JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 7, September 2010

Automated Multigravity Assist Trajectory Planning with a Modified Ant Colony Algorithm

Matteo Ceriotti*
University of Strathclyde, Glasgow, UK

Massimiliano Vasile†
University of Glasgow, Glasgow, UK

DOI: 10.2514/1.48448

The paper presents an approach to transcribe a multigravity assist trajectory design problem into an integrated planning and scheduling problem. A modified Ant Colony Optimization algorithm is then used to generate optimal plans corresponding to optimal sequences of gravity assists and deep space maneuvers to reach a given destination. The modified Ant Colony Algorithm is based on a hybridization between standard ant colony optimization paradigms and a tabu-based heuristic. The scheduling algorithm is integrated into the trajectory model to provide a fast time-allocation of the events along the trajectory. The approach demonstrated to be very effective on a number of real trajectory design problems.

Nomenclature

A, B, C	polynomial coefficients
a	semimajor axis
b	direction of deflection (binary)
E	eccentric anomaly
e	eccentricity
f_{obj}	objective value, km/s
$f_{p/a}$	binary variable for pericenter or apocenter
$\mathbf{G}$	matrix of combinations of types of transfer
i	generic index for the leg
j	generic index
$\mathbf{K}$	set of Keplerian orbital elements
k	periodicity coefficient
$\mathcal{L}$	list
M	point in deep space
m_{DSM}	magnitude of DSM, m/s
n_P	number of planets
n_{eval}	number of function evaluations

Received 7 December 2009; accepted for publication 17 June 2010. Copyright © 2010 by the Matteo Ceriotti and Massimiliano Vasile. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 1542-9423/10 $10.00 in correspondence with the CCC.

* Research Fellow, Advanced Space Concepts Laboratory, Department of Mechanical Engineering, James Weir Building, AIAA Member.

† Senior Lecturer, Space Advanced Research Team, Department of Aerospace Engineering, James Watt Buildling South, AIAA Member.

n_{gen}	number of generations
n_{iter}	number of iterations
n_{pop}	size of population
n_{rev1}	number of full revolutions in the first arc
n_{rev2}	number of full revolutions in the second arc
P	planet
Pr	probability
p	semilatus rectum
$\mathcal{Q}$	ordered set
q	generic element in the set
S	list of solutions
R_P	mean radius of the planet
$\mathbf{r}$	position vector
r_p	radius of pericenter (absolute)
r_{ps}	signed radius of pericenter
$\mathbf{s}$	solution vector
T	time of flight, d
t	time
U	random function
$\mathbf{v}$	velocity vector, km/s
w_{planet}	weight for planet selection
w_{type}	weight for type of transfer selection
x, y	cartesian coordinates, km
α	fraction of time of flight at which the DSM occurs
Δv	change in velocity due to DSM, m/s
Δ_θ	anomaly increment, rad
δ	deflection angle
ϵ	admissibility threshold, km/s
θ	true anomaly
λ	variable for radius of pericenter or launch velocity
μ	planetary constant
σ	time of flight weight, km/s/d
$\boldsymbol{\tau}$	pheromone distribution vector
ϕ	difference in anomalies
φ_0	launch angle
Ω	right ascension of ascending node
ω	anomaly of pericenter

Superscripts

$-$	incoming
$+$	outgoing
$*$	optimal
(1), (2)	first and second solution

Subscripts

0	at launch
d	at discontinuity
dep	absolute at departure
DSM	referred to DSM
int	intersection
l	index for feasible solution

o	on orbit of planet
P	referred to the planet
s	on second arc
temp	temporary
tn	in tangential and normal components
xy	in cartesian components
∞	relative at infinity

I. Introduction

IN the literature on multigravity assist (MGA) trajectories, their automatic design (i.e., the definition of an optimal sequence of planetary encounters and the definition of one or more locally optimal trajectories for each sequence) has been approached with several different techniques. All of them can be classified in two main categories: two-level approaches and integrated approaches.

Two-level approaches split the problem into two subproblems which lay at two different levels: one subproblem is to find the optimal sequence of planetary encounters, the other is to find an optimal trajectory for that sequence. Two-level approaches define the planetary sequence independently of the trajectory itself. Once the sequence (or a set of promising sequences) has been selected, then one or more optimal trajectories can identified for each sequence in the set [1]. Two-level approaches use a simplified, low fidelity, model for representing the trajectory [2] at the first level. The use of a low-fidelity model allows for a quick assessment of many sequences, if not all. At the second level, a higher-fidelity, more computationally expensive model is used instead [3]. Each sequence is represented by a string of integer numbers, whereas the associated trajectory is represented with a string of real and integer numbers defining the time and the characteristics of the events occurring along the trajectory [e.g., launch, deep space maneuver (DSM), arrival at a celestial body, number of revolutions around the Sun, etc.]. Therefore, for each sequence, there is an infinite variety of possible trajectories.

The issue with two-level approaches is the difficult assessment of the optimality of a given planetary sequence, without an exhaustive search for all possible trajectories associated with that sequence. Unfortunately, finding an optimal trajectory is a very difficult global optimization problem in itself. This, combined with the fact that usually there exists a very high number of sequences for a given transfer problem, requires a considerable computational effort. The computational cost can be reduced by discarding nonpromising sequences. However, if the low-fidelity model is not accurate enough, either some good sequences are discarded, or many of the retained ones can result to be actually bad.

As opposed to the two-level approaches, integrated approaches define a mixed integer-continuous optimization problem, which tackles both the search of the sequence and the optimization of the trajectory, using a single model, at the same time [4]. This kind of problem is known in literature as a hybrid optimization problem [5,6]. The main difficulty with integrated approaches is that a variation of even a single celestial body in the sequence corresponds to a substantially different set of trajectories. Therefore, if the solution of the hybrid optimization problem is represented with a single vector, a small variation of some of its components can lead to a huge variation of the cost function. In addition, a variation of the length of the sequence implies varying the number of legs of the trajectory, and thus the total length of the solution vector.

The automatic design of a trajectory with discrete events was recently formulated by Ross and D'Souza [5] as a Hybrid Optimal Control Problem, and a solution was proposed by Wall and Conway [7] with a two-level approach based on Genetic Algorithms. The approach proposed by Wall and Conway does not employ models with different fidelity, removing one of the issue related to other two-level approaches.

In this paper, it is proposed to formulate the automated design of an MGA trajectory as an autonomous planning and scheduling problem. The resulting scheduled plan will provide the planetary sequence for an MGA trajectory and a good estimation of the optimality of the associated trajectories.

Although the proposed method can fall in the category of the integrated approaches, the scheduling and the planning of the events are separated at two different levels. At lower level, a scheduler, integrated within the MGA trajectory model, schedules all the events and provides an estimation of the feasibility and quality of the trajectories. At upper level, an algorithm, partially inspired by the Ant Colony Optimization (ACO) paradigm [8], generates

plans to be submitted to the scheduler. The scheduler is integrated into the trajectory model. The model implements a simplified planar representation of an MGA trajectory in which DSMs are applied only at the apsides of conic arcs and the variation of the velocity is parallel to the local tangent. The experimental results in this paper will demonstrate that these simplifying assumption are reasonable and provide acceptable results. Note that the ACO planner, developed in this paper, is independent of the model and accessible as a black box, or oracle providing the feasibility of the transfer and its cost.

ACO was originally created to solve the Traveling Salesman Problem (TSP) [9], and later successfully applied to a number of other discrete optimization problems. In the literature, some ACO-derived meta-heuristics exist for the specific solution of different scheduling problems. In particular, Merkle et al. [10] proposed to apply ACO to the solution of the Resource-Constrained Project Scheduling Problem, while Blum [11] suggested the hybridization of ACO with a probabilistic version of Beam Search for the solution of the Open Shop Scheduling problem. Here, the original idea behind ACO is elaborated to solve planning problems in which the optimality of a particular action (e.g., a transfer from a celestial body to another, in the case of MGA trajectories) is strongly dependent on the history of all preceding actions.

The paper is structured as follows: at first the trajectory model with the integrated scheduler will be presented, then the ACO-based algorithm is illustrated with a description of how plans are constructed; a discussion will follow comparing the proposed planning algorithm against standard ACO. Finally, two case studies will demonstrate the effectiveness of the proposed approach.

II. Trajectory Model

Conceptually, an MGA trajectory can be seen as a scheduled sequence of events (e.g., launch, DSM, swing-by, planetary capture, etc.) characterized by a set of integer variables, identifying the type of event, and a set of real variables identifying the time and characteristics of the event.

The proposed trajectory model is an integral part of the solution process and is used to schedule the events. The model is based on a two-dimensional linked conic approximation: the trajectory is composed of a sequence of planar conic arcs linked together through discrete, instantaneous events. In particular, the sequence is continuous in position and piecewise continuous in velocity, i.e., each event introduces a discontinuity in the velocity of the spacecraft but not in its position. Note that, although the assumption that the trajectories are planar may seem very reductive, in the solar system, the inclinations of planetary orbits are very small (below 3°), with the exception of Mercury and Pluto. Pluto cannot be used for a swing-by, being the farthest of the bodies in the solar system. Mercury is definitely an appealing target, however, as demonstrated by the NASA Messenger mission [12] and the European Space Agency (ESA) BepiColombo mission [13]. A test case will show that the assumption of planarity is acceptable, and yields good solutions even for a transfer to Mercury. On the other hand, the model cannot be used for missions which have, by necessity, to go out of the ecliptic plane [14], such as the ESA–NASA mission Ulysses [15].

In summary, the proposed trajectory model is composed of: a launch from the departure celestial body; a series of deep space flight legs connected by gravity assist maneuvers (modeled through a linked-conic approximation); and a capture into an orbit at a target celestial body. Each one of these events will be explained in the following together with the way they are scheduled along the trajectory.

A. Launch

The launch event is modeled as an instantaneous change of the velocity of the spacecraft with respect to the departure planet. The velocity change is given in terms of the modulus v_0 (which depends on the capabilities of the launcher) and the in-plane direction, specified through the angle φ_0, measured counterclockwise with respect to the planet's orbital velocity vector $\mathbf{v}_P$ at the time of launch t_0 (see Fig. 1a).

According to Fig. 1a, the initial relative velocity of the spacecraft, defined with respect to a reference frame centered in the planet and having the axes tangential and normal to its orbit ($\hat{\mathbf{t}}$, $\hat{\mathbf{n}}$), is

$$\mathbf{v}_{0,\mathrm{tn}} = v_0 \left[\cos\varphi_0, \sin\varphi_0\right]^{\mathrm{T}} \tag{1}$$

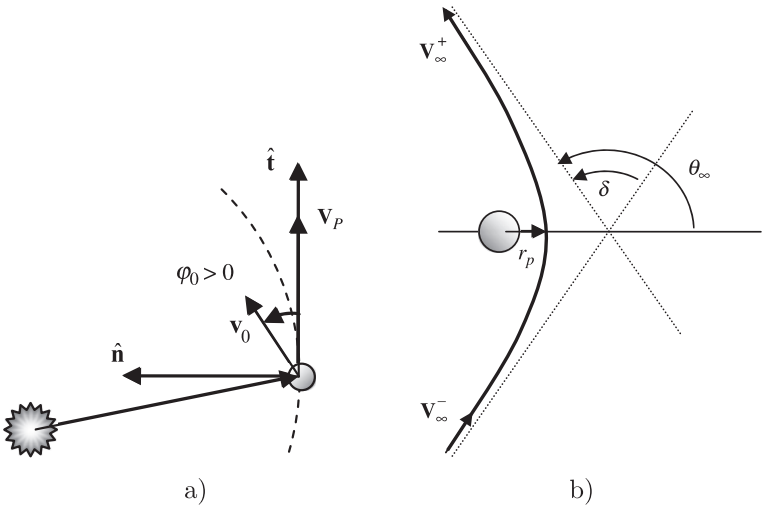

Fig. 1 a) Geometry of the launch event. b) Geometry of the swing-by event.

The vector is then projected onto the heliocentric Cartesian reference frame, to give $\mathbf{v}_{0,xy}$, and added to the velocity $\mathbf{v}_P$ of the planet to give

$$\mathbf{v}_{\text{dep}} = \mathbf{v}_{0,xy} + \mathbf{v}_P \tag{2}$$

The departure time t_0 and the direction φ_0 are free parameters of the model, while the launch velocity modulus v_0 will be used to target the next planetary encounter and solve the phasing problem (see Sec. II.D).

B. Swing-by Model

Gravity assist maneuvers, or swing-by's, are modeled as instantaneous changes of the velocity vector of the spacecraft due solely to the gravity field of the planet. Given the velocity vector $\mathbf{v}^-$ prior to the swing-by (see Fig. 1b), the relative incoming velocity at infinity is defined as

$$\mathbf{v}_\infty^- = \mathbf{v}^- - \mathbf{v}_P \tag{3}$$

The physical properties of unperturbed hyperbolic orbital motion [16] prescribe that

$$v_\infty^+ = v_\infty^- = v_\infty \tag{4}$$

which means that the modulus of the outgoing velocity v_∞^+ at infinity is known. Its direction can be computed considering the anomaly of the incoming asymptote

$$\theta_\infty = \arccos\left(\frac{-\mu_P/r_p}{v_\infty^2 + \mu_P/r_p}\right) \tag{5}$$

In this formula, μ_P is the gravity constant of the planet, and r_p is the radius of the pericenter of the hyperbola. The value of r_p can be used to control the deflection of the incoming velocity and is limited to above the radius of the planet, R_P, to avoid a collision, or to above the atmosphere to preserve incoming v_∞.

The deflection angle of the asymptotic relative velocity vector, due to the planet gravity field, is

$$\delta = b(2\theta_\infty - \pi) \tag{6}$$

where $b = \pm 1$ is a binary variable defining the direction of the deflection, i.e., clockwise or counter-clockwise. In fact, in the linked conic approximation the actual planetocentric trajectory is not defined, thus both $(2\theta_\infty - \pi)$ and $(\pi - 2\theta_\infty)$ are acceptable deflection angles. In order to avoid introducing an additional parameter, in the practical

implementation on this model, we will make use of a signed radius of pericenter r_{ps} that can assume negative values, such that $r_p = |r_{ps}|$ and $b = \text{sgn}(r_{ps})$.

The outgoing relative velocity is found by rotating the incoming velocity by δ:

$$\mathbf{v}_\infty^+ = \begin{bmatrix} \cos\delta & \sin\delta \\ -\sin\delta & \cos\delta \end{bmatrix} \mathbf{v}_\infty^- \tag{7}$$

and finally, the absolute velocity is:

$$\mathbf{v}^+ = \mathbf{v}_\infty^+ + \mathbf{v}_P \tag{8}$$

As for the launch velocity magnitude, the radius of pericenter r_{ps} is tuned to meet the terminal conditions of the transfer leg following the swing-by.

C. Deep Space Flight

Each deep space flight leg is made of two conic arcs linked, at a point M_i, through a single discrete event. The leg starts at a departure planet P_i and ends at an arrival planet P_{i+1}. The event is an instantaneous change in the heliocentric velocity vector of the spacecraft, or DSM, due to an ignition of the engines. In this model, we assume that the DSM is performed either at the apocenter or pericenter of the conic arc preceding the maneuver. In addition, the change in velocity is tangential to that arc. As a consequence, the DSM will raise or decrease either the pericenter or the apocenter of the orbit, without changing the line of apsides.

