JOURNAL OF SPACECRAFT AND ROCKETS
Vol. 46, No. 2, March–April 2009

Hybrid Evolutionary Algorithm for the Optimization of Interplanetary Trajectories

Matteo Rosa Sentinella[*] and Lorenzo Casalino[†]
Politecnico di Torino, 10129 Torino, Italy

DOI: 10.2514/1.38440

A hybrid evolutionary algorithm is applied to the design of interplanetary trajectories with multiple impulses and gravity assists. The optimization procedure runs three different optimizers based on genetic algorithms, differential evolution, and particle swarm optimization "in parallel"; the algorithms, which can also be employed separately, are used synergistically here by letting the best individuals, found by each algorithm, migrate to the others at prescribed intervals. A comparison with the results presented in recent literature, which state that differential evolution is well suited to deal with this kind of problem, is carried out. The performance of the hybrid optimizer is comparable to that of differential evolution in terms of computational time and function evaluations when problems with a reduced number of variables are considered. The hybrid optimizer may instead exhibit better performance when more complex problems are dealt with. The results also show that the algorithm performance is remarkably improved by introducing a "mass mutation" operator to avoid premature convergence to suboptimal solutions and by means of a particular choice of the variables to describe the trajectory.

Nomenclature

C	=	scaling factor for differential evolution
c_1, c_2	=	learning factors for particle swarm optimization
F	=	scaling factor for differential evolution
k_1, k_2	=	random numbers for particle swarm optimization
N	=	number of function evaluations
N_g	=	number of generations
N_i	=	number of individuals
N_p	=	number of optimization variables
n	=	number of legs
n_{rev}	=	number of complete revolutions for second leg
P	=	crossover probability distribution for genetic algorithm
r_p	=	periapsis radius, km
t	=	time, s
$t_{95\%}$	=	average time to obtain global optimum, s
u	=	random number for genetic algorithm
v	=	particle velocity in particle swarm optimization
v_c	=	circular velocity, km/s
v_p	=	orbital velocity at periapsis, km/s
v_∞	=	hyperbolic excess velocity, km/s
x	=	variable value of parent individual
y	=	variable value of new individual
β	=	crossover variable for genetic algorithm
ΔV	=	velocity increment, km/s
δ	=	hyperbolic excess velocity rotation, deg
ϵ	=	efficiency
η	=	crossover parameter for genetic algorithm
μ	=	gravitational parameter, km^3/s^2

Subscripts

0	=	departure
avg	=	average
best	=	best individual
DSj	=	jth deep-space impulse
FBj	=	jth flyby
G_{best}	=	global best
max	=	maximum
n	=	arrival
P_{best}	=	particle best

Received 8 May 2008; accepted for publication 21 January 2009.

[*]Research Assistant, Dipartimento di Energetica, Corso Duca degli Abruzzi, 24.

[†]Associate Professor, Dipartimento di Energetica, Corso Duca degli Abruzzi, 24. Senior Member AIAA.

I. Introduction

EVOLUTIONARY algorithms (EAs) are optimization procedures that search for the solution that maximizes or minimizes a given function in a prescribed search space. Each solution (individual) is represented by the integer or real values of a finite number of variables, which can vary in prescribed intervals. The optimization procedure is started by producing, usually in a random way, an initial population of individuals. Each algorithm is characterized by its own rules that force the evolution of the population to favor the improvement of the function to be optimized. Some parameters, typical of each algorithm, control the evolution and determine the algorithm capability of finding the optimal solution. Genetic algorithms (GA), differential evolution (DE), and particle swarm optimization (PSO) are used in the present paper.

Initial applications of EA to space trajectory optimization mainly employed GAs in conjunction with gradient-based methods [1–4]. GAs have also been used with calculus of variations [5–7] for low-thrust trajectories; a combination of artificial neural networks with EAs has been applied to solar sail trajectories [8].

EAs are better suited to the optimization of impulsive trajectories, which can be described by a limited number of variables. In this case, the number of function evaluations that are required to obtain the optimal solution is usually acceptable, whereas the large number of variables required to describe low-thrust trajectories with sufficient accuracy makes the use of EA for this kind of problem less attractive. Several studies concerning the optimization of impulsive interplanetary trajectories with gravity assists by means of EAs have been published in the recent literature; tuning of the algorithm parameters and comparisons of the performance of the algorithms have been presented [9–16]. These studies usually agree in stating that EAs are suitable means for the optimization of this kind of mission, even though their conclusions sometimes differ in determining which particular method allows for the best performance. Results may be affected by the choice of the parameters that rule the behavior of the optimization algorithms, and comparison of the results is sometimes difficult due to insufficient details given about how the trajectory was modeled.

The tuning of the parameters is useful only when it has a general validity and the algorithm performs well when applied to different problems. From this point of view, the performance of DE in the optimization of missions with up to four gravity assists and one deep-space impulse has been presented in a recent paper [17] (a two-flyby round-trip mission to comet Tempel 1 with two deep-space impulses was also analyzed). The influence of the parameters of the optimization algorithm on the time that is required to obtain the optimal solution with a prescribed confidence level was investigated. A combination of parameters that performed well with all the considered problems was successfully found.

In the present paper, a hybrid evolutionary algorithm is presented and applied to the design of interplanetary trajectories with multiple impulses and gravity assists. Thanks to the synergistic effort of different EAs, hybrid algorithms should perform well on different problems, with generic settings of the algorithm parameters and without any specific tuning; only population size and number of function evaluations should be varied to take the dimension of the problem (i.e., number of optimization variables) into account. The hybrid optimization procedure presented here runs three different optimizers based on GA, DE, and PSO "in parallel"; the algorithms, which can also be employed separately, are used synergistically here by letting the best individuals found by each algorithm migrate to the others at prescribed intervals. In comparison with existing methods, a "mass mutation" (MM) operator is also introduced: a high percentage of individuals (typically over 90%) is eliminated when the mean distance between the individuals is smaller than a prescribed value and the optimal objective function is stuck on the same value for more than a prefixed number of iterations [18].

The combination of parameters chosen in [17] is first tested by using the DE algorithm alone. A comparison with the performance of the hybrid method is then presented. Results will also show that a suitable choice of the variables to describe the trajectory can significantly improve the performance.

