the set $\mathcal{U}$ containing all unbranched nodes, initialized to a set containing only the root node $\mathcal{N}_0$, at each iteration we look in $\mathcal{U}$ for nodes having time of flight smaller than the minimum time of flight T_0 among them plus a small ϵ. We then select bf (branching factor) nodes to branch, using the ranking criteria, and we remove all selected nodes from $\mathcal{U}$. The algorithm stops when T_0 is larger than the 4 years allowed by the problem definition.

6. TREE SEARCH RESULTS

In order to assess the performance of LRTS on our problem, while also trying to beat the best solution known for

[8]http://pagmo.sourceforge.net/pygmo/index.html

[9]In a preliminary implementation of this algorithm we could not reach the score threshold of 300

Algorithm 4 Lazy Race Tree Search

```
$\mathcal{U} := \{\mathcal{N}_{root}\}$ (the set of unbranched nodes)
let $t_0$ be the starting epoch of the trajectory
while $T_0 \leq 4$ years do
    let $T_0 := \min_{\mathcal{U}}(t - t_0)$,
    let $\mathcal{S} := \{\mathcal{N} \in \mathcal{U} | (t - t_0) \in [T_0, T_0 + \epsilon]\}$,
    select $bf$ nodes in $\mathcal{S}$ and branch them,
    remove from $\mathcal{U}$ all nodes in $\mathcal{S}$
end while
```

the GTOC6 problem (i.e. the trajectory scoring 311 found by the group at Polytechnic of Turin and La Sapienza, University of Rome), we launch a large number (360) of tree searches. The LRTS uses a branching factor $bf = 50$, a time of flight bin of $\epsilon = 5$ days and a ranking method defined by the trajectory score augmented by the maximum potential number of points scored on the current moon m (with ties broken by a secondary ranking method defined as the number of non scoring fly-bys). A gravity loss coefficient of $c = 0.1$ is used when instantiating the various $\mathcal{P}_2$. Each search starts from a different root node $\mathcal{N}_{root}$ found by running once Algorithm 3. The results are summarized in Figure 4. Most trajectories found are above the threshold of 300 points, some (34 to be exact) have a higher score than the currently known best of 311. We here note that the overall trajectory ΔV is consistently very small thanks to the pruning at 100 m/s in Algorithm 3 and to the very small gravity loss coefficient c selected. As a consequence, all solutions found by LRTS can be transformed into low-thrust trajectories (i.e. trajectories that can be flown by an electric propulsion system such as that prescribed by the Jet Propulsion Laboratory problem statement). On average, one tree search is completed in 16.8 hours. The average depth of the searched tree is 132. Using 32 CPUs, the whole run lasts 7.87 days. During the entire run an estimated number of half a billion (500,000,000) different global optimization problems (of type Multiple Gravity Assist with one Deep Space Maneuver MGA-1DSM) are solved by the self-adaptive jDE deployed in our asynchronous island model. It is interesting to note how only the sequences $\mathbf{m} = [c, g, g, g]$, $\mathbf{m} = [g, i, i, e]$, $\mathbf{m} = [g, i, i, g]$, $\mathbf{m} = [g, i, i, i]$, $\mathbf{m} = [c, g, c, c]$, $\mathbf{m} = [c, i, c, c]$, $\mathbf{m} = [g, c, c, c]$ were found to be able to define a suitable capture trajectory and, in any case, all best trajectories used the Callisto, Ganymede, Ganymede, Ganymede sequence, indicating that such an entry strategy results in particularly good root nodes for LRTS.

7. OUR SOLUTION

In this section, we describe in more details the solution with the highest score found by our tree search (i.e. the left most point in Figure 4). It has a total score of 316 out of the maximum score of 324, visiting 120 moon faces against the 115 faces visited by the previously best known solution (winner of the 6th edition of the GTOC which scores 311). From Table 2, we note that our solution almost maps all faces on three of the four moons: full scores on Europa and Ganymede and one point less on Io. The solution was sent according to the required competition format to the Jet Propulsion Laboratory experts (the GTOC6 organizers) who kindly validated the new solution using their trajectory analysis tools.