1. *First Arc*

Let us assume that the spacecraft is at a given planet P_i at time t_i. Its position $\mathbf{r}_i$ coincides with that of the planet $\mathbf{r}_{P_i}$, which is known from the ephemeris. The heliocentric velocity of the spacecraft $\mathbf{v}_i$, instead, depends on either v_0, when the first arc starts from planet P_0, or r_{ps}. The initial state $[\mathbf{r}_i, \mathbf{v}_i]$ can be converted into the six Keplerian elements $\mathbf{K}_i = [a_i, e_i, 0, 0, \omega_i, \theta_i]^{\mathrm{T}}$, where a_i is the semimajor axis, e_i is the eccentricity, the inclination and the right ascension of the ascending node are zero, ω_i is the argument of the periapsis, and θ_i is the true anomaly.

If the transfer leg contains a DSM (see Fig. 2a), the position of point M_i is arbitrarily set to be either the pericenter or the apocenter, according to the binary variable $f_{p/a,i}$. Therefore, the true anomaly θ_{DSM_i} of the ith DSM is given by

$$f_{p/a,i} = \begin{cases} 0 \Rightarrow & \theta_{\text{DSM}_i} = 0 \\ 1 \Rightarrow & \theta_{\text{DSM}_i} = \pi \end{cases} \tag{9}$$

The Keplerian parameters at point M_i, before performing the maneuver, are

$$\mathbf{K}_{\text{DSM}_i}^- = \left[a_i, e_i, 0, \omega_i, 0, \theta_{\text{DSM}_i}\right]^{\mathrm{T}} \tag{10}$$

The position vector $\mathbf{r}_{\text{DSM}_i}$ of the DSM and the velocity vector before performing the maneuver $\mathbf{v}_{\text{DSM}_i}^-$ are computed from $\mathbf{K}_{\text{DSM}_i}^-$. The time of the DSM is found by first computing the eccentric anomaly corresponding to the departure point θ_i:

$$E_i = 2\arctan\sqrt{\frac{1-e_i}{1+e_i}}\tan\frac{\theta_i}{2} \tag{11}$$

Then, by using Kepler's time law,

$$t_{\text{DSM}_i} = \sqrt{\frac{a_i^3}{\mu}}(2\pi n_{\text{rev},1} + E_{\text{DSM}_i} - E_i + e_i \sin E_i) + t_i \tag{12}$$

where $E_{\text{DSM}_i} = \theta_{\text{DSM}_i} + 2k\pi$, since the maneuver is either at pericenter or apocenter, and the integer k must be chosen such that E_{DSM_i} follows E_i. The integer quantity $n_{\text{rev1},i} \geqslant 0$ is the number of full revolutions before the DSM.

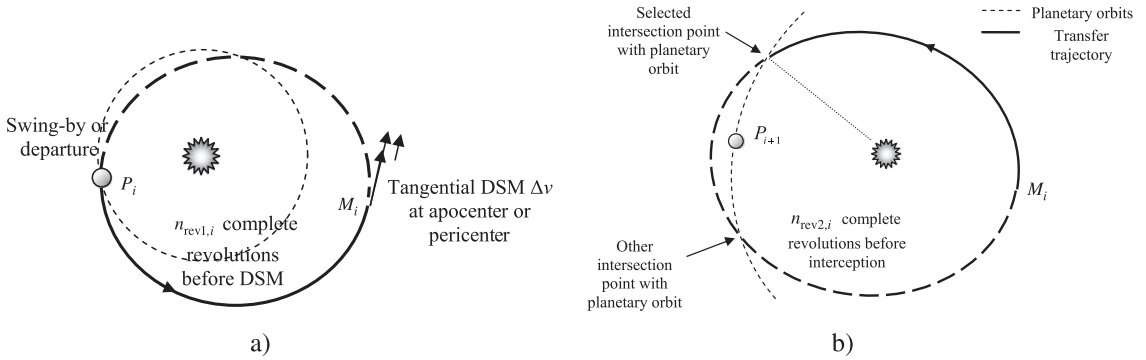

Fig. 2 a) First arc, from planet P_i up to point M_i; the parameter $n_{\text{rev1},i}$ defines the number of full revolutions (dashed trajectory). b) Second arc from M_i to the selected orbital intersection with the planet; $n_{\text{rev2},i}$ full revolutions are performed (dashed trajectory) before the orbital intersection.

The velocity right after performing the DSM is given by

$$\mathbf{v}^+_{\text{DSM}_i} = \mathbf{v}^-_{\text{DSM}_i} + \frac{\mathbf{v}^-_{\text{DSM}_i}}{v^-_{\text{DSM}_i}} m_{\text{DSM}_i} \tag{13}$$

The parameter m_{DSM_i} is the magnitude and direction of the DSM: if m_{DSM_i} is positive, the thrust is along the velocity vector of the spacecraft, otherwise it is against the velocity of the spacecraft. The velocity vector $\mathbf{v}^+_{\text{DSM}_i}$ is used to compute the postmaneuver orbital elements $\mathbf{K}^+_{\text{DSM}_i}$. The time, states and orbital elements right after the DSM define also the time, states and orbital elements at point M_i:

$$\begin{aligned} t_{M_i} &= t_{\text{DSM}_i} \\ \mathbf{r}_{M_i} &= \mathbf{r}^+_{\text{DSM}_i} \\ \mathbf{v}_{M_i} &= \mathbf{v}^+_{\text{DSM}_i} \\ \mathbf{K}_{M_i} &= \mathbf{K}^+_{\text{DSM}_i} \end{aligned} \tag{14}$$

If the leg does not contain any DSM, the first arc is propagated up to a fictitious point M_i defined in terms of anomaly increment Δ_θ. The states of the spacecraft at point M_i, $\mathbf{r}_{M_i}$, $\mathbf{v}_{M_i}$ are computed from the Keplerian parameters:

$$\mathbf{K}_{M_i} = \mathbf{K}_i + [0, 0, 0, 0, 0, \Delta_\theta] \tag{15}$$

The reason for using this forced propagation is twofold: first, to prevent that, if no full revolutions are considered, the first intersection occurs after a null time; second, to prevent any event (e.g., a DSM or another swing-by) from happening immediately after the swing-by or at the same time, which would be infeasible due to operational constraints. The quantity Δ_θ has to be larger than the machine numerical precision but small enough to allow for the modeling of short transfer legs. It is important to underline that Δ_θ is not a design parameter, its value is arbitrary and does not affect the planning process. The only impact is on the time of the first intersection and therefore on the acceptable minimum length of the transfer arc. The acceptable minimum length can be easily decided a priori. For this work, a value $\Delta_\theta = 0.3$ rad (about 17°) was chosen.

The time at M_i is found by solving Kepler's time law

$$t_{M_i} = \sqrt{\frac{a_i^3}{\mu}}(E_{M_i} - E_i - e_i(\sin E_{M_i} - \sin E_i)) + t_i \tag{16}$$

where E_{M_i} is

$$E_{M_i} = 2\arctan\sqrt{\frac{1-e_i}{1+e_i}}\tan\frac{\theta_{M_i}}{2} + 2k\pi \tag{17}$$

with $\theta_{M_i} = \theta_i + \Delta_\theta$ and k such that E_{M_i} follows E_i. Note that, as the DSM can only be at the pericenter or apocenter, transfer legs containing a DSM cannot be shorter than the time required to reach either the pericenter or the aproceter.

2. Second Arc

The second arc starts at point M_i with states $[\mathbf{r}_{M_i}, \mathbf{v}_{M_i}]$ and is propagated until the intersection with the orbit of planet P_{i+1} (see Fig. 2b).

The intersections between the second arc and the orbit of the planet can be found by solving the following system of equations:

$$\begin{aligned} r_s &= r_o \\ \theta_s + (\omega_{M_i} + \Omega_{M_i}) &= \theta_o + (\omega_{P_{i+1}} + \Omega_{P_{i+1}}) \end{aligned} \tag{18}$$

The radius r_s along the second arc and the radius r_o along the orbit of planet P_{i+1} are given by:

$$r_s = \frac{p_{M_i}}{1 + e_{M_i}\cos\theta_s} \tag{19}$$

$$r_o = \frac{p_{P_{i+1}}}{1 + e_{P_{i+1}}\cos\theta_o} \tag{20}$$

where $p_{M_i} = a_{M_i}(1 - e_{M_i}^2)$ and $p_{P_{i+1}} = a_{P_{i+1}}(1 - e_{P_{i+1}}^2)$ are, respectively, the semilatus rectum of the orbit of the spacecraft, and planet. By defining $\phi = (\omega_{M_i} + \Omega_{M_i}) - (\omega_{P_{i+1}} + \Omega_{P_{i+1}})$ and combining Eq. (18) with Eq. (19), after some algebra, we can get

$$(p_{M_i}e_{P_{i+1}}\cos\phi - p_{P_{i+1}}e_{M_i})\cos\theta_s - (p_{M_i}e_{M_i}\sin\phi)\sin\theta_s + p_{M_i} - p_{P_{i+1}} = 0 \tag{21}$$

that is a linear equation in $\sin\theta_s$ and $\cos\theta_s$. Now, using the transformation $t = \tan(\theta_s/2)$, Eq. (21) becomes

$$(C - B)t^2 + 2At + (B + C) = 0 \tag{22}$$

where $A = (p_{M_i}e_{P_{i+1}}\cos\phi - p_{P_{i+1}}e_{M_i})$, $B = (p_{M_i}e_{M_i}\sin\phi)$, and $C = p_{M_i} - p_{P_{i+1}}$. Equation (22) has solutions

$$\theta_s = 2\arctan\left(\frac{-A \pm \sqrt{A^2 + B^2 - C^2}}{C - B}\right) + 2k\pi, \quad k \in \mathbb{Z}. \tag{23}$$

If $A^2 + B^2 - C^2 < 0$, then there are no real solutions to Eq. (22), which means that the spacecraft's orbit does not intersect P_{i+1}'s orbit. Therefore, either the initial conditions of the leg, or the parameters m_{DSM_i} and $f_{p/a,i}$, have to be modified. If instead $\Delta = A^2 + B^2 - C^2 \geqslant 0$, then Eq. (23) yields two solutions $\theta_s^{(1)}$ and $\theta_s^{(2)}$ with periodicity n. Since only the first two intersections are of interest, we can neglect the periodicity by setting $n = 0$. The true anomalies of the two intersections along the orbit of the planet can be derived from the second equation of system (18)

$$\theta_o^{(1)(2)} = \theta_s^{(1)(2)} + \phi \tag{24}$$

One of the two intersections is then selected according to the value of the binary variable $f_{1/2,i}$, such that

$$f_{1/2,i} = \begin{cases} 0 \to \theta_{\mathrm{int}} = \theta_s^{(1)}, & \bar{\theta} = \theta_o^{(1)} \\ 1 \to \theta_{\mathrm{int}} = \theta_s^{(2)}, & \bar{\theta} = \theta_o^{(2)} \end{cases} \tag{25}$$

where θ_{int}, $\bar{\theta}$ are the true anomalies of the selected intersection, respectively, along the orbit of the spacecraft, and of the planet. From θ_{int}, the time of intersection t_{int} can be computed with Kepler's time law:

$$t_{\text{int}} = \sqrt{\frac{a_{M_i}^3}{\mu}}(2\pi n_{\text{rev2},i} + E_{\text{int}} - e_{M_i}\sin E_{\text{int}} - E_{M_i} + e_{M_i}\sin E_{M_i}) + t_{M_i} \tag{26}$$

where E_{int} is computed from θ_{int} using Eq. (11). The integer variable $n_{\text{rev2},i} \geqslant 0$ defines the number of full revolutions along the second arc. Finally, the Keplerian parameters at the intersection point are:

$$\mathbf{K}_{\text{int}} = \left[a_{M_i}, e_{M_i}, 0, \omega_{M_i}, 0, \theta_{\text{int}}\right] \tag{27}$$

from which the state vector of the spacecraft $[\mathbf{r}_{\text{int}}, \mathbf{v}_{\text{int}}]$ can be computed.

D. Solution of the Phasing Problem

In order to perform a gravity assist maneuver or a planetary capture, the terminal position of the spacecraft has to match that of the planet. However, at intersection time t_{int}, planet P_{i+1} is at true anomaly $\theta_{P_{i+1}}$, which is generally different from $\bar{\theta}$. From Eqs. (10), (14), and (26), the time of intersection is a function of the orbital parameters of the first and second arc and therefore of the states at the beginning of the first and second arc. If the DSM is provided by the planner, the time of intersection is a function solely of v_0 or r_{ps}, depending on the starting event. Therefore, if one introduces the parameter λ, defined as

$$\lambda \equiv \begin{cases} r_{ps}, & \text{if } i > 0 \\ v_0, & \text{if } i = 0 \end{cases} \tag{28}$$

the true anomalies of the intersection point and of the planet can be expressed as $\bar{\theta}(\lambda)$ and $\theta_{P_{i+1}}(\lambda)$. Matching the position of the planet with that of the intersection point at time t_{int} (also known as the phasing problem), then, translates into finding a value $\lambda = \lambda^*$ that satisfies the equation (see Fig. 3)

$$\Delta\theta(\lambda^*) = \theta_{P_{i+1}}(\lambda^*) - \bar{\theta}(\lambda^*) = 0 \tag{29}$$

Figures 4 and 5 represent the function $\Delta\theta(\lambda)$ for different transfer cases. Figure 4 shows the nonresonant transfers: Fig. 4a is from Venus to Mercury, following a swing-by of Venus. In this case, the parameter λ is the radius of pericenter of the swing-by r_{ps}. Figure 4b is from Earth to Venus after launching from Earth, so $\lambda \equiv v_0$.

Figure 5, instead, refers to resonant transfers: Fig. 5a is a Venus-to-Venus transfer starting with a swing-by; Fig. 5b is an Earth-to-Earth transfer, starting with launch. It is worth noting that for some values of λ, $\Delta\theta(\lambda)$ is not defined: this is the case when there is no possible orbit intersection. Examples are in Fig. 4a, for $r_p/R_P > 2.1$, and Fig. 4b, for $v_0 < 2.6$ km/s. This is in fact the minimum excess velocity to reach the orbit of Venus from Earth (with $\varphi_0 = \pi$, as in this case). Furthermore, when a leg follows a swing-by, r_p is limited by the radius of the planet, which introduces

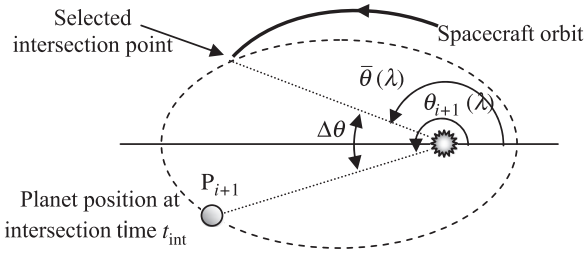

Fig. 3 Geometry of the phasing problem.