II. Evolutionary Algorithms

An EA is a population-based metaheuristic optimization algorithm. An EA borrows its mechanisms from biological evolution: reproduction, mutation, recombination, natural selection, and survival of the fittest. Candidate solutions to the optimization problem play the role of individuals in a population, and a cost or merit function determines how the individual is suited for the environment. Evolution of the population takes place with the repeated application of prescribed operators. Evolutionary algorithms consistently perform well in approximating solutions to all types of problems because they do not make any assumption about the underlying fitness landscape; this generality is shown by successes in fields as diverse as engineering, art, biology, economics, genetics, operations research, robotics, social sciences, physics, and chemistry.

In the present paper, interplanetary trajectories with a fixed sequence of planetary encounters are considered; a deep-space impulse may be applied during each interplanetary leg. Each trajectory is defined by the real values of a fixed number of variables that completely determine the trajectory (i.e., relevant dates and position, magnitude and orientation of each impulse). The variables can assume values in user-specified ranges. The mission ΔV is the function to be minimized.

Usually, an initial population of randomly generated candidate solutions comprise the first generation. The fitness function, in this case the inverse of the mission ΔV, is evaluated for the candidate solutions. Subsequent offspring are generated with the rules of the optimization algorithm, with a bias toward higher fitness. The new candidates compete with old individuals for their place in the next generation (survival of the fittest). This process can be repeated until a solution with sufficient quality is found or a previously determined computational limit (number of iterations or number of function evaluations) is reached. Similar techniques, which differ in the implementation details and the nature of the particular applied problem, have been proposed. An optimization code which employs GA, DE, and PSO has been developed [19] and is used in this paper.

A. Genetic Algorithm

GAs are the most popular type of EA, used in different domains because of their broad applicability and ease of use [20,21]. GAs work on a population of potential solutions by applying the principle of survival of the fittest to produce better and better approximations of the optimal solution. At each iteration, a new set of approximations is created by selecting parent individuals according to their fitness and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals, which are better suited to their environment than the individuals that they were created from, just as in natural adaptation [22,23].

A starting population of N_i individuals is created by means of a random number generator with uniform distribution. Each individual is characterized by the real values of N_p optimization variables. For each individual, the mission ΔV, which is the objective function here minimized, is evaluated; a large value is associated with the combinations that do not allow for a solution. The fitness function is defined as the inverse of ΔV. In the proposed algorithm, the extended random initialization (ERI) procedure of Bramlette [24] is introduced, whereby random initializations are tried for each individual until a significant solution is found or a maximum number of tries (20 in this paper) is reached. This initial effort in terms of computational cost is repaid with a faster convergence to the global optimum, which is reached after a lower number of function evaluations [18].

Genetic operators are then used iteratively to get an improved population. The principle of *elitism* is first used to avoid the loss of good individuals caused by other genetic operators (mutation, crossover). The best individuals, that is, those with the highest fitness (lowest ΔV), are stored and replace the worst ones of the following generation; in the present work, three individuals are selected and saved during each iteration.

The new generation is obtained by first forming a "parent" population, which emphasizes the good solutions and eliminates the bad solutions. The GA optimization tool that has been developed can use three different types of *selection operator*:

1) Tournament selection: Two individuals are randomly chosen and compared with each other; the best is placed in a slot of the parent population. This is done systematically and each individual participates in exactly two tournaments, so that it could be present twice in the parent population if it wins both tournaments or it could be eliminated if it loses twice. The tournament selection has faster convergence and lower computational time and complexity compared with any other selection operator that exists in the literature [23], but can become stuck very easily in local minima.

2) Roulette wheel selection: The probability of each individual being selected for reproduction is chosen to be proportional to the fitness value. The basic roulette wheel selection method is stochastic sampling with replacement, and it can potentially have an unlimited spread, because any individual with a positive fitness function could entirely fill the parent population.

3) Stochastic universal sampling: This operator is an evolution of the previous one, with the aim of reducing the spread of the parent population. Instead of the single selection pointer employed in roulette wheel methods, the stochastic universal sampling uses N equally spaced pointers, where N is the number of selections required.

Only tournament selection is used here.

A crossover operator is then applied to get a new generation of individuals: couples of individuals are randomly selected among the parent population so that each individual is chosen once. In the real-coded formulation, crossover is applied to each variable according to a probability distribution defined as follows:

$$P(\beta) = \begin{cases} 0.5(\eta + 1)\beta^{\eta} & \text{for } \beta \leq 1 \\ 0.5(\eta + 1)/\beta^{(\eta+2)} & \text{otherwise} \end{cases} \tag{1}$$

The procedure determines the new variable values y_1 and y_2 from those of the parent individuals x_1 and x_2; a random number $0 \leq u \leq 1$ is first chosen, $\bar{\beta}$ is determined so that

$$\int_0^{\bar{\beta}} P(\beta)\,\mathrm{d}\beta = u$$

and the "children" are computed: $y_1 = 0.5[(x_1 + x_2) - \bar{\beta}|x_2 - x_1|]$ and $y_2 = 0.5[(x_1 + x_2) + \bar{\beta}|x_2 - x_1|]$. The parameter η controls how close children solutions are with respect to parent solutions (the larger η, the closer the solutions); $\eta = 2$ is usually suggested and this value is adopted here.

Mutation is applied to the new population to increase the number of explored solutions and keep diversity in the population. Some of the variables are changed according to a small specified probability (2% is used in the present paper; results are only slightly affected by the chosen value); the new variable value is chosen randomly in the specified range. The objective function of each individual of the new generation is finally evaluated; the worst ones are discarded and replaced by the elite individuals of the former generation. The whole procedure is repeated for a fixed number of generations, N_g, or until a prefixed number of function evaluations is reached.

