Analysis of some Global Optimization Algorithms for Space Trajectory Design

M. Vasile* and E. Minisci†

University of Glasgow, Glasgow, G12 8QQ, United Kingdom

M. Locatelli‡

Università degli Studi di Parma, Parma, 43124, Italy

In this paper, we analyze the performance of some global search algorithms on a number of space trajectory design problems. A rigorous testing procedure is introduced to measure the ability of an algorithm to identify the set of ϵ-optimal solutions.

From the analysis of the test results, a novel algorithm is derived. The development of the novel algorithm starts from the redefinition of some evolutionary heuristics in the form of a discrete dynamical system. The convergence properties of this discrete dynamical system are used to derive a hybrid evolutionary algorithm that displays very good performance on the particular class of problems presented in this paper.

Nomenclature

A	Search algorithm
A_g	Archive
C_R	Crossover probability
c	Dissipation constant
c_1, c_2	Weighting parameters in Particle Swarm Optimization
D	Search space
d_{err}	Confidence interval

*Senior Lecturer, Aerospace Engineering, James Watt South Building, AIAA Member.

†Research Fellow, Aerospace Engineering, James Watt South Building.

‡Associate Professor, Dipartimento di Ingegneria Informatica, via G. P. Usberti, 181/A.

$\mathbf{e}$	Mask vector for the selection of solution vector components
f	Objective function
F	Amplification control factor in Differential Evolution
j_s	Number of successful runs
$\mathbf{J}$	Transition matrix or mapping function in the search process
M_p	Mutation probability
N	Total number of function evaluations
N_L	Number of transfer legs
N_P	Number of planets
N_p	Normal distribution function
N_ρ	Neighborhood of a solution in the search space
n	Total number of repeated runs
n_{pop}	Size of the population P
n_{feval}	Function evaluation counter
P	Population
p	Optimization problem
p_s	Success rate
r_1, r_2	Random numbers in Particle Swarm Optimization
r_p	Normalized radius of the pericenter
S	Selection function
T	Transfer time, day
t_0	Departure time, MJD2000
$\mathbf{u}$	Control vector in the search process
$\mathbf{v}$	Variation of the solution vector or velocity in the search space
w	Weighting factor
$\mathbf{x}$	Solution vector
$\mathbf{x}_c$	Candidate point
$\mathbf{x}_l$	Local minimum
α	Position of a deep space manoeuvre as percentage of the transfer time
γ	Attitude angle of the gravity assist hyperbola, km/s
Δ	Variation of the objective function
Δv	Variation in velocity, km/s
δ	Escape declination, rad
δ_r	Reduction in the objective value
ϵ	Modulus of the difference between two solutions
θ	Escape right ascension, rad
θ_p	True proportion of success

ν Velocity limiting factor

ρ Size of the neighborhood N_ρ

Subscript

h Subseries index counter

i, j Variable numbers

l Local search algorithm or local minimum

k Search process iteration number

Superscript

T Transpose

I. Introduction

In the last decade many authors have used global optimization techniques to find optimal solutions to space trajectory design problems. Many different methods have been proposed and tested on a variety of cases from pure Genetic Algorithms[1–4] to Evolutionary Strategies to Differential Evolution[5] to hybrid methods.[8] The general intent is to improve over the pure grid or enumerative search. Sometimes the actual advantage of using a global method is difficult to appreciate, in particular when stochastic techniques are used. On one hand, a stochastic search provides a non-zero probability to find an optimal solution even with a small number of function evaluations while on the other hand, the repeatability of the result and therefore the reliability of the method can be questionable.

The first actual assessment of the suitability of global optimization methods to the solution of space trajectory design problems can be found in two recent studies.[6,7] The first study[7] presented a small set of test problems mainly focusing on multiple gravity assist trajectories, while the other[6] included results for low-thrust transfers using a wide range of global optimizers. One of the interesting outcomes of both studies was that Differential Evolution[13] (DE), belonging to a subclass of Evolutionary Algorithms, performed particularly well on most of the problems, compared to other methods. It should be noted that the application of global methods to space trajectory problems often considers the problem as a black-box with limited exploitation of problem characteristics. Ad hoc techniques however can exploit problem characteristics,[7] providing a sensible improvement over the direct application of general purpose methods.