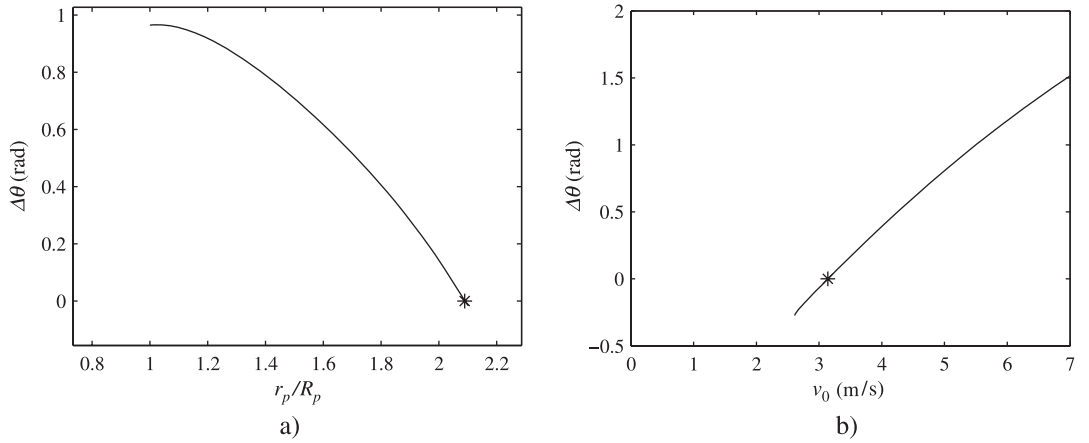

Fig. 4 a) Venus to Mercury leg following a swing-by of Venus ($m_{\mathrm{DSM}} = 0$ m/s, four full revolutions). b) Earth to Venus leg following launch from Earth ($m_{\mathrm{DSM}} = 600$ m/s, no full revolutions).

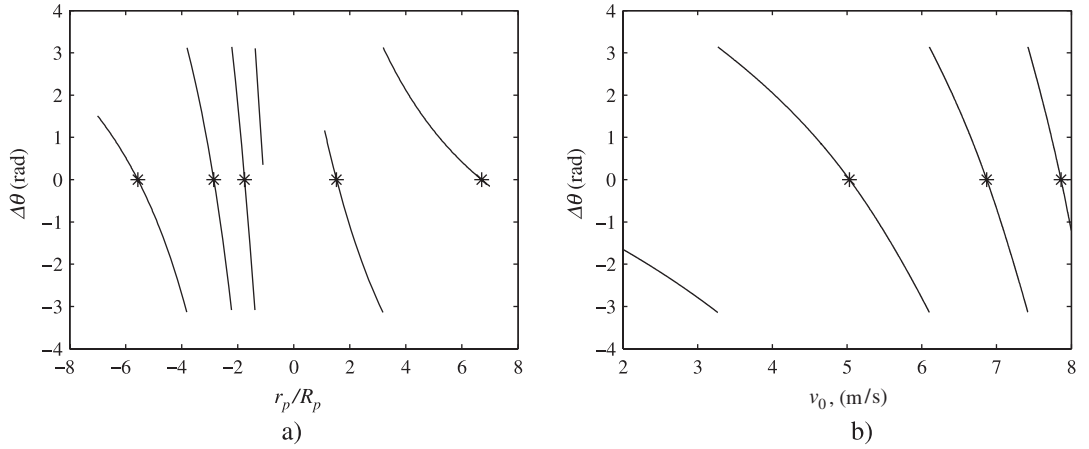

Fig. 5 a) Venus-to-Venus leg following a swing-by of Venus ($m_{\mathrm{DSM}} = 0$, three full revolutions). b) Earth-to-Earth leg following launch from Earth ($m_{\mathrm{DSM}} = 0$, no full revolutions).

the constraint

$$\left(\frac{r_p}{R_P} < -1\right) \vee \left(\frac{r_p}{R_P} > 1\right) \tag{30}$$

Constraint (30) is the reason for the gap in Fig. 5a. The cases depicted in Figs. 4a and 4b, show that the function $\Delta\theta(\lambda)$, is continuous, smooth, and monotonic over the range of interest of λ. Hence, the phasing problem has only one solution. This solution can be found with a simple Newton–Raphson method in one dimension. However, when a resonant transfer is considered, as in Figs. 5a and 5b, $\Delta\theta(\lambda)$ is discontinuous and multiple zeros exist. Each zero corresponds to a different resonance with the planet (and of course a different transfer time). The discontinuity is due to the cyclic nature of $\Delta\theta$. In fact, say λ_d is the value of λ at which $\Delta\theta$ is discontinuous, then $\lim_{\lambda\to\lambda_d^-} \Delta\theta = -\pi$, and $\lim_{\lambda\to\lambda_d^+} \Delta\theta = +\pi$, i.e., the planet and the spacecraft are on the opposite sides of the planet's orbit.

Note that, since there is no easy way, at a given transfer, to prefer one value of λ^* over another, all the solutions need to be retained for the evaluation of the following transfers.

In the present implementation, the search for the zeros of the function $\Delta\theta$ is performed with the Brent method [17]. This method resulted to be fast and robust, since it uses a Newton-based iteration for quick local convergence, but switches to a bisection-like method to overcome discontinuities and capture multiple solutions. A set of starting

points, defining multiple intervals for the bisection method, need to be provided to initialize the Brent method and are specified case by case.

E. Complete Trajectory

A complete trajectory is made of a sequence of transfers connecting n_P celestial bodies $[P_0, P_1, \ldots, P_{n_P}]$. Thus, a complete trajectory with $n_P + 1$ planets has n_P transfers, and $n_P - 1$ swing-by's.

The solution of Eq. (29), together with Eqs. (12), (16), and (26) provides a complete scheduling of the trajectory given the initial time t_0 and the five parameters m_{DSM_i}, $n_{\mathrm{rev1},i}$, $n_{\mathrm{rev2},i}$, $f_{p/a,i}$, and $f_{1/2,i}$ for every $i = 0, \ldots, n_P - 1$.

Since these five parameters fully characterize all possible legs from a planet P_i to a planet P_{i+1}, they are said to define a *type of transfer*. Conversely, because of the multiplicity of the zeros of Eq. (29), each transfer corresponds to a set of legs.

Hence, assigning a value to t_0, φ_0, P_i, m_{DSM_i}, $n_{\mathrm{rev1},i}$, $n_{\mathrm{rev2},i}$, $f_{p/a,i}$, and $f_{1/2,i}$ for $i = 0, \ldots, n_P - 1$ creates a tree structure in which every branch is a trajectory. Algorithm 1 illustrates the procedure to keep track of all the trajectories in the tree. The algorithm yields a list $\mathcal{L}$ containing all the possible conditions of arrival at the last reachable planet. In fact, if no leg in the set associated to transfer i satisfies the phasing problem, then planet $i + 1$ cannot be reached and the algorithm terminates. Furthermore, an upper bound on the time of flight of the entire trajectory, or of some legs, is introduced. Trajectories that exceed the total or partial time of flight constraint are discarded from the list. The information of infeasibility at a given transfer will be used to fill in a tabu list of broken or impracticable solutions.

Algorithm 1 $\mathcal{L}$ list generation

```
For i = 0 find all possible v_0^* | Δθ(v_0^*) = 0
for all v_0^* do
    find the final conditions of the first leg
    add the final conditions to the list L
    for all i = 1, ..., n_P do
        L_temp ← Ø
        for all elements in L do
            Find all possible r_ps^* | Δθ(r_ps^*) = 0
            for all r_ps^* do
                Find the final conditions at planet P_{i+1}
                Add final conditions to the list L_temp
            end for
        end for
        if L_temp = Ø then
            Exit
        end if
    end for
    L ← L_temp
end for
```

The entire tree is a complete transfer from P_0 to P_{n_P} and represents a solution of the MGA trajectory planning problem. Thus, a plan is fully defined by assigning a value to the parameters in Table 1 for all $i = 0, \ldots, n_P - 1$. A partial or incomplete plan is the set of parameters sufficient to describe a solution up to transfer i. Furthermore, if Algorithm 1 exits at planet P_i, the plan is broken and the solution is said to be infeasible at transfer i.

For each solution of the MGA planning problem, the trajectory model computes:

1) The sum of all the DSMs, or total Δv and the launch excess velocity, v_0, which is the result of the phasing problem for the first leg,
2) The relative velocity at the last planet, v_∞. This value is usually important for assessing the optimality of a trajectory, as a low v_∞ implies that a small maneuver is needed for the spacecraft to be captured by the target planet,

Table 1 Summary of the free design parameters defining a solution to the MGA trajectory planning problem

Description	Variables
Planetary sequence	$[P_0, P_1, \ldots, P_{n_P}]$
Departure time	t_0
Departure angle	φ_0
Transfer types for $i = 0, \ldots, n_P - 1$	$[m_{\mathrm{DSM}_i}, n_{\mathrm{rev1},i}, n_{\mathrm{rev2},i}, f_{p/a,i}, f_{1/2,i}]$

3) The total time of flight of the trajectory. The total time of flight is important when assessing the trajectory, as long missions may not be feasible due to excessive cost of the operations.

The whole trajectory model was implemented in ANSI C and compiled as a MEX-file for interfacing with MATLAB.

III. ACO–MGA Algorithm

The model described in the previous section yields a set of scheduled trajectories provided that a complete or partial plan is available. In this section, we present a planning algorithm based on the ant colony paradigm.

At first, the continuous space of the real parameters t_0, φ_0, and m_{DSM_i} is reduced to a finite set of states. Then the optimization algorithm, called ACO–MGA in the following, operates a search in the finite space of possible values for the design parameters in Table 1. A complete description of the algorithm ACO–MGA follows.

A. Solution Coding

In ACO–MGA, a plan solution is fully defined by assigning values to all the parameters in Table 1. However, the set of parameters is inhomogeneous as it is made of real, integer, and binary variables. In particular t_0, φ_0, and m_{DSM_i} are real, continuous variables and need to be properly discretized. In the present implementation, the values of the departure date t_0 and the departure angle φ_0 are assumed to be preassigned, therefore the two parameters are removed from the list of the variables. The rationale behind this choice is that, if an algorithm exists that is able to efficiently generate a complete plan for a given t_0, then an unidimensional search in the time domain can be performed to find the optimal launch date. The angle φ_0 on the other hand can very often be estimated depending on the mission: usually a tangential departure excess velocity is used for nonresonant legs in order to maximize the change in the semimajor axis. The departure excess velocity will be in the same direction of the planet heliocentric velocity, i.e., $\varphi_0 = 0$, if the second planet in the sequence is outwards; vice versa, the launch will be in the opposite direction, $\varphi_0 = \pi$, if the second planet is inwards [1].

For resonant legs, instead, very often $\varphi_0 = \pm\pi/2$ as this value allows for the maximization of the radial component of the relative velocity vector at the following swing-by [18]. Furthermore, it is assumed that the departure planet P_0 is given, as is consistent with a great majority of the applications.

Using the additional assumptions on t_0, φ_0, and P_0, each solution representing a plan can be encoded using a vector $\mathbf{s}$ of positive integers. The vector has $2n_P$ components. Each pair of consecutive components encodes all the parameters necessary to characterize one transfer, or segment of the plan (see Fig. 6). The first element of the pair encodes the identification number of the target planet according to the following procedure: an ordered set $\mathcal{Q}_{P,i}$

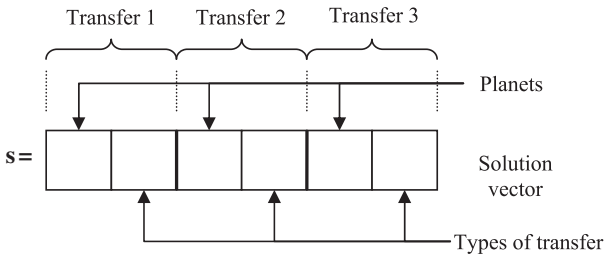

Fig. 6 Vector for coding a three-leg solution.

containing all the celestial bodies available as targets for transfer i is predefined, then if $k = s_{2(i-1)+1}$, the target planet is $q_{P,ik} \in \mathcal{Q}_{P,i}$.

The second element of the pair is the row index of the matrix $\mathbf{G}_i$ containing all possible combinations of indexes identifying the elements of the five sets: $\mathcal{Q}_{1,i} = \{q_{1,i} | q_{1,i} \in \mathbb{R}\}$, $\mathcal{Q}_{2,i} = \{q_{2,i} | q_{2,i} \in \mathbb{N}\}$, $\mathcal{Q}_{3,i} = \{q_{3,i} | q_{3,i} \in \mathbb{N}\}$, $\mathcal{Q}_{4,i} = \{q_{4,i} | q_{4,i} \in \{0, 1\}\}$, and $\mathcal{Q}_{5,i} = \{q_{5,i} | q_{5,i} \in \{0, 1\}\}$. If $\mathcal{Q}_{4,i} = \{0, 1\}$ and $\mathcal{Q}_{5,i} = \{0, 1\}$ then the matrix $\mathbf{G}_i$ is

$$\mathbf{G}_i = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 & 2 \\ 1 & 1 & 2 & 2 & 1 \\ 1 & 1 & 2 & 2 & 2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ |\mathcal{Q}_{1,i}| & |\mathcal{Q}_{2,i}| & |\mathcal{Q}_{3,i}| & 2 & 2 \end{bmatrix} \tag{31}$$

where $|\cdot|$ is the cardinality of a set. Each row of $\mathbf{G}_i$ is a vector representing a different type of transfer. In general, the matrix has $|\mathcal{Q}_{1,i}| \cdot |\mathcal{Q}_{2,i}| \cdot |\mathcal{Q}_{3,i}| \cdot |\mathcal{Q}_{4,i}| \cdot |\mathcal{Q}_{5,i}|$ rows, which is also the number of possible different transfers for a given segment of a plan. The parameters for the jth type of transfer (i.e., jth row of $\mathbf{G}_i$) can be obtained from:

$$m_{\mathrm{DSM}_i} = q_{1,ik_1} \tag{32}$$

$$n_{\mathrm{rev}1,i} = q_{2,ik_2} \tag{33}$$

$$n_{\mathrm{rev}2,i} = q_{3,ik_3} \tag{34}$$

$$f_{p/a,i} = q_{4,ik_4} \tag{35}$$

$$f_{1/2,i} = q_{5,ik_5} \tag{36}$$

where $k_1 = G_{i,j1}$, $k_2 = G_{i,j2}$, $k_3 = G_{i,j3}$, $k_4 = G_{i,j4}$, and $k_5 = G_{i,j5}$.

B. Tabu and Feasible Lists

The transfer from planet P_i to planet P_{i+1} can be feasible or infeasible, for the same set of parameters, depending on all the preceding transfers from 1 to $(i - 1)$. For this reason, when a plan contains an infeasible transfer, it is necessary to store the whole path that led to that infeasible transfer. Thus, all the parameters characterizing the partial solution up to P_i are stored in a tabu list.

In particular, if the problem involves n_P transfers, the same number of tabu lists are used. The tabu list of transfer i contains all the partial solutions, which are feasible up to P_i. The tabu list is stored in a matrix (one for each transfer), which has an arbitrary number of rows and $2i$ columns.

The number of elements in the tabu lists can be limited, to limit the memory requirements and the search time. Once one of the tabu lists is full, the optimizer can either stop or simply start replacing the older elements.