B. Differential Evolution

DE [25,26] is a parallel direct search method that uses a population of N_i individuals of N_p dimension for N_g generations, just as the GA does. ERI can be optionally added as well, to initialize the population. Basically, DE generates new vectors of variables by adding the weighted difference between two population vectors to a third one. If the resulting individual exhibits a higher fitness than a predetermined population member, in the next generation the new individual replaces the one it was compared with; otherwise, the old individual is retained. This basic principle can be varied and, in fact, there are several practical variants of DE. For example, a linear combination of three vectors, instead of two, can be used to determine the new individuals, or the comparison can be made between the new vector and the best individual, instead of one chosen randomly. Six variants of DE, named according to [26], can be used in the developed algorithm to determine the new variable values, y:

1) DE/best/1:

$$y = x_{\text{best}} + F(x_1 - x_2)$$

2) DE/rand/1:

$$y = x_1 + F(x_2 - x_3)$$

3) DE/rtb/2:

$$y = x_1 + F(x_{\text{best}} + x_2 - x_3)$$

4) DE/best/2:

$$y = x_{\text{best}} + F(x_1 + x_2 - x_3 - x_4)$$

5) DE/rand/2:

$$y = x_5 + F(x_1 + x_2 - x_3 - x_4)$$

6) DE/rtb/1:

$$y = x_1 + C(x_{\text{best}} - x_1) + F(x_2 - x_3)$$

where F and C are scaling factors, $x_j (j = 1, 5)$ are the variable values of vectors chosen randomly, x_{best} is the variable value of the best individual in the population, and rtb refers to random to best. DE/rand/1 is used here with either $F = 0.6$ or F chosen randomly between -1 and 1, as suggested by [17]. The crossover probability [25,26] is fixed at 0.8.

C. Particle Swarm Optimization

PSO is a population-based stochastic optimization technique, inspired by the social behavior of bird flocking or fish schooling [27,28]. The system is initialized with a population of random solutions and searches for the optimum by updating generations. However, unlike the GA or DE, PSO has no operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. Compared with the GA, PSO is quite easy to implement and there are few parameters to adjust. PSO has been successfully applied in many areas: function optimization, artificial neural network training, fuzzy system control, and other areas in which the GA can be applied.

The original intent was to graphically simulate the choreography of a bird flock or fish school, in which each individual follows the one that is nearest to the food. PSO learns from this scenario and uses it to solve optimization problems. In PSO, each single solution is a "bird" in the search space and is called a "particle." The values of the N_p optimization variables define the particle position. A velocity vector, which rules the motion of the particle, is also introduced. The particles fly through the problem space by following the particle with the best fitness function. PSO is initialized with a group of random particles; each particle moves according to its instantaneous velocity, which is updated at each iteration to improve the solution. Two criteria are used: the velocity is changed to move the particle toward 1) the best solution P_{best} that the particle itself has reached in the previous iterations (cognitive acceleration) and 2) the best solution G_{best} that any particle in the population has reached (social acceleration). The particle updates its velocity and positions according to the relations

$$v = v + c_1 \cdot k_1 \cdot (x_{P_{\text{best}}} - x) + c_2 \cdot k_2 \cdot (x_{G_{\text{best}}} - x) \tag{2}$$

$$y = x + v \tag{3}$$

where v is the particle velocity, x and y are the current and updated particle positions (i.e., variable values), and k_1 and k_2 are random numbers in the $[0, 1]$ interval. The learning factors c_1 and c_2 are assumed here to be equal to 2. The particles' velocities are limited to a fixed value, $V_{\max}$, equal to one-quarter of the variable range. The Trelea type 2 strategy is adopted [29].

Compared with GAs and DE, the information-sharing mechanism in PSO is significantly different. In GAs and DE, individuals share information with each other, so that the whole population moves like a single group toward an optimal area. In PSO, only one solution, G_{best}, shares information with the others, with a one-way information-sharing mechanism. Compared with the GA, all the particles tend to converge to the best solution more quickly, in most cases.

III. Trajectory Model and Search Space

The patched-conic approximation is adopted; the time spent inside the planet sphere of influence during a flyby is neglected, and gravity-assist maneuvers are represented by discontinuities in the spacecraft velocity. Under these assumptions, only the heliocentric motion can be considered and the trajectory consists of ballistic (i.e., Keplerian) arcs joined in correspondence of impulses and flybys. Static ephemerides are used, as it has been verified that the use of more accurate models (such as the Jet Propulsion Laboratory DE405 ephemerides) does not alter significantly the results. The relevant data are provided in Table 1. In the present paper, trajectories from the Earth to Saturn are considered; the spacecraft performs a predetermined sequence of planetary flybys before reaching its target. The departure date is the first optimization variable. The trajectory that joins two planets is called a "leg"; either pure ballistic legs or legs with a deep-space impulse are considered. Ballistic legs are described by a single variable, that is, the leg time length; departure and arrival position are determined by the corresponding dates, and Lambert's problem can be solved to determine the spacecraft velocities at departure and arrival. Four additional variables are required for each deep-space impulse, to define the impulse. Two different choices of variables are adopted here. First, as suggested in [17], position (three variables, spherical coordinates are adopted) and impulse date (one variable, i.e., the time from the leg start to the impulse divided by the overall leg time length) can be chosen (variable set A); in this case, two Lambert's problems must be solved. This formulation allows for the maximum generality, but

Table 1 Planet data

Planet	Venus	Earth	Jupiter	Saturn
Semiaxis, AU	0.72333566	1.00000261	5.20288700	9.53667594
Eccentricity	0.00677672	0.01671123	0.04838624	0.05386179
Inclination, °	3.39467605	−0.00001531	1.30439695	2.48599187
Mean longitude, °	181.97909950	100.46457166	34.39644051	49.95424423
Long. of perihelion, °	131.60246718	102.93768193	14.72847983	92.59887831
Long. of ascending node, °	76.67984255	0.0	100.47390909	113.66242448
Radius, km	6052	6378	71,492	60,268
Grav. parameter, km^3/s^2	324,860	398,600	126,700,000	37,900,000

many solutions are affected by bad performance, because, in the general case, there is not agreement between the impulse time and position.

Alternatively, a different approach can be adopted by making reference to the ballistic trajectory joining the planets (variable set B). In fact, one searches for solutions with small impulses, and the leg that joins two planets with a deep-space impulse should be relatively close to the ballistic arc. Given the leg initial and final dates, Lambert's problem is first solved to find the ballistic arc that joins the planets and the initial velocity of the ballistic leg is evaluated. Three optimization variables are used to obtain the actual components of the leg initial velocity from the components of the ballistic solution. The fourth variable defines the impulse date, as in the previous case. Kepler's equation from departure to the impulse is solved to determine the velocity just before the impulse and its position. Lambert's problem from the impulse to the planet encounter is then solved to determine the velocity just after the impulse and at arrival. Numerical methods described in [30] are used to solve Kepler's and Lambert's problems.