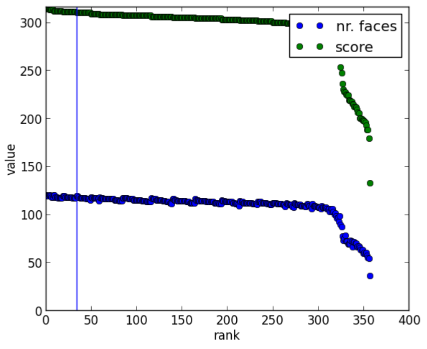

Figure 4: Results after 360 tree searches. The vertical line corresponds to the previous best known solution. The best score (most left point) is 316.

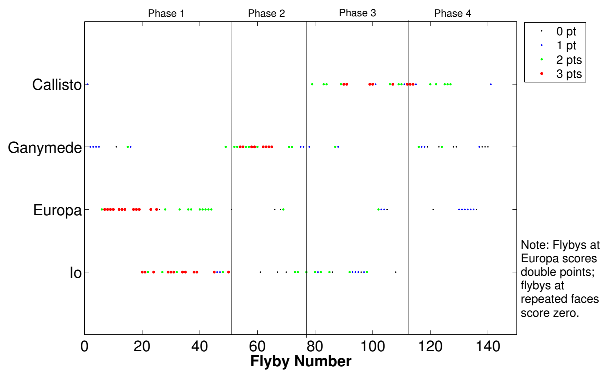

Figure 5: Scores obtained at each flyby of the four moons.

The scores on each flyby of the moons are plotted in Fig. 5, in which different colors indicates the points received: 3, 2, 1, or 0 for flyby on faces visited before, while Europa receives double points. From the plot, we notice that the flyby sequence and scores obtained are quite well structured instead of scattered evenly. The flybys on the same moon and scores are clustered together. They are also mostly ordered from first visiting the high score faces then the low score faces. From the dynamical point of view, visiting the same moon repeatedly implies the use of a resonance transfer (i.e. integer ratio of the orbital periods of the spacecraft and the moon) and is rather easy to obtain, while switching between moons may result in a time gain, but it is subject to stricter planetary phasing consideration and may thus cost more ΔV or just be not possible. Our solution clearly makes use of both possibilities trying to take advantage of quick moon

Table 2: Summary of our Jovian tour.

Moon	No. of Flybys	Scores Obtained/Max. Score
Io	37	67/68
Europa	39	136/136
Ganymede	40	60/60
Callisto	25	53/60
Total	141	316/324

switches as much as possible and otherwise mapping the same moon on resonances. This overall strategy, which we call moon-hopping, is very different from that employed by the GTOC6 winning trajectory that, instead, mapped one moon at a time.

From Fig. 5 we observe how the whole trajectory can be divided into four phases. In Phase 1, starting from a fast Callisto-Ganymede flyby using a 2.3 days long transfer, the spacecraft is captured in the Jovian system. A few more flybys at Ganymede are then able to further reduce the orbital period, allowing a visit to the high-score faces of the inner moons, Io and Europa. Phase 2 consists of mostly high-score flybys at Ganymede, mixed with flybys at the mid-score faces of Io. In the 3rd phase, the spacecraft visits Callisto and Io on faces with high and low scores, respectively. The remaining low-scores faces in Europa, Ganymede, and Callisto are visited in the final phase. The trajectories of the 4 phases are plotted in Figure 6.

Solutions from the tree searches are under an impulsive MGA-1DSM model, which is a good approximation of the low-thrust propulsion model. In the MGA-1DSM model, our solution has a total ΔV of 83 m/s, which is mostly used in the first leg for the capture and that implies the rest of the trajectory is basically ballistic. We optimized this solution for final mass in a continuous low-thrust model [15], as required by the problem. As the trajectories coming out from the search are mainly ballistic (i.e. very little ΔV is used) we did not encounter any problem in such a low-thrust conversion. The final mass of the trajectory is 1036 kg and the total time of flight is 3.9 years. The final mass is mainly accounting for the mass penalty that needs to be assigned at each successive Jupiter close encounter as described in the JPL problem statement [12]. We did not check such a penalty during the tree search, only at the end. As a consequence some of the trajectories (roughly one out of ten) in Figure 4 are actually infeasible for the GTOC6 problem (luckily not the highest ranked one).