Dual to the list of tabu partial solutions, the feasible list stores all the solutions, which are completely feasible, i.e., reach the destination planet. This is, once more, a matrix with an arbitrary number of rows and $2n_P$ columns. Since each solution contained in the feasible list is complete, then it is possible to associate an objective value to each one of them because the value of the launch excess velocity v_0, all the DSMs, the arrival relative velocity v_∞, and the time of flight T are available. A scalar value can be computed from these quantities identifying the value of the trajectories. In the following test cases, for example, we will use, as objective value, v_∞ and a combination of v_∞ and T. Note that, since, in general, there is more than one trajectory for a given solution, the objective value of a solution is given by the best trajectory value. As for the tabu list, the feasible list length can also be limited for memory saving. In this case, when the list is full, the optimization can either stop or simply the feasible solutions with the worst objective value can be replaced.

C. Plan Generation

The search space is organized as an acyclic oriented tree. Each branch of the tree represents a transfer, whereas each node (or leaf) represents a different destination planet and type of transfer. A population of m virtual ants are dispatched to explore the tree, searching for an optimal solution. The search runs for a given number of iterations $n_{\text{iter,max}}$, or until a maximum number of objective function evaluations $n_{\text{eval,max}}$ has been reached. An evaluation is a call to the trajectory model, in order to compute the objective value associated to a given solution. Algorithm 2 illustrates the main iteration loop. Each iteration consists of two steps: first, a solution generation step (lines 2–8), and then a solution evaluation step (line 9). In the former step, the ants incrementally compose a set of solution vectors, while the latter invokes the trajectory model to assess the feasibility and the objective value of each generated solution. When the main loop of the search stops, the feasible list contains all the solutions, which were found feasible, with their corresponding objective value. The solutions are then sorted according to their objective value.

Algorithm 2 Main ACO–MGA search engine

1: Set number of ants equal to m
2: **for all** $k = 1, \ldots, m$ **do**
3: Generate planetary sequence
4: Generate types of transfers
5: **if s** is not discarded **then**
6: $S \leftarrow S \cup \{\mathbf{s}\}$
7: **end if**
8: **end for**
9: Evaluate all solutions in S
10: Update feasible list and tabu list
11: **Termination** Unless $n_{\text{iter}} > n_{\text{iter,max}} \vee n_{\text{eval}} > n_{\text{eval,max}}$, **goto** Step 1

At each iteration, each one of the m ants explores the tree independently of the others, but taking into account the information collected in the feasible and tabu lists by all the ants at the previous iterations. As an ant moves along a branch, it progressively composes a complete solution by first assigns a value to the odd entries of the solution vector, i.e., composes the sequence of planetary encounters, and then to the even entries of the solution vector, i.e., the parameters defining the types of transfers.

1. Planetary Sequence Generation

As the departure planet is given, an ant has to choose the destination planet for each transfer. The choice is made probabilistically by picking from the list $\mathcal{Q}_{P,i}$. The selection depends on the pheromone distribution vector $\boldsymbol{\tau}_{P,i}$ (one for every transfer), which contains the pheromone level associated to each body in the list $\mathcal{Q}_{P,i}$. Note that we use the same notion of pheromone as in standard ACO [8], however there are some differences. Here, the pheromone level of each possible choice at each leg depends on the previous legs, and therefore it is computed at every step. Furthermore, due to the different pheromone update rule, here the amount of pheromone is not upper limited to 1.

Every time an ant is at transfer i, the pheromone distribution vector is reset to $\boldsymbol{\tau}_{P,i} = [1, 1, \ldots, 1]^{\mathrm{T}}$. As it will be explained, this is equivalent to state that all the planets have equal probability to be chosen. The ant sweeps the entire list $\mathcal{Q}_{P,i}$ substituting the identification number of each element in $\mathcal{Q}_{P,i}$ into the ith odd component of the solution vector $\mathbf{s}$. Then, the feasible list is searched for all the solutions that have a (partial) planetary sequence which matches the one in $\mathbf{s}$. Say that the jth element of $\mathcal{Q}_{P,i}$ is added to $\mathbf{s}$, and the resulting partial sequence matches the partial sequence of the lth solution in the feasible lists, then the pheromone level $\tau_{P,ij}$ associated to the jth element of $\mathcal{Q}_{P,i}$ is increased as follows

$$\tau_{P,ij} \leftarrow \tau_{P,ij} + \frac{1}{f_{\text{obj},l}} w_{\text{planet}} \tag{37}$$

The amount of pheromone which is added depends on the objective value $f_{\text{obj},l}$ of the matching solution in the feasible list, and on the weight w_{planet}. Once the pheromone update has been done for all the possible choices, the

probability of selecting one of them is given by

$$\Pr_{P,ij} = \frac{\tau_{P,ij}}{\sum_j \tau_{P,ij}} \tag{38}$$

and a random selection is performed according to this probability distribution. Thus, the probability of choosing the jth planet increases according to how many times it generates a promising sequence (leading to a feasible solution), to the value of the feasible solution itself, and to the parameter w_{planet}.

This mechanism (summarized in Algorithm 3) is analogous to the pheromone deposition of standard ACO and aims at driving the search of the ants toward good planetary sequences. In fact, those planets which generate (partial) sequences that appear either frequently in the feasible list, or rarely but with a low objective function, are selected with a higher probability. On the other hand, the probability of selecting other planets remains positive, such that one or more ants can probabilistically choose a planet that generates an undiscovered sequence. Note that, if the feasible list is empty, then all the planets have the same probability to be selected.

Algorithm 3 Planetary sequence generator

for all $i = 1, \ldots, n_P$ **do**
  set $\boldsymbol{\tau} \leftarrow [1, 1, \ldots, 1]^{\mathrm{T}}$
  for all target body j available at transfer i **do**
    $s_{(1+2(i-1))} \leftarrow j$
    for all solutions l, in the feasible list, that match **s do**
      $\tau_{P,ij} \leftarrow \tau_{P,ij} + \frac{1}{f_{\text{obj},l}} w_{\text{planet}}$
    end for
  end for
  $s_{(1+2(i-1))} \leftarrow$ SelectProbabilityDistribution($\boldsymbol{\tau}_{P,i}$)
end for

The parameter w_{planet} controls the learning rate of the ants. A low value of w_{planet} will make the term $w_{\text{planet}}/f_{\text{obj},l}$ small, and thus the probability distribution will not change much, even if the solution appears repeatedly in the feasible list, or with low values of f_{obj}. Thus, a relatively low value of w_{planet} will favor a global exploration of the search space, whereas a high value of w_{planet} will greatly increase the probability of choosing a planet which led to a feasible sequence.

Algorithm 4 assigns a value to the index j, given the pheromone distribution vector $\boldsymbol{\tau}_{P,i}$ [8]. The procedure iterates for all the transfers. At the end, all the odd entries of the solution **s** contain a target planet and the planetary sequence is complete. The next step is to find the type of transfers for each segment of the plan, thus filling the even entries of **s** and complete the solution.

Algorithm 4 Function $j \leftarrow$ SelectProbabilityDistribution($\boldsymbol{\tau}$)

$r \leftarrow U(0, 1) \sum_j \tau_j$
$j \leftarrow 1$
$p \leftarrow d_1$
while $p < r$ **do**
  $j \leftarrow j + 1$
  $p \leftarrow p + d_j$
end while

2. Type of Transfer Generation

Once an ant has filled in the odd components of a solution **s**, it proceeds by assigning values to the even components. Similarly to the planet sequence generation, for each transfer all the available types of transfer are assigned, one

at a time, to the solution **s**. A vector **s** for which a value is assigned to both the odd and even components up to leg i represents a partial solution. For each new partial solution, the tabu list is first checked. If the partial solution appears in the tabu list, then it means that this solution will be infeasible regardless of the way it is completed. The pheromone of the type of transfer associated to that sequence is set to zero to avoid future selection of that type of transfer. If the partial solutions does not appear in the tabu list, the feasible list is searched for any matching partial solution. For every match found, the pheromone distribution for that type of transfer is modified as follows:

$$\tau_{t,ij} \leftarrow \tau_{t,ij} + \frac{1}{f_{\text{obj},l}} w_{\text{type}} \tag{39}$$

where the vector $\boldsymbol{\tau}_{t,i}$ contains the pheromone distribution associated to the rows of the matrix $\mathbf{G}_i$, and the weight w_{type} is introduced with analogous meaning to w_{planet}. In fact, the higher the coefficient, the higher the chances that solutions similar to the feasible ones are generated. Conversely, a low value of w_{type} will favor the selection of sequences with a different type of transfer, thus increasing the random exploration of the whole solution space. If, at a given i, all possible transfer types correspond to partial solutions in the tabu list, the vector of pheromone distribution $\boldsymbol{\tau}_{t,i}$ will be full of zeros. As a consequence, the solution **s** (which can be partial or complete) is discarded, and the ant can stop its exploration of that branch of the tree.

At the end of the solution generation step, the solution **s** is either discarded or completed. Once all the ants complete their exploration, the result is a number of solutions (less than or equal to the number of ants m) to be evaluated. The procedure is summarized in Algorithm 5.

Algorithm 5 Transfer type generator

for all $i = 1, \ldots, n_P$ **do**
  set $\boldsymbol{\tau}_{t,i} \leftarrow [1, 1, \ldots, 1]^{\mathrm{T}}$
  for all target body j available at transfer i **do**
    $s_{(2+2(i-1))} \leftarrow j$
    if **s** is in tabu list of transfer i **then**
      $\tau_{t,ij} \leftarrow 0$
    else
      for all solutions l, in the feasible, that match **s** **do**
        $\tau_{t,ij} \leftarrow \tau_{t,ij} + \frac{1}{f_{\text{obj},l}} w_{\text{type}}$
      end for
    end if
  end for
  if $\sum_j \tau_{t,ij} = 0$ **then**
    Discard solution, **Terminate**
  else
    $s_{(2+2(i-1))} \leftarrow$ SelectProbabilityDistribution($\boldsymbol{\tau}_{t,i}$)
  end if
end for

3. Solution Evaluation

Once a set of plans S has been composed by the ants, each plan has to be evaluated to assess its feasibility and objective value. This is done by calling the trajectory model. If a solution is infeasible at transfer number i, its objective value is set to $f_{\text{obj}} = +\infty$ and the solution is stored in the ith tabu list. If a solution is feasible, instead, it is stored in the feasible list.

D. Comparison with Standard ACO

The way in which the ants generate the solutions in ACO–MGA (or tours, to use ACO nomenclature) is similar to what happens in the TSP with standard ACO [8]: each ant, independently of the others, generates a tour by adding

nodes (or cities) one at a time. Each node is chosen probabilistically among a set of available nodes: for the TSP, the available nodes are the cities which have not been visited in the current tour; for the MGA, nodes are all the possible pairs of bodies and types of transfers. For both frameworks, the pheromone is distributed over all the possible choices, and then a selection is made, according to the pheromone distribution. In the case of standard ACO, the probability associated to each city depends on a heuristic function and on the pheromone deposited along the edge connecting the current city to the next one. ACO–MGA, instead, progressively builds a surrogate model of the feasible and infeasible regions of the search space by saving the feasible and infeasible solutions in the feasible and tabu lists. The decision on which city (planet) to visit next, therefore, is made by interrogating the feasible and tabu lists rather than the model.

The model is interrogated only to evaluate the feasibility and cost of a solution not already in either the feasible or tabu list. In this sense, the evaluation step can be seen as analogous to the pheromone deposition in standard ACO.

On the other hand, in the case of the MGA trajectory model presented in this paper, the pheromone cannot be assigned to individual transfers: this is due to the fact that each transfer (identified by its pair of integers) has no intrinsic value within the plan, if disconnected from the previous transfers. In fact, the actual value of a transfer depends on its initial conditions, which are in turn dependent on all the previous transfers. Therefore, there is a strong dependency of every decision on all previous ones.

To illustrate the dependency problem, we make reference to Figs. 7 and 8a: the former shows a typical instance of the TSP. In this problem, the distance between each pair of cities is fixed, and the relative distances of n cities can be stored in a $n \times n$ matrix [8]. This means that an edge will give the same contribution to the overall length of the tour, regardless of the rest of the tour. For example, Fig. 7 shows two different tours for the given TSP instance:

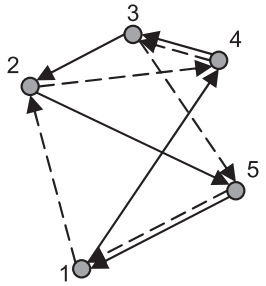

Fig. 7 A five-node instance of the TSP, with two possible solutions identified by continuous and dashed arrows.

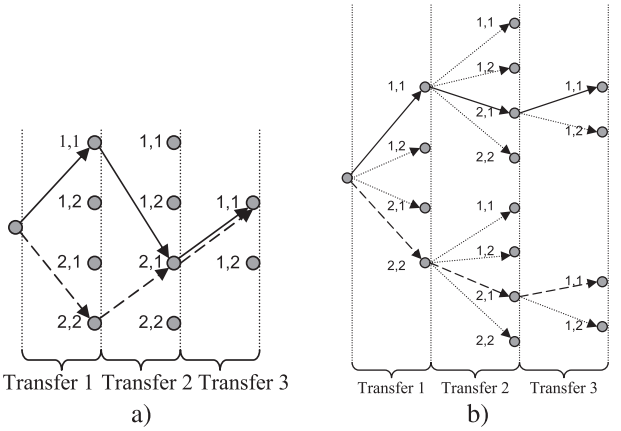

Fig. 8 Two different representations of the MGA problem: a) TSP-like representation of a three-leg MGA problem with two solutions, identified by continuous and dashed arrows; b) expanded tree representation of the same MGA problem.

1–4–3–2–5 (continuous line) and 1–2–4–3–5 (dashed line). The edge 3-4 is shared by both tours and will obviously contribute in the same way to their objective functions, i.e., the total distance covered by the tour. This is not true in the MGA case. Figure 8a is a representation of a simple instance of the MGA problem: it has three transfers, two sets of parameters for each transfer, two planets for the swing-by's, and one target planet. Each node represents a possible planet in combination with a type of transfer. The pairs of numbers next to each node in Fig. 8a are the two integers identifying the transfer in the solution vector (see Sec. III.A). A solution is generated by selecting one node for each transfer, thus generating a tour which connects the starting node to one of the final nodes. The figure represents two possible solutions to the MGA problem: [1, 1, 2, 1, 1, 1] (continuous line) and [2, 2, 2, 1, 1, 1] (dashed line). These two solutions share the same parameters for the last transfer: [1, 1]. This means that they reach the same target planet with the same type of transfer. Because of the dependency of each transfer on the initial conditions, it is not possible to state that the last transfer has the same value for both solutions: in fact, the two trajectories can be consistently different, and lead to different final conditions and objective functions. For this reason, it makes no sense, for example, to assign a value to the set of parameters [1, 1] of Transfer 3 in Fig. 8a; whereas it is possible to assign a value to the edge 3-4 in Fig. 7.

A different representation of the continuous-line solution in Fig. 8a is the one shown in Fig. 8b in which every branch of the tree depends on the previous ones. In Fig. 8b, it is clear that the set of parameters [1, 1] for Transfer 3 belongs to two different solutions.

Note that the dependency problem would affect any method (exact or stochastic) that proceeds incrementally along the graph, evaluating one leg at the time.