The algorithms operate synergistically: every 20 iterations, the five best individuals are selected and distributed randomly to the populations, where they replace the worst individuals, to share information between the algorithms. A similar cooperative approach has been employed with good results in [16], where DE and PSO were used "in series" for the optimization of ballistic trajectories with multiple flybys. In the present paper, instead, the GA, DE, and PSO are used "in parallel," allowing them to share information during the optimization process.

For the sake of comparison, the DE algorithm is also run separately, with the optimal tuning of the optimization parameters suggested by [17]. In particular, when the number of variables is around 10, 4000 generations for a 28-individual population are evaluated with $F = 0.6$ (112,000 function evaluations); larger populations and numbers of function evaluations are tried for more complex problems with a larger number of variables, with F taken randomly between -1 and 1. General nonoptimized settings are adopted for the parameters of the hybrid optimization algorithm. Only the size of the population and the number of function evaluations at which the search is stopped are changed according to the number of variables following generic heuristic rules of thumb. According to past experience [18], a fixed population of 30 individuals is chosen for PSO; 100 and 60 individuals are chosen for the GA and DE, respectively, when the number of variables is 10; 300 and 120 individuals are instead chosen when the number of variables is larger. Note that the maximum number of function evaluations (approximately the number of individuals times the number of generations) must be chosen carefully; a low value may prevent attainment of the global optimum, whereas a large value increases the computational time. A detailed analysis of the influence of this parameter is beyond the scope of the present paper and is not carried out; generic settings, driven by the user's experience, should be adopted: 30,000 or 40,000 times the number of optimization variables is a reasonable value. Mass mutation is performed when the solution remains unchanged for 50 generations, by deleting all but one individual of the GA and PSO populations; 10% of the individuals are instead retained in the DE population.

The trajectory consists of n legs. Subscripts $j-1$ and j correspond to the departure and arrival planets of the jth leg. The mission ΔV is the sum of the velocity changes that are performed: at Earth departure (ΔV_0), at target arrival (ΔV_n), at the flybys (ΔV_{FBj}, $j = 1, 2, \ldots, n-1$), and in correspondence of the deep-space impulses (ΔV_{DSj}, $j = 1, 2, \ldots, n$).

$$\Delta V = \Delta V_0 + \Delta V_n + \sum_{j=1}^{n-1} \Delta V_{FBj} + \sum_{j=1}^{n} \Delta V_{DSj} \tag{4}$$

At departure, the spacecraft leaves a given parking orbit with an impulse at the orbit perigee; in a similar way, the insertion into a prescribed parking orbit around the target planet is performed by means of an impulse at periapsis. One has

$$\Delta V_0 = \sqrt{v_{\infty 0}^2 + 2v_{c0}^2} - v_{p0} \qquad \Delta V_n = \sqrt{v_{\infty n}^2 + 2v_{cn}^2} - v_{pn} \tag{5}$$

where v_∞ is the relative velocity at the boundary of the sphere of influence (hyperbolic excess velocity), and v_c and v_p are the circular and orbital velocity at the periapsis of the parking orbit, respectively.

At the jth flyby ($j = 1, 2, \ldots, n-1$), a minimum height over the planet surface is considered to determine the maximum allowable rotation $\delta_{\max}$ of the hyperbolic excess velocity v_∞; one has

$$\sin(\delta_{\max}/2) = \frac{\mu_j/r_{pj}}{v_\infty^2 + \mu_j/r_{pj}} \tag{6}$$

where μ_j is the planet's gravitational parameter, and r_{pj} is the radius corresponding to the minimum flyby altitude. The lower value of the hyperbolic excess velocity must be considered when the magnitude of the hyperbolic excess velocity before the flyby, $v_{\infty j-}$, differs from the value after the flyby, $v_{\infty j+}$. When the maximum rotation is not exceeded, the velocity change at the flyby is the change in v_∞ magnitude:

$$\Delta V_{FBj} = |v_{\infty j+}| - |v_{\infty j-}| \tag{7}$$

On the other hand, when δ exceeds $\delta_{\max}$, the velocity rotation must be partially provided by propulsion with an impulse just before (when $v_{\infty j+} < v_{\infty j-}$) or after (otherwise) the flyby (propulsion is excluded inside the planet's sphere of influence). One has

$$\Delta V_{FBj} = \sqrt{v_{\infty j+}^2 + v_{\infty j-}^2 - 2v_{\infty j+}v_{\infty j-}\cos(\delta - \delta_{\max})} \tag{8}$$

Finally, the velocity changes in correspondence of the deep-space impulses ($j = 1, 2, \ldots, n$) must be computed as the vectorial difference of the velocity just before (v_{j-}) and after (v_{j+}) the impulse:

$$\Delta V_{DSj} = |v_{j+} - v_{j-}| \tag{9}$$

IV. Results

Different problems are treated to test the method performance. The optimization is carried out 100 times for each mission, each choice of the set of variables that describe the trajectory, and each setting of the optimization algorithm. Results are compared in terms of efficiency ϵ (i.e., the number of times that the obtained ΔV is within 1% of the global optimum divided by the total number of tries), average computational time per try t, and time required to obtain the global optimum with a 95% confidence $t_{95\%} = t \log(1 - 0.95)/\log(1 - \epsilon)$. A standard 2.13 GHz PC is used in the present analysis to perform the calculations. The computational time grows almost linearly with the number of function evaluations N, and $t_{95\%}$ could be optimized a posteriori by choosing N properly. For this purpose, the average number of function evaluations N_{avg}, which is required to obtain the threshold value (1.01 times the global optimum) in the successful runs, is also determined; the difference with respect to N is an index of how much $t_{95\%}$ can be improved by reducing N.