8. CONCLUSIONS

The design of extremely complex interplanetary trajectories including dozens of multiple gravity assists may be performed autonomously by computers with the aid of carefully planned searches powered by evolutionary algorithms. We consider the case of the GTOC6 challenge proposed by the Jet Propulsion Laboratory and improve significantly over the best known solution (winner of the competition) using a new completely automated scheme that makes extensive use of self-adaptation and of the asynchronous island model. Compared to the previous best solution the trajectory found by our computers appears to be using a quite different orbital strategy, frequently hopping between moons, which is of interest to interplanetary trajectory designers.

9. ACKNOWLEDGMENTS

We thank Dr. Anastassios Petropoulos from Jet Propulsion Laboratory (the organizer of GTOC6) for having numerically validated our trajectory to be a solution to the GTOC6 problem.

10. REFERENCES

[1] B. Addis, A. Cassioli, M. Locatelli, and F. Schoen. A global optimization method for the design of space trajectories. *Computational Optimization and Applications*, 48(3):635–652, 2011.

[2] R. H. Battin. *An introduction to the mathematics and methods of astrodynamics.* American Institute of Aeronautics and Astronautics, revised edition, 1999.

[3] J. Brest, V. Zumer, and M. S. Maucec. Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In *IEEE Congress on Evolutionary Computation, CEC 2006*, pages 215–222. IEEE, 2006.

[4] A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm Corrigenda for this article is available here. *ACM Transactions on Mathematical Software (TOMS)*, 13(3):262–280, 1987.

[5] K. Deb, N. Padhye, and G. Neema. Interplanetary trajectory optimization with swing-bys using evolutionary multi-objective optimization. In L. Kang, Y. Liu, and S. Zeng, editors, *Advances in Computation and Intelligence*, volume 4683 of *Lecture Notes in Computer Science*, pages 26–35. Springer, 2007.

[6] N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

[7] D. Izzo. 1st ACT global trajectory optimisation competition: Problem description and summary of the results. *Acta Astronautica*, 61(9):731–734, 2007.

[8] D. Izzo. Global Optimization and Space Pruning for Spacecraft Trajectory Design. In B. Conway, editor, *Spacecraft Trajectory Optimization*, Cambridge Aerospace Series, chapter 7, pages 178–201. Cambridge University Press, 2010.

[9] D. Izzo, V. M. Becerra, D. R. Myatt, S. J. Nasuto, and J. M. Bishop. Search space pruning and global optimisation of multiple gravity assist spacecraft trajectories. *Journal of Global Optimization*, 38(2):283–296, 2007.

[10] X. Liu, H. Baoyin, and X. Ma. Five special types of orbits around Mars. *Journal of guidance, control, and dynamics*, 33(4):1294–1301, 2010.

[11] A. E. Lynam, K. W. Kloster, and J. M. Longuski. Multiple-satellite-aided capture trajectories at Jupiter using the Laplace resonance. *Celestial Mechanics and Dynamical Astronomy*, 109(1):59–84, 2011.

[12] A. E. Petropoulos. Problem description for the 6th global trajectory optimisation competition. `http://goo.gl/FmrOc`. Accessed: 01/01/2013.

[13] M. Rosa Sentinella and L. Casalino. Cooperative evolutionary algorithm for space trajectory optimization. *Celestial Mechanics and Dynamical Astronomy*, 105(1–3):211–227, 2009.

[14] M. Vasile and M. Locatelli. A hybrid multiagent approach for global trajectory optimization. *Journal of Global Optimization*, 44(4):461–479, 2009.

[15] C. H. Yam, D. Izzo, and F. Biscani. Towards a high fidelity direct transcription method for optimisation of low-thrust trajectories. In *4th International Conference on Astrodynamics Tools and Techniques*, May 2010.