IV. Case Studies

The proposed optimization method was applied to two case studies inspired by the BepiColombo [13] and Cassini [19] missions. The two test cases were taken from a previous work by the authors [20] and made more challenging by increasing the number of degrees of freedom. Furthermore, the analysis and comparisons were extended, with respect to Ceriotti and Vasile [20], with new results on the performance of all the tested algorithms.

For both tests, t_0 and φ_0 are preassigned and correspond to the launch date and direction of known optimal solutions. The tests, in fact, aim at assessing the ability of ACO–MGA to efficiently generate a complete plan given a set of initial conditions. ACO–MGA was tested against two implementations of genetic algorithms: the MATLAB® Genetic Algorithm and Direct Search Toolbox (GATBX) [21], and NSGA-II [22]. Settings for all the optimizers will be specified for each test case. While NSGA-II can deal with discrete variables, GATBX operates on real variables only, therefore a wrapper for the objective function was coded to round the continuous solution vector to the closest integer. Due to the stochastic nature of the heuristics used in the tests, all the algorithms were run 100 times. Two performance indexes are used to compare ACO–MGA against the other global optimizers: the percentage of times an algorithm finds feasible solutions, called *feasibility rate* in the following, and the percentage of times the objective value f_{obj} of the feasible solutions is $f_{\text{obj}} < \tilde{f}_{\text{obj}} + \epsilon$, called *admissibility rate* in the following. The value $\tilde{f}_{\text{obj}}$ is the best known objective function for a given problem. According to the theory developed in Vasile et al. [23–25], 100 runs give an error in the determination of the exact rate (admissibility or feasibility) of less than 6% with 92% confidence. This means that two results that differ by less than 12% cannot be said, with 100% confidence, to be different. For the sake of completeness, the mean and variance of the best solution over 100 runs were also reported.

It is important to underline the differences and commonalities between the application of ACO–MGA, NSGA-II, and GTABX to the solution of the test cases presented in this section. All three optimizers are applied to the same instances of the same problems. They all interrogate exactly the same black-box function (the trajectory model in Sec. II), operate on exactly the same solution vector (the vector $\mathbf{s}$ in Fig. 6), and explore exactly the same search space. However, while ACO–MGA builds $\mathbf{s}$ incrementally, both NSGA-II and GATBX assign a value to all the components of $\mathbf{s}$ simultaneously. Furthermore, ACO–MGA interrogates both the feasible and tabu lists before calling the model, if necessary, to fill in the components of $\mathbf{s}$, whereas NSGA-II and GATBX only call the model to decide whether to retain or reject an individual.

Some preliminary tests showed that the best performance of ACO–MGA is achieved if the algorithm is run in two steps, using different sets of parameters. In particular, in the first step the weights w_{planet}, w_{type} are set to 0. Remembering Eqs. (37) and (39), this choice translates into an initial pure random search. In fact, the solutions in the feasible list do not alter the pheromone distribution. On the other hand, the pheromone of tabu partial solutions

is still set to zero to avoid their reexploration. In the second step, weights are set to nonnull values to intensify the exploration around known feasible solutions. The values of w_{planet} and w_{type} are chosen such that:

$$w_{\text{planet}}, w_{\text{type}} = \overline{w} \cdot \hat{f}_{\text{obj}} \tag{40}$$

where $\hat{f}_{\text{obj}}$ is the expected minimum value for the objective function. In this way, by choosing for example $\overline{w} = 1$, a 1 is added to the pheromone of a given element every time a matching solution with objective $\hat{f}_{\text{obj}}$ appears in the feasible list. The value of the added pheromone is higher if the objective value of the matching feasible solution is lower than $\hat{f}_{\text{obj}}$.

This two-step procedure can be explained in the following way. The first step allows a random sampling of the solution space, with the aim of finding a good number of feasible solutions. This is done to prevent the algorithm stagnating around the first feasible solution found. The second step intensifies the search around the feasible solutions which were found in the first step. Because of Eqs. (37) and (39), feasible solutions with low objective values are likely to be investigated further. In addition, the random component in the process does not forbid the exploration of the rest of the search space.

All the tests were run on an Intel® Core™ 2 Quad Q9650 (3 GHz) machine running Microsoft® Windows® Vista, without using any multitasking.

A. BepiColombo Case Study

In this mission, the spacecraft departs from Earth on 15 August 2013 ($t_0 = 4974.5$ MJD2000) to reach a scientific orbit around Mercury with a minimum relative arrival velocity v_∞. The magnitude of the DSMs is assumed to be limited and can only be one of the values in $\mathcal{Q}_1$. The relative arrival velocity is instead free and needs to be minimized to have acceptable transfers. As such it was decided to include only v_∞ in the objective function for this problem. The launch date was set to match the one of the ESA chemical option for BepiColombo [26]. Four transfers (and thus three swing-by's) are considered for the planning problem, with the launch angle set to $\varphi_0 = \pi$. For the first and second transfer, the following sets of values were used:

$$\begin{aligned}
\mathcal{Q}_P &= \{\text{Mercury, Venus, Earth}\}\\
\mathcal{Q}_1 &= \{0\}\\
\mathcal{Q}_2 &= \{\emptyset\}\\
\mathcal{Q}_3 &= \{0, 1, 2, 3, 4\}\\
\mathcal{Q}_4 &= \{\emptyset\}\\
\mathcal{Q}_5 &= \{0, 1\}
\end{aligned}$$

Since there is no DSM, the sets $\mathcal{Q}_2$ and $\mathcal{Q}_4$ are empty. In general, there is no easy way to identify whether the first or the second orbital intersection is the best one, thus $\mathcal{Q}_5$ has cardinality 2. For the third leg, the following sets of values were used:

$$\begin{aligned}
\mathcal{Q}_P &= \{\text{Mercury}\}\\
\mathcal{Q}_1 &= \{-50, 0, 50\}\ \text{m/s}\\
\mathcal{Q}_2 &= \{0\}\\
\mathcal{Q}_3 &= \{0, 1, 2, 3, 4\}\\
\mathcal{Q}_4 &= \{0, 1\}\\
\mathcal{Q}_5 &= \{0, 1\}
\end{aligned}$$

In this case, a DSM can be exploited to reach Mercury with a minimum v_∞. The fourth and last leg is a Mercury resonant swing-by. Here, the DSM is particularly important to change the relative velocity, therefore a wider set of

magnitudes were adopted:

$$\mathcal{Q}_P = \{\text{Mercury}\}$$
$$\mathcal{Q}_1 = \{-100, -50, 0, 50, 100\} \text{ m/s}$$
$$\mathcal{Q}_2 = \{0\}$$
$$\mathcal{Q}_3 = \{0, 1, 2, 3, 4\}$$
$$\mathcal{Q}_4 = \{0, 1\}$$
$$\mathcal{Q}_5 = \{0, 1\}$$

The modulus of the departure excess velocity v_0 is constrained to be between 2 and 4 km/s, which implies the following set of starting guess points for the Brent's method: [2, 2.5, 3, 3.5, 4] km/s. The following set of starting points for r_p was used instead for both Venus and Mercury: $[0.9,\ 0.92,\ 0.94, \ldots,\ 5]R_P$; and $r_{ps} = [-r_p,\ +r_p]$. Note that swing-by's with radius of pericentre lower than R_P are not physically feasible: there are two reasons which motivated this choice. The first is that due to the fast dynamics of the inner part of the solar system, a higher number of feasible solutions are found if we consider lower radii of pericenter. The second is that solutions with radii of pericenter within the extended range can still be reoptimized with a complete model, and the proper constraint $r_p > R_P$. The total time of flight was limited to a maximum of 10 years, with the objective function set to the v_∞ at Mercury. Based on experience and similar previous missions, we define admissible solutions as those solutions whose objective value is below 6 km/s.

With the sets of values presented above, the average time for the evaluation of one plan is 0.64 ms, and there exist 5,400,000 distinct possible plans. Thus, a systematic scan of all the possibilities would require about 3456 s (57.6 h).

ACO–MGA always used 10 ants, and was tuned with the following weights: w_{planet}, $w_{\text{type}} = 0$ for the first step, followed by a second step with w_{planet}, $w_{\text{type}} = 20\hat{f}_{\text{obj}}$ and $\hat{f}_{\text{obj}} = 3$ km/s. However, because of the normalization in Eq. (40), the weight values appear to have general validity and can be applied to other transfer problems, as will be shown in the next case study. The algorithm was run for an increasing number of function evaluations (500, 1000, 2000 function evaluations) until the feasibility rate reached 100% and the admissibility rate was over 90%. For 500 function evaluations, the first step was limited to 50 iterations, whereas the second step was limited to 125 iterations, which is enough to reach the required maximum number of function evaluations. For the tests with higher number of function evaluations, the number of iterations was increased proportionally, such that $n_{\text{eval}}/n_{\text{iter}} = \text{constant}$.

The performance indexes for 500, 1000, and 2000 function evaluations are presented in Fig. 9. It is worth noting that even for 500 evaluations, the feasibility rate of ACO–MGA is 100% with an admissibility rate of 45%, i.e., all runs are feasible and one out of two is admissible. For 2000 function evaluations, the admissibility rate increases up to 95%, therefore $n_{\text{eval}} = 2000$ evaluations will be used as reference value for this problem.

Figure 10 shows the minimum, mean, standard deviation, and maximum of the best solution over 100 runs of ACO-MGA using 2000 function evaluations, as a function of the iteration number. Each run stops after about 450 iterations, as at that point it reaches 2000 function evaluations. Figure 10a shows that after about 350 iterations, the number of admissible runs reaches 95%. Nevertheless, Fig. 10b shows that in the last 50 iterations, the standard deviation of the feasible runs decreases dramatically, meaning that the last few iterations are used to converge locally but the basin of attraction of the admissible solutions is identified earlier on for a lower number of iterations. Note also the change in the slope of the admissibility rate and the mean of the best solution after about 200 iterations: this is the point in which ACO–MGA switches from the first step to the second step, changing the weights in Eq. (40), and favoring a local search.

Since GATBX and NSGA-II are all-purpose optimizers that work with any black-box problem, a tuning of the main parameters of these optimizers was performed before comparing them to ACO–MGA. This is done to ensure that they achieve the best performances on this specific problem. The tuning was performed by running the optimizers with different settings. For each setting, the optimizer was run 100 times and feasibility rate, admissibility rate, mean, and variance of the best solution were computed. For each run, the optimization was stopped after 2000 function evaluations.

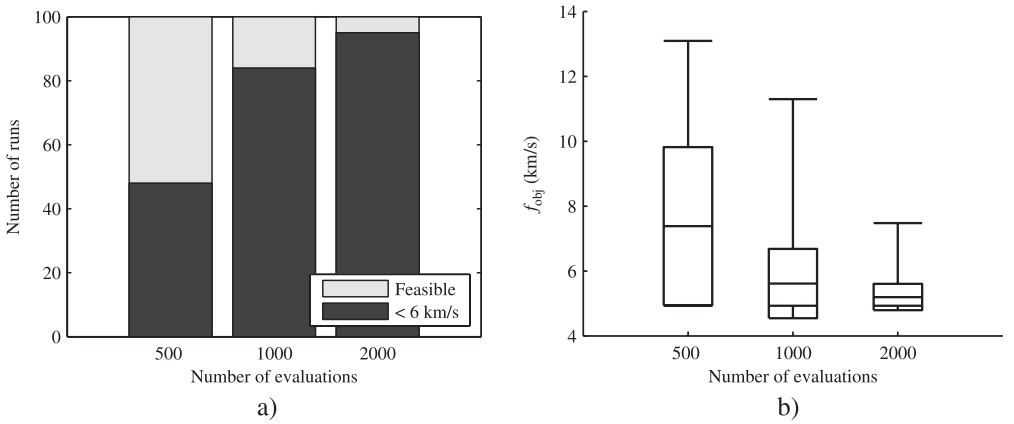

Fig. 9 Performance indexes of ACO–MGA on the BepiColombo problem, over 100 runs, for different number of function evaluations. a) Number of runs returning an admissible (<6 km/s) or feasible solution. b) Minimum, mean, standard deviation, and maximum of the best solution over the runs that returned a feasible solution.

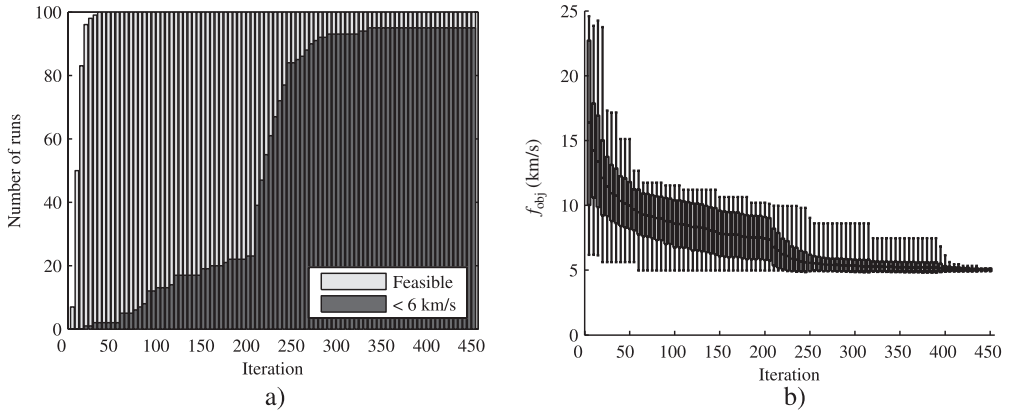

Fig. 10 Performance indexes of ACO-MGA over 100 runs for 2000 function evaluations. a) Number of runs returning an admissible (<6 km/s) or feasible solution. b) Minimum, mean, standard deviation, and maximum of the best solution over the runs that returned a feasible solution.

The size of the population n_{pop} was tuned for both NSGA-II and GATBX. In addition, we tuned the parameters *pcross_bin* $\in (0, 1)$ and *pmut_bin* $\in (0, 1)$ (whose default value is 0.5 for both) for NSGA-II and parameters *CrossoverFraction* $\in (0, 1)$ and *MigrationFraction* $\in (0, 1)$ (whose default values are 0.8 and 0.2, respectively) for GATBX. These parameters are the ones that resulted to have the most influence on the final outcome of an optimization.