A. Cassini Mission with One Deep-Space Impulse

The trajectory that the spacecraft Cassini–Huygens flew en route to Saturn is a significant test for any optimization method. The trajectory leaves the Earth and reaches Saturn after gravity assists from Venus (twice), Earth, and Jupiter; a deep-space impulse is performed during the Venus–Venus leg ($j = 2$); the other legs are ballistic, except for negligible correction maneuvers. This mission structure is retained here; the trajectory is defined by 10 variables (departure date, four flyby dates, arrival date, and four variables for the deep-space impulse). The same analysis presented in [17] is carried out here, with the launch date confined to 1997. The overall trip time is constrained to 7 years. The same variable ranges as in [17] are adopted: deep-space maneuvers are limited to the region between the orbits of Earth and Jupiter within 5 deg of the ecliptic plane, unless otherwise specified. Transfer times between inner planets are allowed to range from 0.1 to 1.5 orbital periods of the outermost planet involved. Resonant transfers range from 1:1 to 1:3 spacecraft revolutions to planet revolutions. Transfers that involve an outer planet can range in duration from 0.1 to 0.4 orbital periods of the outermost planet involved. The variable that defines the impulse date may assume any value from 0.001 to 0.999.

The spacecraft leaves a 300 km circular low Earth orbit; the flyby height is constrained above 300 km for all the planets; the final orbit has an altitude of 0.33×150 Saturn radii. The global optimum presents $\Delta V = 4.892$ km/s, and the corresponding trajectory is shown in Fig. 1. No impulse is required during the flybys. Note the good agreement with the actual Cassini mission, which presents almost the same velocity changes and dates, except for a slightly shorter trip time (15 October 1997, 26 April 1998, 3 December 1998, 24 June 1999, 18 August 1999, 30 December 2000, and 1 July 2004). The minimum ΔV is slightly different from that presented in [17], possibly because of different values adopted for the relevant planetary constants (the departure phase was also modeled differently).

First, the DE optimizer is used alone with the optimal settings provided in [17]. The same set of optimization variables (set A), with the impulse defined by position in spherical coordinates and date, is used. A maximum number of 112,000 function evaluations (4000 generations of 28 individuals) is fixed for each try. A good agreement

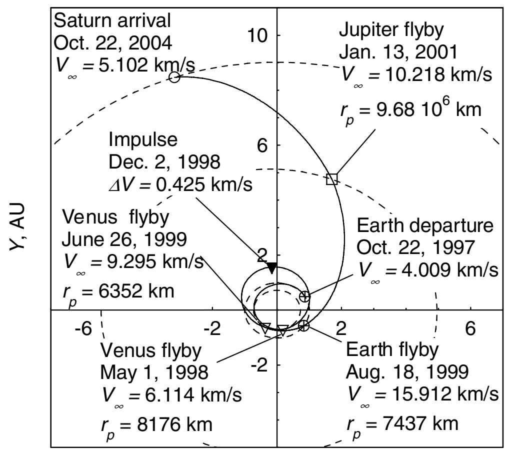

Fig. 1 Optimal trajectory for the Cassini mission.

Table 2 Results for the one-impulse Cassini mission with variable set A ($F = 0.6$)

Algorithm	N	ϵ	t, s	$t_{95\%}$, s	N_{avg}
DE	112,000	0.16	10.5	194	35,637
DE MM	112,000	0.15	10.8	201	41,065
Hybrid	300,000	0.16	25.0	462	153,532
DE	300,000	0.20	29.0	389	55,099
DE MM	300,000	0.37	30.1	195	96,470

is obtained, as the results, shown in Table 2, are essentially the same as in [17]; a lower efficiency (16 vs 19%) is obtained here, but an exact comparison is not possible as the range for the impulse date adopted in [17] is not given explicitly. The introduction of the MM operator does not improve the algorithm efficiency and causes a slight increase of $t_{95\%}$. The hybrid optimizer is then tested on the same problem. Populations of 100, 60, and 30 individuals are chosen for the GA, DE, and PSO, respectively. The number of function evaluations is increased to 300,000 (roughly 1500 generations), to take the larger populations into account. No improvement in terms of efficiency is obtained, but the larger number of function evaluations increases $t_{95\%}$. Table 2 shows that the DE algorithm without MM finds the solution after a small number of function evaluations, whereas more evaluations are required when the MM operator is introduced; this observation suggests that an increase in the number of function evaluations should be beneficial when MM is used. In fact, as expected, the improvement in terms of efficiency caused by an increase in N is small for the simple DE algorithm, but becomes remarkable when MM is introduced, so that $t_{95\%}$ becomes satisfactory. The random generation of new populations permits the escape from suboptimal solutions when the algorithm becomes stuck on a local optima.

The convergence toward the global optimum suffers from the independence of the variables that describe the deep-space impulse. In fact, there is no relation between the parameters used to define the impulse date and position, and many solutions require a very large ΔV (e.g., if the impulse date is close to the leg departure and the position is far from the departure planet). This consideration suggests a change in the variables used to define the impulse; when small deep-space impulses are sought, as should occur in a good-performance mission, one can make reference to the ballistic trajectory, which joins the relevant planets for each pair of leg departure and arrival dates, and the corresponding departure velocity. Three optimization variables are introduced to determine the leg initial velocity; the actual components of the relative velocity vector v_∞ at the leg departure are expressed as the product of the optimization variable and the corresponding components of the ballistic trajectory. The range for these variables is initially fixed between 0 and 2. This choice of the optimization variables (variable set B) should guarantee an improved convergence (as the search space is reduced) while not eliminating any significant (i.e., good-performance) solution, as only high-ΔV missions should be excluded. The ranges of the variables are to be broadened if good solutions on, or close to, the boundaries are found. The results obtained with this new formulation are shown in Table 3; in comparison with the previous results, the efficiency improvement is striking; moreover, MM now also provides a significant improvement for the smallest number of function evaluations. The increase of N is again effective only when the MM operator is introduced and the efficiency reaches 100% for the DE algorithm. A remarkable reduction of $t_{95\%}$ is obtained in comparison with the use of variable set A, even though the average time per try is almost doubled, due to the addition of a Kepler's problem in the leg with the deep-space impulse (a slow but robust solution method based on linear interpolation is adopted here).

B. Cassini Mission with Multiple Deep-Space Impulses

In general, one does not know how many deep-space impulses are required for the optimal mission and in which legs they should occur.