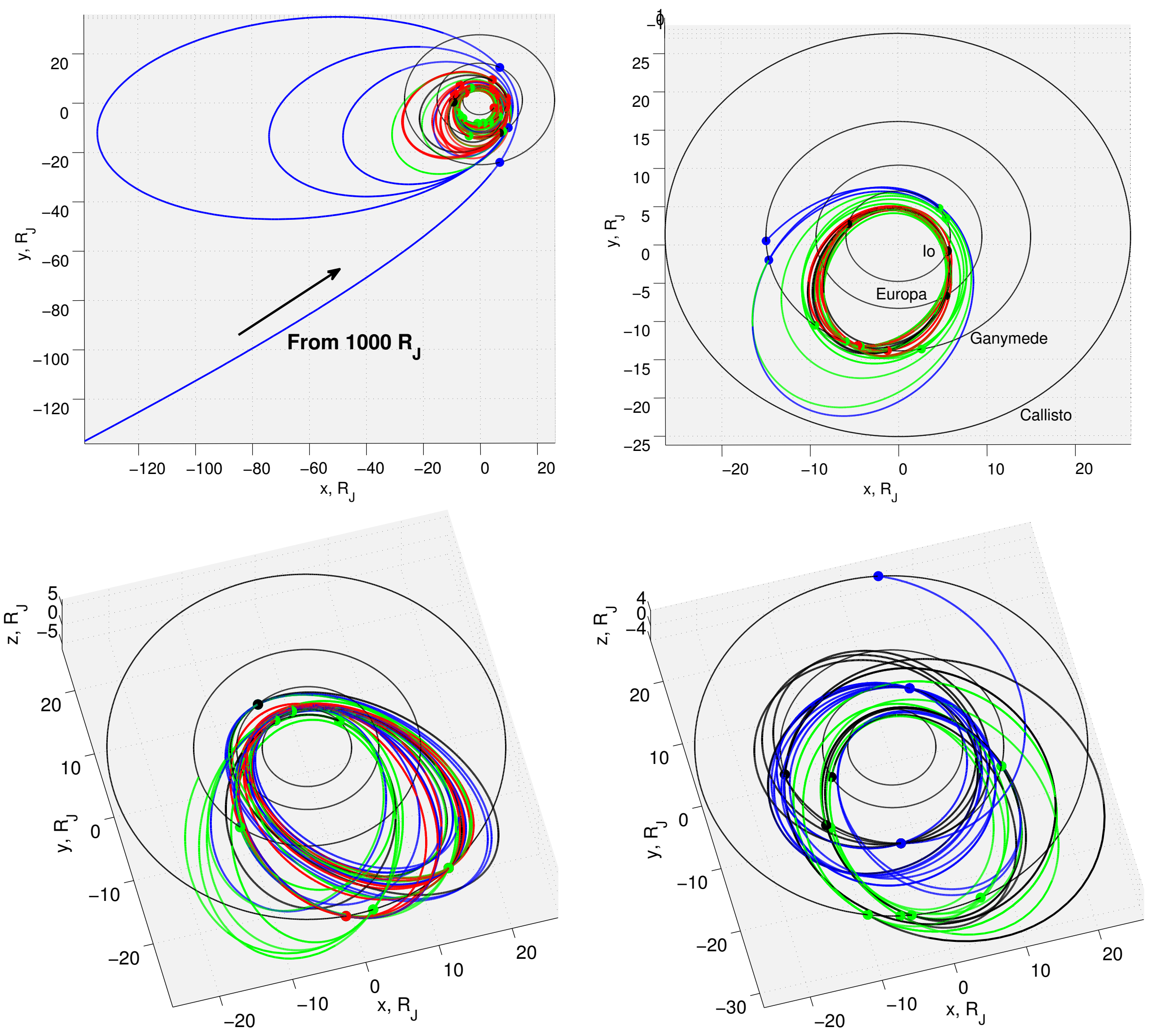

Figure 6: Visualization of the four phases of the moon tour. Phase 1: leg 1-51 (top-left); Phase 2: leg 52-78 (top-right); Phase 3: leg 79-115 (bottom-left); Phase 4: leg 116-141 (bottom-right). Colors on the trajectory represent scores obtained on the upcoming flyby (same color scheme as Fig. 5).