It was assumed that each parameter could only take values from a predefined set. The total number of settings is given by the Cartesian product of all the sets of all the parameters. Hence, each setting corresponds to a possible combination of values for each parameter. For NSGA-II, the sets were:

$$n_{\text{pop}} \in \{12, 20, 40, 80\}$$

$$pcross\_bin \in \{0.25, 0.5, 0.75\}$$

$$pmut\_bin \in \{0.25, 0.5, 0.75\}$$

These sets of parameters generate 36 different settings: for example, the first setting is obtained by taking the first parameter in each set; the second setting is obtained by taking $n_{\text{pop}} = 12$, $pcross\_bin = 0.25$, and $pmut\_bin = 0.5$;

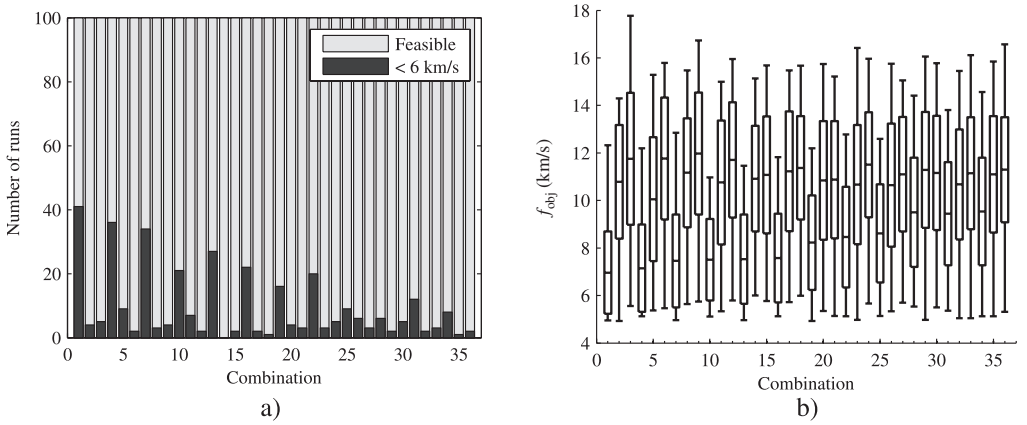

Fig. 11 NSGA-II tuning on the BepiColombo problem. Results over 100 runs for each setting. a) Number of runs returning an admissible (<6 km/s) or feasible solution. b) Minimum, mean, standard deviation, and maximum of the best solution over the runs that returned a feasible solution.

and so on. Analogously, for GATBX, the sets are:

$$n_{\text{pop}} \in \{12, 20, 40, 80\}$$

$$\mathit{CrossoverFraction} \in \{0.2, 0.5, 0.8\}$$

$$\mathit{MigrationFraction} \in \{0.2, 0.5, 0.8\}$$

The number of generations n_{gen} was set such that $n_{\text{pop}} \cdot n_{\text{gen}} = n_{\text{eval}}$. Figure 11 shows the performances of each setting over 100 runs. Figure 11a shows the number of runs that produced feasible and admissible solutions, whereas Fig. 11b shows minimum, mean, standard deviation, and maximum of the best solution over the feasible runs.

The combinations that produced the best results are 1, 4, and 7. Table 2 shows the parameters for these combinations and the corresponding performance indexes. Combination 1 led to the lowest mean and the highest number of admissible solutions, therefore it was considered to be optimal for this problem.

The results of the tuning of GATBX are reported in Fig. 12. In this case, the combinations that returned the highest number of admissible solutions are 31, 32, and 36 (see Table 3). Combination 31 has a lower mean, but a lower value of admissible solutions with respect to Combination 32, whereas Combination 36 has a lower number of admissible solutions with respect to the other two. The number of admissible solutions is assumed here to be the most significant performance index (see Vasile et al. [23–25] for a technical justification of this choice), for the selection of the most appropriate setting. Therefore, Combination 32 was considered to be optimal for GATBX. Note that the best combination for GATBX has a large population, whereas NSGA-II resulted to work better with a small population (and more iterations).

Table 2 Three combinations of settings which provided the highest percentage of admissible solutions for NSGA-II

Combination no.	1	4	7
n_{pop}	12	12	12
pcross_bin	0.25	0.5	0.75
pmut_bin	0.25	0.25	0.25
$n_{\text{eval}}/n_{\text{pop}}$	166.67	166.67	166.67
% admissible (<6 km/s)	41	36	34
% feasible	100	100	100
Mean (km/s)	6.9629	7.1451	7.4541
Standard deviation (km/s)	1.7349	1.8367	1.9501

Table 3 The three settings which provided the highest percentage of admissible solutions for GATBX

Combination no.	32	31	36
n_{pop}	80	80	80
CrossoverFraction	0.5	0.5	0.8
MutationFraction	0.5	0.2	0.8
$n_{\text{eval}}/n_{\text{pop}}$	25	25	25
% admissible (<6 km/s)	19	18	16
% feasible	82	82	90
Mean (km/s)	9.1658	9.1371	9.6991
Standard deviation (km/s)	3.0166	2.7518	3.2726

Figures 13 and 14 show the performance indexes over 100 runs of NSGA-II and GATBX, respectively, for an increasing number of function evaluations. Since the performance indexes of both optimizers were relatively poor compared to ACO–MGA using 2000 evaluations, the tests were repeated extending the number of function evaluations

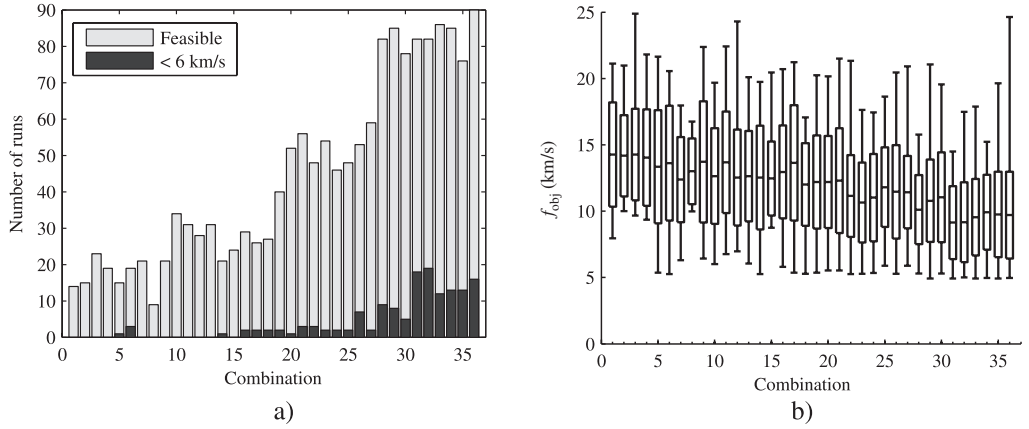

Fig. 12 GATBX tuning on the BepiColombo problem. Results over 100 runs for each combination of the parameters of the optimizer. a) Number of runs returning an admissible (<6 km/s) or feasible solution. b) Minimum, mean, standard deviation, and maximum of the best solution over the runs that returned a feasible solution.

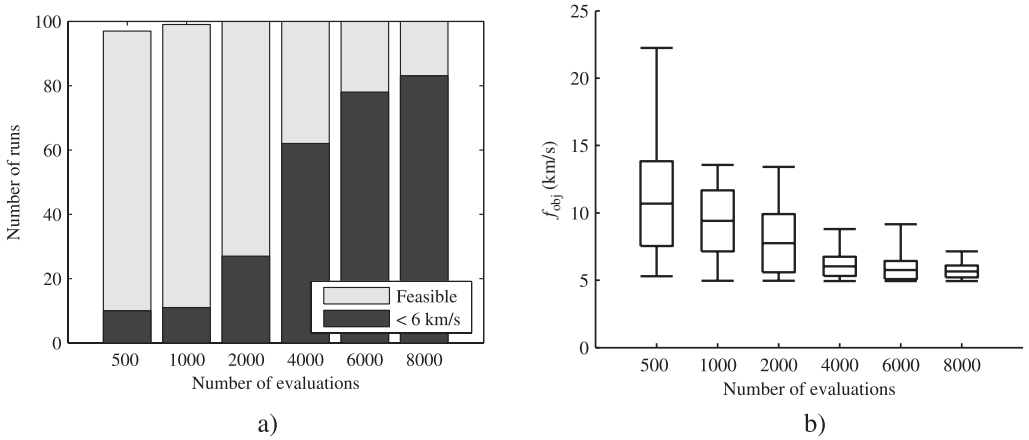

Fig. 13 Performance of NSGA-II on the BepiColombo problem, over 100 runs, for different number of function evaluations. a) Number of runs returning an admissible (<6 km/s) or feasible solution. b) Minimum, mean, standard deviation, and maximum of the best solution over the runs that returned a feasible solution.

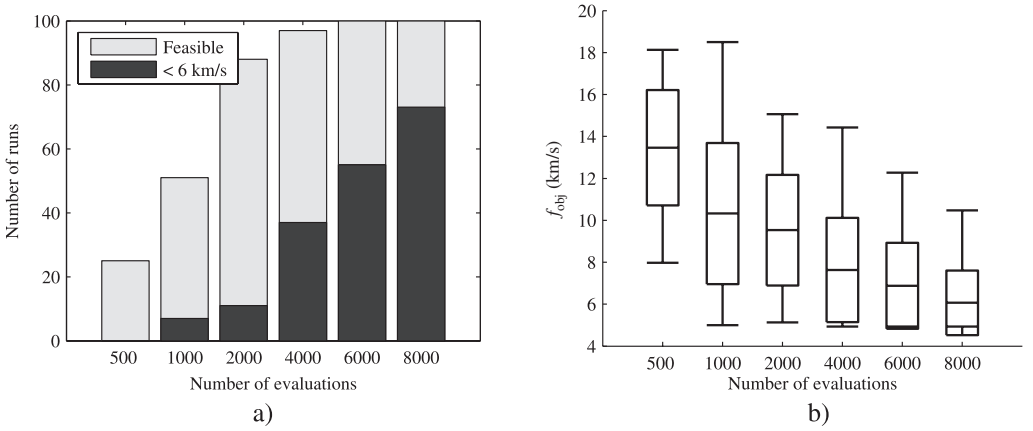

Fig. 14 Performance of GATBX on the BepiColombo problem, over 100 runs, for different number of function evaluations. a) Number of runs returning an admissible (<6 km/s) or feasible solution. b) Minimum, mean, standard deviation, and maximum of the best solution over the runs that returned a feasible solution.

up to 8000. Nonetheless, neither optimizer could reach the performance of ACO–MGA, even with 8000 function evaluations.

Figure 15 shows a plot of the best solution found by ACO–MGA for this problem. The sequence for this solution is EVVMe. The solution has an objective value (relative velocity at Mercury) of $f_{\text{obj}} = v_\infty = 4.8275$ km/s with a time of flight of 4.8275 years, and a departure velocity $v_0 = 3.6293$ km/s. The parameters for this solution can be found in Table 4. As a comparison, the solution chosen as chemical baseline at ESA/ESOC [27] departs with an excess velocity of 3.794 km/s, and the velocity relative to Mercury at the second swing-by is 5.472 km/s. The trajectory exploits only one DSM of 45 m/s. This trajectory, however, is computed using a three-dimensional model.

The solutions presented so far were found by fixing the departure time t_0. In order to find the optimal launch date, ACO-MGA can be reiterated for different t_0. An example is shown in Fig. 16a, where the feasibility and admissibility rates ($f_{\text{obj}} < 6$ km/s) for each launch date are shown. Figure 16b instead represents the average of the best solutions found over 100 runs, as a function of t_0. Note that in the given range of t_0, the optimal sequence does not change.

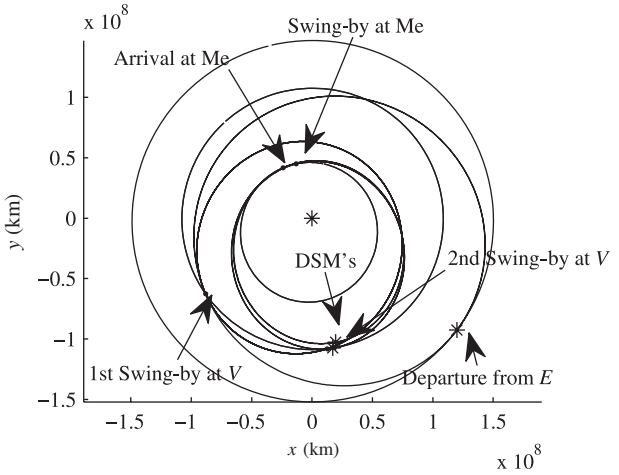

Fig. 15 Trajectory of the best solution found by ACO–MGA for the BepiColombo problem.

Table 4 Parameters of the best solution found by ACO–MGA for the BepiColombo case study

Parameter	Leg 1	Leg 2	Leg 3	Leg 4
Planet	V	V	Me	Me
m_{DSM} (m/s)	0	0	−50	100
n_{rev1}	0	0	0	0
n_{rev2}	1	4	2	1
$f_{p/a}$	0	0	1	1
$f_{1/2}$	0	1	1	1

Figure 16a reveals that for some launch dates, the probability of finding a good solution is quite low. One reason is that the region of the search space containing the best solution is particularly narrow. This suggests that for those dates, small variations of the departure time would result in a steep increase of the mission cost.

B. Cassini Case Study

Cassini is the ESA–NASA mission to Saturn. The planetary sequence chosen for this mission, Earth–Venus–Venus–Earth–Jupiter–Saturn, allowed a substantial reduction of the total required Δv to reach Saturn. For testing

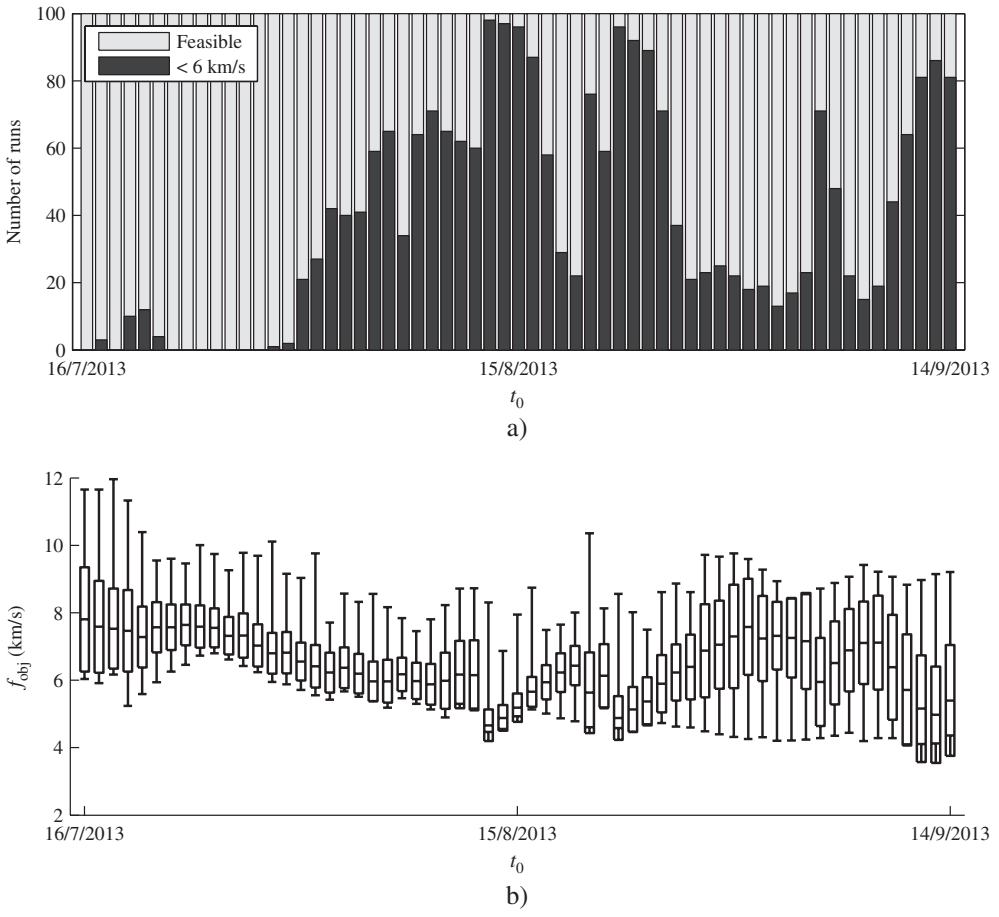

Fig. 16 BepiColombo Launch window: 100 runs for each launch date. a) Number of runs returning an admissible (<6 km/s) or feasible solution. b) Minimum, mean, standard deviation, and maximum of the best solution over the runs that returned a feasible solution.