Table 3 Results for the one-impulse Cassini mission with variable set B ($F = 0.6$)

Algorithm	N	ϵ	t, s	$t_{95\%}$, s	N_{avg}
DE	112,000	0.39	19.5	118	17,669
DE MM	112,000	0.89	16.8	23	38,513
Hybrid	300,000	0.82	40.8	71	77,334
DE	300,000	0.40	51.5	302	17,510
DE MM	300,000	1.00	45.3	45	48,222

Table 4 Results for the three-impulse Cassini mission with variable set B ($N = 500,000$)

Algorithm	F	ϵ	t, s	$t_{95\%}$, s	N_{avg}
DE	0.6	0.20	104.2	1398	84,127
DE MM	0.6	0.36	94.6	635	157,803
Hybrid	0.6	0.41	66.2	376	307,593
DE	$[-1, 1]$	0.35	101.5	706	47,514
DE MM	$[-1, 1]$	0.67	92.2	249	97,206
Hybrid	$[-1, 1]$	0.64	84.1	247	255,136

Because four variables are required for each deep-space impulse, the problem dimension grows significantly when multiple impulses are allowed. In particular, 26 variables are required if a deep-space impulse is allowed in all five legs of the Cassini mission. A simpler problem with 18 variables arises when the impulses are allowed only in the three central legs ($j = 2$, 3, and 4). Note that the optimal solution does not change, as only the impulse during the Venus–Venus leg is required, whereas the other impulses must vanish. The three-impulse mission is first considered. The formulation with variable set A never reaches the global optimum; the minimum value obtained for ΔV is 8.55 km/s, even though different population sizes have been tried (the range of the position of the impulse in the Venus–Earth leg has also been reduced to 0.7–1.2 AU). The initial solutions are too spread out to allow for convergence. On the contrary, the use of variable set B makes the algorithms able to obtain the global optimum; the results are shown in Table 4. A population of 60 individuals is adopted for the DE algorithm, populations of 300, 120, and 30 are assumed for the GA, DE, and PSO algorithms of the hybrid optimization method, respectively. Different values of the mutation factor F of the DE algorithm are tried. $F = 0.6$, optimal for the single-impulse mission when a 28-individual population is adopted [17], is now outperformed by the assumption of a random number between -1 and 1, which constitutes the setting that has the best average performance on different problems [17]. The introduction of MM again provides a very large benefit in terms of efficiency, with a consequent large reduction of $t_{95\%}$. The hybrid algorithm performs better than the DE algorithm when the nonoptimal DE setting ($F = 0.6$) is adopted, whereas ϵ and $t_{95\%}$ are almost the same when the optimal setting $F = [-1, 1]$ is adopted (note that the hybrid algorithm benefits from a lower average computational time per try). Larger values of N have also been tried; again, a small number of function evaluations is sufficient when the simple DE algorithm is employed (an increase in N only marginally improves the efficiency). On the contrary, the DE and hybrid algorithms with MM exhibit better performance and a remarkable increase in efficiency as N grows (typically, a 10–15% increase for $N = 1,000,000$ and an additional 5–10% increase for $N = 1,500,000$). One could a posteriori determine the optimal value for N to minimize $t_{95\%}$, but this is beyond the scope of the present paper.

The problem becomes extremely complex when a deep-space impulse is allowed in each leg. Only the algorithms with MM have been tested, with the same population sizes of the three-impulse case and with 1,000,000 function evaluations. The results are presented in Table 5. In this case, the hybrid algorithm shows better performance compared with the DE algorithm, with both the adopted values of F; also, the average time per try is remarkably lower, with a conspicuous reduction of $t_{95\%}$. However, the DE algorithm finds the optimum after a lower number of evaluations when a random value is assumed for F, and $t_{95\%}$ could be greatly reduced a posteriori by choosing N properly.

Table 5 Results for the five-impulse Cassini mission with variable set B ($N = 500,000$)

Algorithm	F	ϵ	t, s	$t_{95\%}$, s	N_{avg}
DE MM	0.6	0.35	207.1	1440	485,626
Hybrid	0.6	0.41	199.7	1134	411,157
DE MM	$[-1, 1]$	0.51	220.1	924	156,375
Hybrid	$[-1, 1]$	0.53	181.4	720	392,854

C. Fast Mission to Saturn with Multiple Deep-Space Impulses

A fast mission to Saturn is also considered because of some peculiarities. A 6 year trip time is imposed and one flyby of Venus is dropped, compared with the Cassini mission. The spacecraft performs three gravity assists (Venus, Earth, and Jupiter). Twenty-one variables are required to describe the mission when four deep-space impulses are allowed. A simplified problem, which allows a single deep-space impulse during the Venus–Earth leg, has nine optimization variables. Variable set B is adopted, and the ranges for the optimization variables are unchanged compared with the Cassini mission.

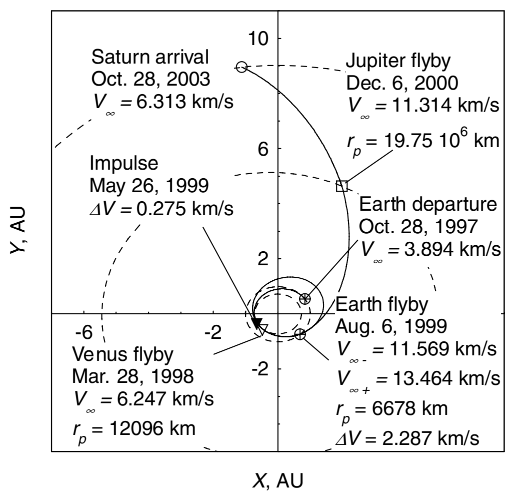

Fig. 2 Optimal fast mission to Saturn ($\Delta V = 7.211$ km/s).

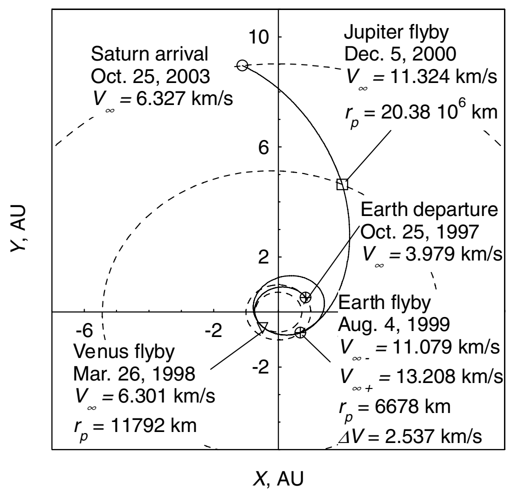

Fig. 3 Suboptimal fast mission to Saturn ($\Delta V = 7.217$ km/s).