In this paper, a number of global search methods are tested on a selected set of space trajectory problems. A rigorous testing procedure is proposed to identify the ability of an algorithm to identify the set of ϵ-optimal solutions (i.e. the set of solutions in the criteria space at distance ϵ from the expected global optimum).

The analysis of the results led to the identification of common patterns in the relationship between solution method and problem features. From this, a hybrid algorithm was derived that blends together some evolutionary heuristics with the basic search strategy of Monotonic Basin Hopping[16,17] (MBH). The new hybrid algorithm significantly improves the performance of both standard DE and MBH on the class of problems presented in the paper.

II. Problem Description

We consider a benchmark comprised of four different test cases of increasing complexity: a direct bi-impulsive transfer from the Earth to an asteroid, a transfer to Mars via a gravity assist maneuver around Venus, and a multi-gravity assist transfer to Saturn both without, and with mid-course manoeuvres. In all of these cases, the objective is to minimize the total change in the velocity of the spacecraft due to all propelled maneuvers, or total Δv.

A. Bi-impulsive Earth-Apophis Transfer

A simple but significant test case is to find the best launch date t_0 and time of flight T to transfer a spacecraft from the Earth to the asteroid Apophis (99942 MN4). The transfer is computed as the solution of a Lambert's problem.[11] The objective function for this problem is the sum of the change in departure velocity Δv_0 and arrival velocity Δv_f:

$$f(\mathbf{x}) = \Delta v_0 + \Delta v_f \tag{1}$$

with the solution vector:

$$\mathbf{x} = [t_0, T]^T \tag{2}$$

The search space D is a box defining the limits of the two components in the solution vector. In particular, the launch date from the Earth was taken over the interval [3653, 10958] MJD2000 (where MJD2000 measures the number of elapsed days since 01 January 2000), and the time of flight was taken over the interval [50, 900] days.
The known best solution in D is $f_{best} = 4.3746$ km/s, with $\mathbf{x}_{best} = [10027.6216, 305.12163]^T$.

B. Earth-Venus-Mars Transfer with DSM's

The second test case consists of a transfer from Earth to Mars with a gravity assist manoeuvre at Venus and a deep-space manoeuvre (DSM) after Venus. This is the simplest instance of a multi-gravity assist trajectory with deep-space manoeuvres (MGA-DSM). The MGA model is taken from Vasile et al.[9]
Given N_P planets and the number of legs of the trajectory $N_L = N_P - 1$, the model yields

the total Δv to reach the destination given the solution vector:

$$\mathbf{x} = [v_0, \overline{\theta}, \overline{\delta}, t_0, \alpha_1, T_1, \gamma_1, r_{p,1}, \alpha_2, T_2, ..., \gamma_i, r_{p,i}, T_{i-1}, \alpha_{i-1}, ..., \gamma_{N_L-1}, r_{p,N_L-1}, \alpha_{N_L}, T_{N_L}] \quad (3)$$

where t_0 is the departure date, v_0 is the asymptotic escape velocity, and $\overline{\theta}$ and $\overline{\delta}$ are:

$$\overline{\theta} = \frac{\theta}{2\pi} \qquad \overline{\delta} = \frac{\cos(\delta + \pi/2) + 1}{2}$$

The angles δ and θ are, respectively, the declination and right ascension of the escape velocity with respect to a local reference. The local reference frame here has the x axis aligned with the velocity vector of the planet, the z axis normal to orbital plane of the planet and the y axis completing the coordinate frame. The variable T_i is the transfer time from one planet to another, where α_i is the fraction of the transfer time at which a deep space manoeuvre can occur. The angle γ_i represents the attitude of the planetocentric hyperbola of the i^{th} gravity assist manoeuvre and $r_{p,i}$, the ratio between the radius of its pericenter and the radius of the planet.