ACO–MGA, the launch date was set to $t_0 = -779$ MJD2000, corresponding to 13 November 1997, and the following sets of values were used for the first three transfers:

$$\mathcal{Q}_P = \{\text{Venus, Earth, Mars, Jupiter}\}$$

$$\mathcal{Q}_1 = \{-600, -350, -200, 0, 200, 350, 600\}\ \text{m/s}$$

$$\mathcal{Q}_2 = \{0\}$$

$$\mathcal{Q}_3 = \{0\}$$

$$\mathcal{Q}_4 = \{0, 1\}$$

$$\mathcal{Q}_5 = \{0, 1\}$$

For the forth transfer, no DSM is allowed:

$$\mathcal{Q}_P = \{\text{Venus, Earth, Mars, Jupiter}\}$$

$$\mathcal{Q}_1 = \{0\}\ \text{km/s}$$

$$\mathcal{Q}_2 = \{\emptyset\}$$

$$\mathcal{Q}_3 = \{0\}$$

$$\mathcal{Q}_4 = \{\emptyset\}$$

$$\mathcal{Q}_5 = \{0, 1\}$$

And finally, for the last transfer, the Saturn has to be targeted:

$$\mathcal{Q}_P = \{\text{Saturn}\}$$

$$\mathcal{Q}_1 = \{0\}$$

$$\mathcal{Q}_2 = \{\emptyset\}$$

$$\mathcal{Q}_3 = \{0\}$$

$$\mathcal{Q}_4 = \{\emptyset\}$$

$$\mathcal{Q}_5 = \{0, 1\}$$

The maximum number of full revolutions was set to 0 to limit the total time of flight of the mission. Since the trajectory is going outwards from the orbit of the Earth, every full revolution implies more than one additional year in transfer time. The total number of distinct solutions for this test is 22,478,848 and the average time to evaluate a solution is 0.39 ms. This translates into 8765 s (or about 146 h) to systematically evaluate all the solutions.

As for BepiColombo, the launch excess velocity module was bounded between 2 and 4 km/s. For the swing-by's of Earth and Venus, the radii of pericenter are $[1.1,\ 1.2,\ 1.3,\ \ldots,\ 5]R_P$ while a different choice was adopted for Jupiter. Since, the mass of this planet is considerably higher than the masses of Venus or Earth, higher radii of pericenter are sufficient to achieve considerable deviations. Furthermore, since the function $\Delta\theta(r_{ps})$ is smooth in this case, the first guesses are spaced with a five Jupiter radii step size: $[5,\ 10,\ 15,\ \ldots,\ 100]R_P$.

Regarding the choice of the objective function, it has to be noted that for all the missions to outer planets, the time of flight becomes very important, as very long missions are needed to reach farther destinations. Even limiting the number of complete revolutions to zero is not enough to guarantee a mission with reasonable duration. Therefore, it is important to include the total time of flight T in the objective function, in addition to the total Δv. Since the current

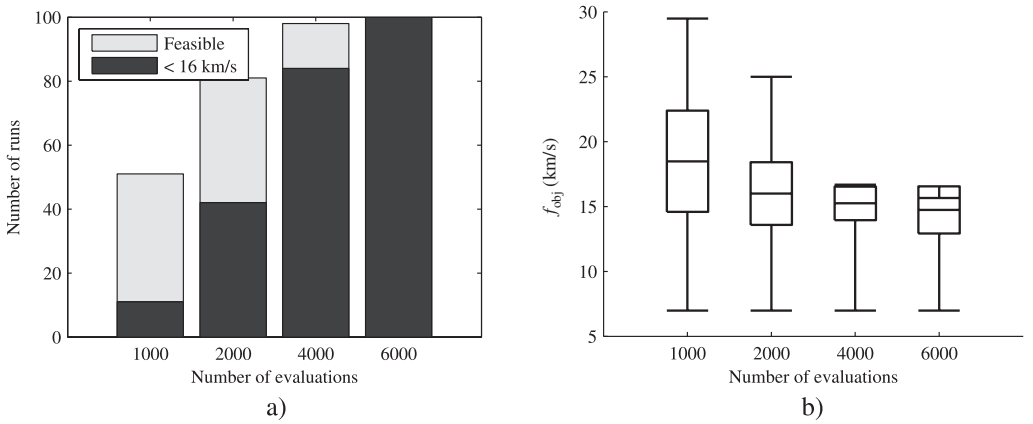

Fig. 17 Performance of ACO-MGA on the Cassini problem, over 100 runs, for different number of function evaluations. a) Number of runs returning an admissible (<16 km/s) or feasible solution. b) Minimum, mean, standard deviation. and maximum of the best solution over the runs that returned a feasible solution.

algorithm cannot deal with multiobjective optimization, the total time of flight and the v_∞ were weighed inside the objective function in the following way:

$$f_{\text{obj}} = v_\infty + \sigma T \tag{41}$$

with the weight $\sigma = 1/1000$ km/s/d.

The total time of flight was limited to a maximum of 100 years. This bound may seem to be too high since a realistic time span of a transfer to Saturn is around 10 years, however the model considers all the solutions longer than the specified time of flight threshold infeasible, and the optimizer saves them as tabu. Therefore, limiting the time of flight to lower values would over-constrain the search for optimal solutions. Better results are obtained by allowing long solutions to be returned as feasible, and introducing their duration into the objective function. The admissibility threshold was set to 16 km/s.

For this case study, a procedure similar to the one presented in the previous section was followed. ACO–MGA used the same weights as for BepiColombo, but since the search space is larger, the algorithm was run for 1000, 2000, 4000, and 6000 function evaluations at which point more than 95% of the runs returned admissible solutions. At 6000 function evaluations, the number of iterations for the first and second step was set to 1000. For other numbers of function evaluations, the number of iterations was modified proportionally. The performance indexes are presented in Fig. 17. Note that ACO–MGA is able to identify, already for 1000 function evaluations, a solution that has an objective value of 6.9 km/s, despite the mean value of the best solutions is about 15 km/s. This solution is particularly difficult to find in the search space, and more than 6000 evaluations would be necessary to ensure that ACO–MGA hits the threshold value with over 95% probability.

The behavior of ACO–MGA at 6000 function evaluations is shown in Fig. 18. Figure 10a shows that after about 1600 iterations, the number of admissible runs reaches 95%. Figure 10b shows that in the last 400 iterations, the standard deviation of the feasible runs decreases dramatically, meaning that the last few iterations are used to converge locally but the basin of attraction of the admissible solutions is identified earlier on for a lower number of iterations. Note also the change in slope of the admissibility rate and the mean of the best solution after about 1000 iterations: this is the point in which ACO–MGA switches from the first step to the second step, changing the weights in Eq. (40), and favoring local search.

NSGA-II and GATBX were fine-tuned again on this problem following the same procedure presented in Sec. IV.A. In this case, the reference number of function evaluations for the tuning is 6000. The set of parameters for the two optimizers are the same as in Sec. IV.A, except for the range of the population size, which was increased as the

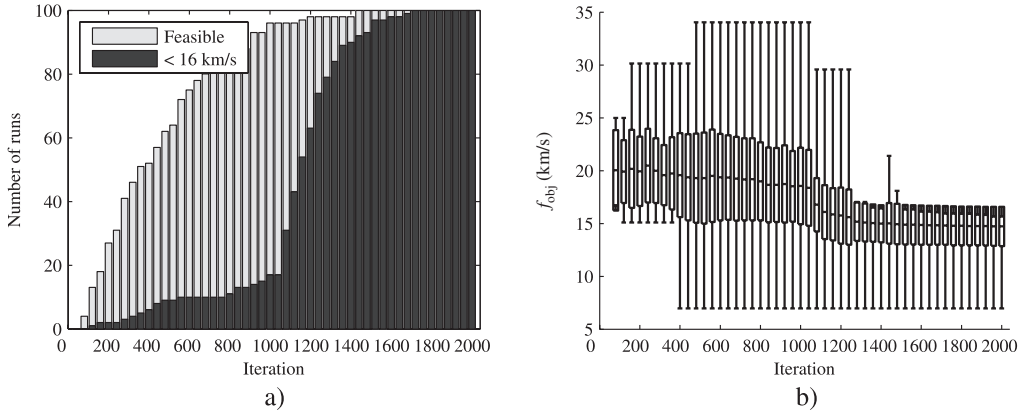

Fig. 18 Performance indexes over 100 runs, for to 6000 function evaluations. a) Number of runs returning an admissible (<16 km/s) or feasible solution. b) Minimum, mean, standard deviation, and maximum of the best solution over the runs that returned a feasible solution.

problem is more difficult. For NSGA-II

$$n_{\text{pop}} \in \{20, 60, 100, 200\}$$
$$pcross\_bin \in \{0.25, 0.5, 0.75\}$$
$$pmut\_bin \in \{0.25, 0.5, 0.75\}$$

For GATBX:

$$n_{\text{pop}} \in \{20, 60, 100, 200\}$$
$$CrossoverFraction \in \{0.2, 0.5, 0.8\}$$
$$MigrationFraction \in \{0.2, 0.5, 0.8\}$$

Figure 19 shows the performances indexes of each setting.

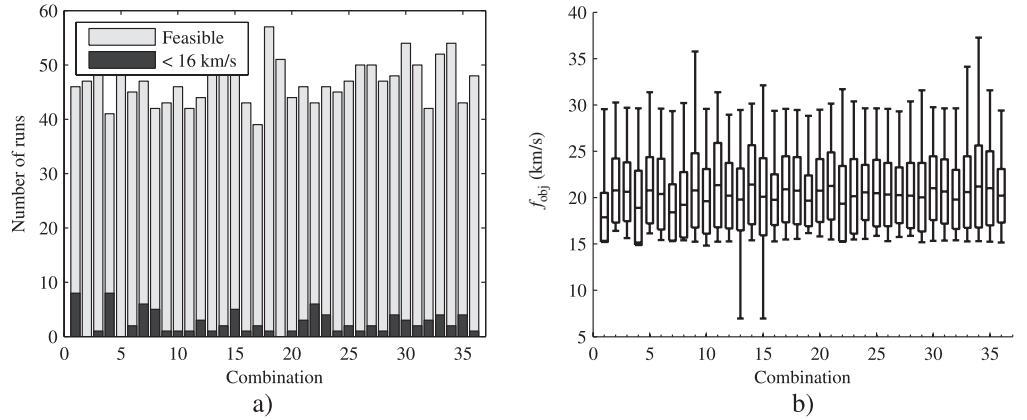

Fig. 19 NSGA-II tuning on the Cassini problem. Results of 100 runs for each combination of the parameters of the optimizer. a) Number of runs returning an admissible (<16 km/s) or feasible solution. b) Minimum, mean, standard deviation, and maximum of the best solution over the runs that returned a feasible solution.

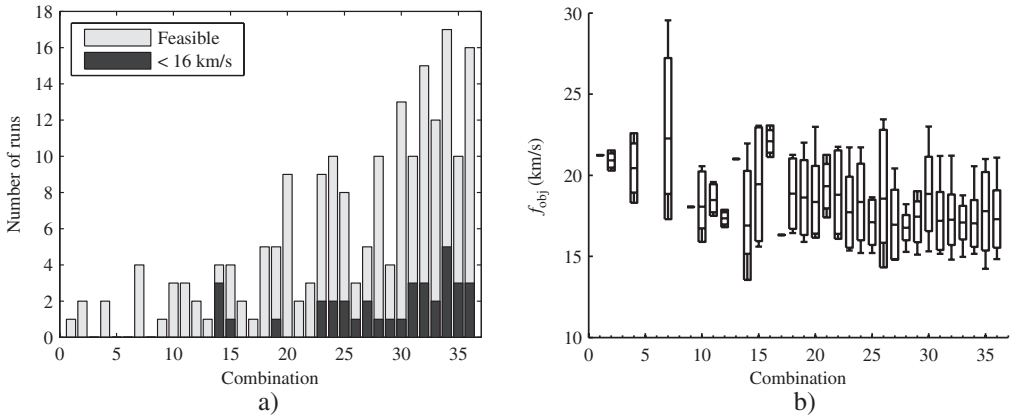

Fig. 20 GATBX tuning on the Cassini problem. Results of 100 runs for each combination of the parameters of the optimizer. a) Number of runs returning an admissible (<16 km/s) or feasible solution. b) Minimum, mean, standard deviation, and maximum of the best solution over the runs that returned a feasible solution.

The combinations with the best performance indexes are 1, 4, and 7. Table 5 shows the parameters for these combinations and the corresponding statistical figures over 100 runs. In this case, Combinations 1 and 4 led to the same admissibility, but 1 has a higher feasibility rate, therefore this combination was chosen as the best tuning of NSGA-II.

The results of the tuning of GATBX are reported in Fig. 20. In this case, the combinations that yielded the best performance indexes are 32, 34, and 36 (see Table 6). Combination 34 has the highest admissibility and feasibility, and hence it was selected as the optimal tuning for GATBX.

Table 5 The four settings which provided the highest percentage of admissible solutions for NSGA-II

Combination no.	1	4	7	22
n_{pop}	20	20	20	100
pcross_bin	0.25	0.5	0.75	0.5
pmut_bin	0.25	0.25	0.25	0.25
$n_{\text{eval}}/n_{\text{pop}}$	300	300	300	60
% admissible (<16 km/s)	8	8	6	6
% feasible	46	41	47	43
Mean (km/s)	17.8707	18.8932	18.4111	19.3408
Standard deviation (km/s)	2.6524	4.0020	3.0397	4.0692

Table 6 The three combinations of settings which provided the highest percentage of admissible solutions for GATBX

Combination no.	34	36	32
n_{pop}	200	200	200
CrossoverFraction	0.8	0.8	0.5
MutationFraction	0.2	0.8	0.5
$n_{\text{eval}}/n_{\text{pop}}$	30	30	30
% admissible (<16 km/s)	5	3	3
% feasible	17	16	15
Mean (km/s)	17.0286	17.2979	17.2576
Standard deviation (km/s)	1.4476	1.7689	1.5494

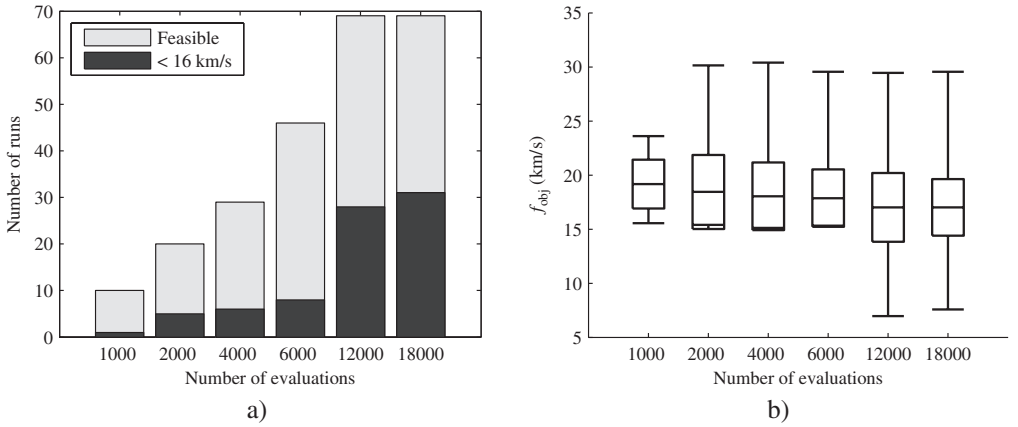

Fig. 21 Performance of NSGA-II on the Cassini problem, over 100 runs, for different number of function evaluations. a) Number of runs returning an admissible (<16 km/s) or feasible solution. b) Minimum, mean, standard deviation, and maximum of the best solution over the runs that returned a feasible solution.