Table 6 Results for the fast mission to Saturn

Algorithm	Impulses allowed	n_{rev} allowed	ϵ	N_{avg}	$\bar{\epsilon}$	$\bar{N}_{\text{avg}}$
DE MM	1	0 or 1	0.73	74,404	0.07	136,978
Hybrid	1	0 or 1	0.36	70,000	0.10	327,502
DE MM	1	1	0.97	33,953	0.15	100,879
Hybrid	1	1	0.88	59,888	0.20	289,442
DE MM	4	0 or 1	0.05	119,082	0.00	——
Hybrid	4	0 or 1	0.08	154,637	0.02	678,361
DE MM	4	1	0.39	85,666	0.02	284,259
Hybrid	4	1	0.18	170,282	0.01	346,042

Two close optimal solutions with a powered Earth flyby have been found; the best mission ($\Delta V = 7.211$ km/s) has a single deep-space impulse just before the powered Earth flyby; a close suboptimal solution ($\Delta V = 7.218$ km/s) without the deep-space impulse also exists. These trajectories, shown in Figs. 2 and 3, respectively, perform more than one complete revolution during the Venus–Earth leg and cannot be found by methods that rely on the solution of Lambert's problem unless the correct number of revolutions is specified for the leg. In fact, Lambert's problem may have more than one solution connecting the same points with the same trip time; these solutions require different velocities at the leg extremities and, consequently, the mission ΔV depends on which solution is picked. Three approaches are possible: 1) an additional optimization variable, which specifies the number of complete revolutions (n_{rev}) is introduced (this approach is not pursued here); 2) n_{rev} is fixed in advance; 3) n_{rev} is related to the leg trip time and relative positions of the planets. In this paper, the number of complete revolutions of the first two legs is fixed at zero unless the trip time is longer than 1 year and the transfer angle (measured in the motion direction and taken between 0 and 360 deg) is lower than 90 deg; in this case, one complete revolution is allowed and the left branch solution [30] is picked. For the sake of comparison, the same analysis is also carried out by always assuming one complete revolution during the Venus–Earth leg.

The DE and hybrid algorithms have been tested with the same population sizes as the previous example, while performing 1,000,000 function evaluations when four impulses are allowed and 500,000 function evaluations for the simplified one-impulse problem. The results are compared in Table 6. The best solutions perform more than one revolution during the Venus–Earth leg, and the performance of the algorithms is obviously improved when only this kind of trajectories is considered. In fact, a large number of solutions with a short leg time length are excluded, and convergence toward the correct region of the search space is favored. If the number of revolutions is related to the leg time length and short solutions are not excluded, other local optima may be (and are quite frequently) found, thus reducing the efficiency of the algorithms. Obviously, the problem with only one deep-space impulse allowed is simpler and the efficiency is quite high, whereas the algorithms may suffer when four impulses are allowed.

The presence of two close local minima is challenging for the optimization algorithms, because the difference in terms of ΔV is less than 1%. In addition to the efficiency and number of function evaluations required to reach a ΔV value within 1% of the global optimum (ϵ and N_{avg}, the attainment of either local optimum is considered a successful run), the capability of identifying the correct optimum is examined by means of the efficiency and number of function evaluations to attain $\Delta V < 7.218$ (i.e., the value of the suboptimal solution without the deep-space impulse), $\bar{\epsilon}$, and $\bar{N}_{\text{avg}}$. Differential evolution typically shows a larger ϵ compared with the hybrid algorithm and requires a number of function evaluations N_{avg} that is remarkably lower; only for the most challenging problem, when up to four deep-space impulses are allowed and the number of revolutions of the Venus–Earth leg is not specified, the hybrid algorithm presents a larger efficiency. On the other hand, the hybrid algorithm seems more capable of finding the correct optimum, as shown by the larger $\bar{\epsilon}$.

V. Conclusions

A hybrid evolutionary algorithm, which synergistically employs a genetic algorithm, differential evolution, and particle swarm optimization, has been applied to the design of interplanetary trajectories with multiple impulses and gravity assists. A set of optimization variables that makes reference to ballistic trajectories joining the relevant planets is introduced to determine the location of the impulse during each leg; this particular position of the problem improves the convergence toward the global optimum. Particular care is, however, required to avoid the exclusion of significant solutions.

For simple problems, the performance of the algorithm is comparable to that of differential evolution, with longer computational times due to the necessary increase of the population size. For more complex problems with a larger number of variables, differential evolution may experience difficulties in finding the solution even with optimal settings of the algorithm parameters; the introduction of the mass mutation operator effectively improves the performance of DE. The hybrid method, even with generic settings of the algorithm parameters, also exhibits good performance for the most complex problems and is therefore well suited for the preliminary design of multiple-impulse, multiple-gravity-assist missions. Convergence to the optimum is sometimes slow, and the use of a different optimization method to refine the solutions provided by the evolutionary algorithm may be considered. Changes in the algorithm settings (namely, selection criteria for the genetic algorithm, strategies for the differential evolution and particle swarm optimization algorithms, and parameters ruling the mass mutation operator) and a different choice of optimization variables may also be tried, should the baseline algorithm described here not provide satisfactory results.

References

[1] Gage, P., Braun, R., and Kroo, I., "Interplanetary Trajectory Optimization Using a Genetic Algorithm," *Journal of the Astronautical Sciences*, Vol. 43, No. 1, 1995, pp. 59–76.

[2] Rauwolf, G., and Coverstone-Carroll, V., "Near-Optimal Low-Thrust Orbit Transfers Generated by a Genetic Algorithm," *Journal of Spacecraft and Rockets*, Vol. 33, No. 6, 1996, pp. 859–862. doi:10.2514/3.26850

[3] Rauwolf, G., and Coverstone-Carroll, V., "Near-Optimal Low-Thrust Trajectories via Micro-Genetic Algorithms," *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 1, 1997, pp. 196–198. doi:10.2514/2.4020

[4] Hartman, J., Coverstone-Carroll, V., and Williams, S., "Optimal Interplanetary Spacecraft Trajectories via Pareto Genetic Algorithm," *Journal of the Astronautical Sciences*, Vol. 46, No. 3, 1998, pp. 267–282.