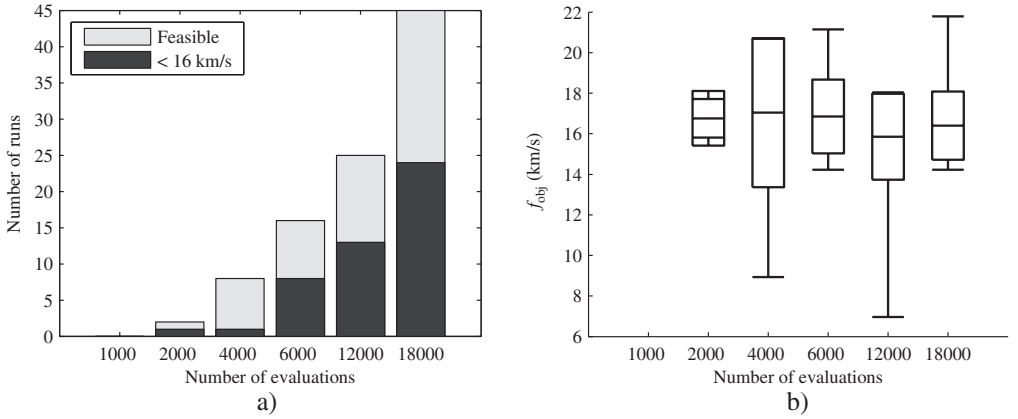

Fig. 22 Performance of GATBX on the Cassini problem, over 100 runs, for different number of function evaluations. a) Number of runs returning an admissible (<16 km/s) or feasible solution. b) Minimum, mean, standard deviation, and maximum of the best solution over the runs that returned a feasible solution.

Figures 21 and 22 show the results of 100 runs of NSGA-II and GATBX, respectively, for an increasing number of function evaluations. Since the performance of both the optimizers was relatively poor at 6000 evaluations, the tests were extended up to 18,000 evaluations. Even for this test case, ACO–MGA outperforms the other general-purpose optimizers, using far less function evaluations. Note that NSGA-II and GATBX are able to find the best-known solution only for 12,000 function evaluations or more.

The parameters associated to the best found solution are shown in Table 7, and the trajectory is shown in Fig. 23a. As a comparison, in Table 8 and Fig. 23b, we report the the best solution known so far with a complete 3D model.[‡]

Table 8 and Fig. 23b demonstrate that although the model in Sec. II is only a bi-dimensional, low-fidelity approximation of an MGA trajectory, it is accurate enough to correctly identify optimal MGA transfers and to provide a good estimation of their cost.

The sequence EVVEJS is not the only one that ACO–MGA found for this problem. All other feasible sequences that were found during the optimization process are reported in Fig. 24 together with their objective value. Note that Mars appears only in one sequence.

[‡] `http://www.esa.int/gsp/ACT/inf/op/globopt/edvdvdedjds.htm`, last accessed 10 June 2010

Table 7 Parameters of the best solution found by ACO–MGA for the Cassini case study

	Leg 1	Leg 2	Leg 3	Leg 4	Leg 5
Planet	V	V	E	J	S
m_{DSM} (m/s)	600	−350	0	0	0
n_{rev1}	0	0	0	0	0
n_{rev2}	0	0	0	0	0
$f_{p/a}$	0	1	0	0	0
$f_{1/2}$	0	1	0	0	1

Table 8 Comparison between the best solution found by ACO–MGA and the best-known solution for this trajectory

Variable	ACO–MGA	Best known
v_0 (km/s)	3.139	3.266
Δv_1 (m/s)	600	473
Δv_2 (m/s)	350	398
Δv_3 (m/s)	0	0
Δv_4 (m/s)	0	0
Δv_5 (m/s)	0	0
v_∞ (km/s)	4.216	4.247
T_1 (d)	168.18	167.36
T_2 (d)	423.68	424.09
T_3 (d)	53.00	53.31
T_4 (d)	596.37	589.74
T_5 (d)	2290.27	2199.97
α_1	0.83	0.77
α_2	0.52	0.53
α_3	0.16	0.35
α_4	0.02	0.10
α_5	0.13	0.48
$r_{p,1}$	1.61	1.36
$r_{p,2}$	1.25	1.05
$r_{p,3}$	1.32	1.31
$r_{p,4}$	68.3	71.38

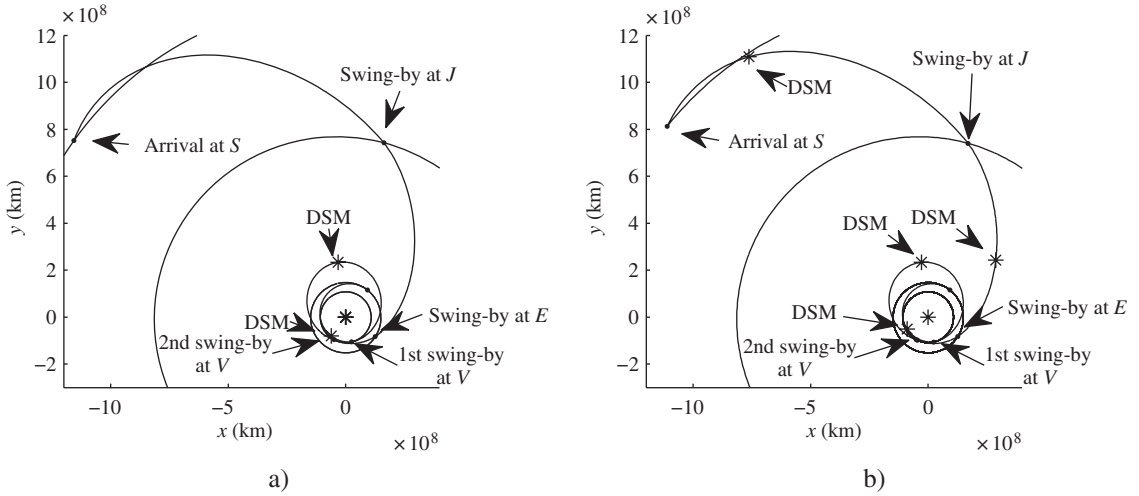

Fig. 23 Solution to the Cassini problem: a) 2D solution from ACO–MGA, b) Cassini best-known solution.

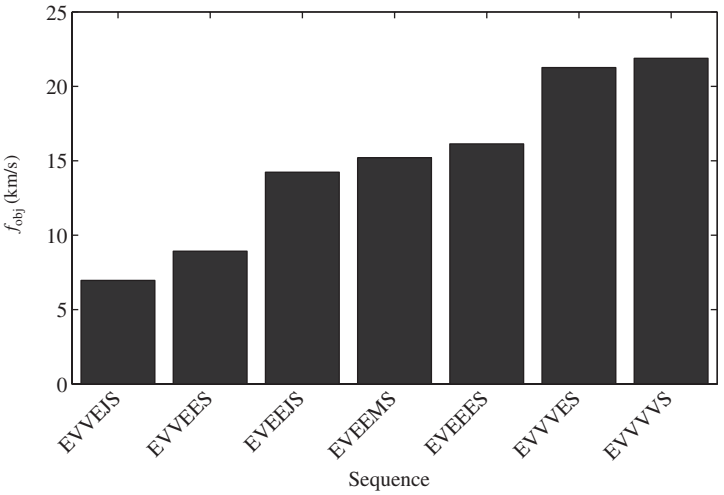

Fig. 24 Classification of the transfer sequences for the Cassini test case.

V. Conclusions

The paper introduced a novel formulation of the automatic trajectory planning problem and proposed an algorithm (ACO–MGA), based on the ant colony paradigm, to generate optimal plans. Each plan is then translated into a complete optimal trajectory made of a scheduled sequence of events. A specific model was developed to efficiently generate families of scheduled trajectories for MGA transfers.

The 2D trajectory model proved to be accurate enough to closely reproduce known MGA transfers even with moderate inclinations. Furthermore, the scheduling of the trajectories was shown to be fast and reliable, allowing for the evaluations of thousands of plans in a few seconds.

The planning algorithm, ACO–MGA, operates an effective search in the finite space of possible plans. The algorithm demonstrated the remarkable ability to find good solutions with a very high success rate, outperforming known implementations of genetic algorithms.

As ACO–MGA requires very little information on the MGA problem under investigation, it represents a valuable tool for the complete automatic design of future space missions. Furthermore, the proposed use of tabu lists appears to be an effective solution to those planning problems in which the value of one segment of the plan depends on all the preceding segments.

Future work aims at a more efficient handling of the lists, which is currently the major bottleneck of the ACO–MGA implementation.

References

[1] Petropoulos, A. E., Longuski, J. M., and Bonfiglio, E. P., "Trajectories to Jupiter via Gravity Assists from Venus, Earth, and Mars," *Journal of Spacecraft and Rockets*, Vol. 37, No. 6, 2000, pp. 776–783. doi: 10.2514/2.3650

[2] Labunsky, A. V., Papkov, O. V., and Sukhanov, K. G., *Multiple Gravity Assist Interplanetary Trajectories*, Earth Space Institute Book Series, Gordon and Breach Science Publishers, 1998, pp. 33–49.

[3] Battin, R. H., *An Introduction to the Mathematics and Methods of Astrodynamics*, AIAA Education Series, AIAA, New York, revised ed., 1999, pp. 419–436.

[4] Vasile, M., and De Pascale, P., "Preliminary Design of Multiple Gravity-Assist Trajectories," *Journal of Spacecraft and Rockets*, Vol. 43, No. 4, 2006, pp. 794–805. doi: 10.2514/1.17413

[5] Ross, I. M., and D'Souza, C. N., "Hybrid Optimal Control Framework for Mission Planning," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 4, 2005, pp. 686–697. doi: 10.2514/1.8285

[6] Stryk, O. V., and Glocker, M., "Decomposition of Mixed-Integer Optimal Control Problems Using Branch and Bound and Sparse Direct Collocation," *ADPM 2000 The 4th International Conference on Automation of Mixed Processes: Hybrid Dynamic Systems*, Shaker Verlag GmbH, Dortmund, Germany, 2000.

[7] Wall, B. J., and Conway, B. A., "Genetic Algorithms Applied to the Solution of Hybrid Optimal Control Problems in Astrodynamics," *Journal of Global Optimization*, Vol. 44, No. 4, 2009, pp. 493–508. doi: 10.1007/s10898-008-9352-4

[8] Dorigo, M., and Stützle, T., *Ant colony optimization*, The MIT Press, Cambridge, MA, 2004.

[9] Dorigo, M., and Gambardella, L. M., "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, 1997, pp. 53–66. doi: 10.1109/4235.585892

[10] Merkle, D., Middendorf, M., and Schmeck, H., "Ant Colony Optimization for Resource-Constrained Project Scheduling," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 4, 2002, pp. 333–346. doi: 10.1109/TEVC.2002.802450

[11] Blum, C., "Beam-ACO—Hybridizing Ant Colony Optimization with Beam Search: An Application to Open Shop Scheduling," *Computers and Operations Research*, Vol. 32, No. 6, 2005, pp. 1565–1591. doi: 10.1016/j.cor.2003.11.018

[12] McAdams, J. V., Dunham, D. W., Farquhar, R. W., Taylor, A. H., and Williams, B. G., "Trajectory Design and Maneuver Strategy for the MESSENGER Mission to Mercury," *Journal of Spacecraft and Rockets*, Vol. 43, No. 5, 2006, pp. 1054–1064. doi: 10.2514/1.18178

[13] Yárnoz, D. G., Jehn, R., and Croon, M., "Interplanetary navigation along the low-thrust trajectory of BepiColombo," *Acta Astronautica*, Vol. 59, No. 1–5, 2006, pp. 284–293.

[14] Pisarevsky, D. M., and Gurfil, P., "A Memetic Algorithm for Optimizing High-Inclination Multiple Gravity-Assist Orbits," *Proceedings of IEEE Congress on Evolutionary Computation, CEC 2009*, IEEE, Trondheim, Norway, 2009.

[15] Marsden, R., and Angold, N., "The Epic Voyage of Ulysses," *European Space Agency Bulletin*, No. 136, 2008, pp. 3–7.

[16] Kaplan, M. H., *Modern Spacecraft Dynamics and Control*, Wiley, New York, NY, 1976.

[17] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed., Cambridge University Press, 1992.

[18] Ceriotti, M., Vasile, M., and Bombardelli, C., "An Incremental Algorithm for Fast Optimisation of Multiple Gravity Assist Trajectories," *Proceedings of 58th International Astronautical Congress*, International Astronautical Federation, Hyderabad, India, 2007.

[19] Peralta, F., and Flanagan, S., "Cassini Interplanetary Trajectory Design," *Control Engineering Practice*, Vol. 3, No. 11, 1995, pp. 1603–1610. doi: 10.1016/0967-0661(95)00171-P

[20] Ceriotti, M., and Vasile, M., "Automatic MGA Trajectory Planning with a Modified Ant Colony Optimization Algorithm," *Proceedings of the 21st International Symposium on Space Flight Dynamics*, CNES, Toulouse, France, Sept. 28, Oct. 2 2009.

[21] Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, Boston, MA, 1989.

[22] Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T., "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II," *Proceedings of 6th International Conference on Parallel Problem Solving from Nature, PPSN VI*, Springer, Paris, France, 2000.

[23] Vasile, M., Minisci, E., and Locatelli, M., "On Testing Global Optimization Algorithms for Space Trajectory Design," *Proceedings of AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, AIAA, Honolulu, Hawaii, 2008.

[24] Vasile, M., Minisci, E., and Locatelli, M., "A Dynamical System Perspective on Evolutionary Heuristics Applied to Space Trajectory Optimization Problems," *Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC2009)*, IEEE, Trondheim, Norway, May 2009.

[25] Vasile, M., Minisci, E., and Locatelli, M., "Analysis of some global optimization algorithms for space trajectory design," *Journal of Spacecraft and Rockets*, Vol. 47, No. 2, 2010, pp. 334–344. doi: 10.2514/1.45742

[26] Yárnoz, D. G., De Pascale, P., Jehn, R., Campagnola, S., and Corral, C., "BepiColombo Mercury Cornerstone Consolidated Report on Mission Analysis," MAO Working Paper 466, ESA-ESOC Mission Analysis Office, 2006.

[27] Jehn, R., Yárnoz, D. G., and Yanez, A., "BepiColombo Mercury Cornerstone Mission Analysis: Chemical Options, MAO Working Paper 486, ESA-ESOC Mission Analysis Office, 2005.

Christopher Rouff
Associate Editor