[5] Crain, T., Bishop, R., Fowler, W., and Rock, K., "Interplanetary Flyby Mission Optimization Using a Hybrid Global-Local Search Method," *Journal of Spacecraft and Rockets*, Vol. 37, No. 4, 2000, pp. 468–474. doi:10.2514/2.3607

[6] Woo, B., Coverstone-Carroll, V., and Cupples, M., "Low-Thrust Trajectory Optimization Procedure for Gravity-Assist, Outer-Planet Missions," *Journal of Spacecraft and Rockets*, Vol. 43, No. 1, 2006, pp. 121–129. doi:10.2514/1.14665

[7] Rosa Sentinella, M., and Casalino, L., "Genetic Algorithm and Indirect Method Coupling for Low-Thrust Trajectory Optimization," AIAA Paper 06-4468, June 2006.

[8] Dachwald, B., and Wie, B., "Solar Sail Kinetic Energy Impactor Trajectory Optimization for an Asteroid-Deflection Mission," *Journal of Spacecraft and Rockets*, Vol. 44, No. 4, 2007, pp. 755–764. doi:10.2514/1.22586

[9] Biesbroek, R., "Study of Genetic Algorithm Settings for Trajectory Optimisation," International Astronautical Federation Paper IAF-03-A.P.30, Sept.–Oct. 2003.

[10] Biesbroek, R., "A Comparison of Differential Evolution Method with Genetic Algorithms for Orbit Optimisation," International Astronautical Federation Paper IAF-06-C1.4.02, Oct. 2006.

[11] Myatt, D., Becerra, V., Nasuto, S., and Bishop, J., "Advanced Global Optimization Tools for Mission Analysis and Design," ESA, ESA Ariadna ITT AO4532/ 18138/04/NL/MV, Call 03/4101, 2004.

[12] Di Lizia, P., and Radice, G., "Advanced Global Optimization Tools for Mission Analysis and Design," European Space Agency, ESA Ariadna ITT AO4532/18139/04/NL/MV, Call 03/4101, 2004.

[13] 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

[14] Bessette, C., and Spencer, D., "Optimal Space Trajectory Design: A Heuristic-Based Approach," *Advances in the Astronautical Sciences, Univelt Inc., San Diego, CA*, Vol. 124, 2006, pp. 1611–1628; also American Astronautical Society, Paper AAS 06-197.

[15] Izzo, D., Becerra, V., Myatt, D., Nasuto, S., and Bishop, J., "Search Space Pruning and Global Optimization of Multiple Gravity Assist Spacecraft Trajectories," *Journal of Global Optimization*, Vol. 38, No. 2, 2007, pp. 283–296. doi:10.1007/s10898-006-9106-0

[16] Vinko, T., Izzo, D., and Bombardelli, C., "Benchmarking Different Global Optimisation Techniques for Preliminary Space Trajectory Design," International Astronautical Federation, Paper IAC-07-A1.3.01, Oct. 2007.

[17] Olds, A., Kluever, C., and Cupples, M., "Interplanetary Mission Design Using Differential Evolution," *Journal of Spacecraft and Rockets*, Vol. 44, No. 5, 2007, pp. 1060–1070. doi:10.2514/1.27242

[18] Rosa Sentinella, M., "Comparison and Integrated Use of Differential Evolution and Genetic Algorithms for Space Trajectory Optimisation," *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, Inst. of Electrical and Electronics Engineers, New York, 2007, pp. 973–978.

[19] Rosa Sentinella, M., "Development Of New Procedures and Hybrid Algorithms for Space Trajectories Optimisation," Ph.D. Thesis, Politecnico di Torino, Turin, Italy, April 2008.

[20] Holland, J., *Adaption in Natural and Artificial Systems*, Univ. of Michigan Press, Ann Arbor, MI, 1975.

[21] Mitchell, M., *Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA, 1996.

[22] Goldberg, D., *Genetic Algorithms in Engineering Design*, Wiley, New York, 1997.

[23] Goldberg, D., and Deb, K., "A Comparison of Selection Schemes Used in Genetic Algorithms," *Foundations of Genetic Algorithms*, edited by G. Rawlins, Vol. 1, Morgan Kaufmann, San Francisco, 1991, pp. 450–457.

[24] Bramlette, M., "Initialization, Mutation, and Selection Methods in Genetic Algorithms for Function Optimization," *Proceedings of the Fourth International Conference on Genetic Algorithms*, edited by R. K. Belew, and L. B. Booker, Morgan Kaufmann, San Mateo, CA, July 1991, pp. 100–107.

[25] Storn, R., and Price, K., "Differential Evolution—A Simple and Efficient Adaptive Scheme for Global Optization over Continuos Spaces," International Computer Science Institute, TR-95-012, March 1995.

[26] Storn, R., "On the Usage of Differential Evolution for Function Optimization," *1996 Biennial Conference of the North American Fuzzy Information Processing Society*, North American Fuzzy Information Processing Society, Berkeley, CA, 1996, pp. 519–523.

[27] Kennedy, J., and Eberhart, R., "Particle Swarm Optimisation," *Proceedings of the IEEE International Conference on Neural Networks*, Institute of Electrical and Electronics Engineers, Piscataway, NJ, 1995, pp. 1942–1948.

[28] Eberhart, R., and Kennedy, J., "A New Optimizer Using Particle Swarm Theory," *Sixth International Synposium on Micro Machine and Human Science*, Institute of Electrical and Electronics Engineers, Piscataway, NJ, 1995, pp. 39–43.

[29] Trelea, I. C., "The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection," *Information Processing Letters*, Vol. 85, No. 6, 2003, pp. 317–325. doi:10.1016/S0020-0190(02)00447-7

[30] Battin, R. H., *An Introduction to the Mathematics and Methods of Astrodynamics*, AIAA, New, York, 1987, pp. 192–195, 295–341.

C. McLaughlin
Associate Editor