With the model of Vasile et al.,[9] the design of the multi-gravity assist transfer can be transcribed into a general nonlinear programming problem, with simple box constraints, of the form:

$$\min_{\mathbf{x} \in D} f(\mathbf{x}) \quad (4)$$

For the tests in this paper, the objective function $f(\mathbf{x})$ is:

$$f(\mathbf{x}) = v_0 + \sum_{i=1}^{N_p} \Delta v_i + \Delta v_f \quad (5)$$

where Δv_i is the velocity change due to the DSM in the i^{th} leg, and Δv_f is the maneuver needed to inject the spacecraft into the final orbit.

For a transfer to Mars via Venus, the solution vector in Eq. (3) has six parameters, or dimensions. The initial velocity v_0 with respect to the Earth is not a free parameter, but is computed as the result of the Lambert's problem for the Earth-Venus leg. Therefore we can define the following reduced solution vector:

$$\mathbf{x} = [t_0, T_1, \gamma_1, r_{p,1}, \alpha_2, T_2] \quad (6)$$

Since the initial velocity is not a free parameter, v_0 is defined as the modulus of the vector difference between the velocity of the Earth and the velocity of the spacecraft at time t_0. The final Δv_f is the Δv needed to inject the spacecraft into an ideal operative orbit around Mars, with a pericenter radius of 3950 km and an eccentricity of 0.98. This choice does

not alter the nature of the problem but scales down the contribution of the last impulsive manoeuvre.

The search space D is defined by the following intervals: $t_0 \in [3650, 9129]$ MJD2000, $T_1 \in [50, 400]$ d, $\gamma_1 \in [-\pi, \pi]$, $r_{p,1} \in [1, 5]$, $\alpha_2 \in [0.01, 0.9]$, $T_2 \in [50, 700]$ d. The best known solution for this problem in the given search space D is $f_{best} = 2.9811$ km/s, $\mathbf{x}_{best} = [4472.0133, 172.2893, 2.9784, 1, 0.5094, 697.6100]^T$.

C. Earth-Saturn Transfer

The third test is a multi gravity assist trajectory from the Earth to Saturn following the sequence Earth-Venus-Venus-Earth-Jupiter-Saturn (EVVEJS). Gravity assist maneuvers are modeled through a linked-conic approximation with powered maneuvers,[7] i.e., the mismatch in the outgoing velocity is compensated through a Δv maneuver at the pericenter of the gravity assist hyperbola for each planet. No deep space maneuvers are allowed, with each planet-to-planet transfer computed as the solution of a Lambert's problem. Therefore, the whole trajectory is completely defined by the departure time t_0 and the transfer time for each leg T_i, with $i = 1, ..., N_P - 1$.

The normalized radius of the pericenter $r_{p,i}$ of each swing-by hyperbola is derived a posteriori once each powered swing-by manoeuvre is computed. Thus, a constraint on each pericenter radius has to be introduced during the search for an optimal solution. In order to take into account this constraint, the objective function is augmented with the weighted violation of the constraints:

$$f(\mathbf{x}) = \Delta v_0 + \sum_{i=1}^{N_p-2} \Delta v_i + \Delta v_f + \sum_{i=1}^{N_p-2} w_i (r_{p,i} - r_{pmin,i})^2 \tag{7}$$

for a solution vector:

$$\mathbf{x} = [t_0, T_1, T_2, T_3, T_4, T_5]^T \tag{8}$$

The final Δv_f is the Δv needed to inject the spacecraft into an ideal operative orbit around Mars with a pericenter radius of 108950 km and an eccentricity of 0.98. The weighting functions w_i are defined as follows:

$$\begin{aligned} w_i &= 0.005[1 - \text{sign}(r_{p,i} - r_{pmin,i})], i = 1, ..., 3 \\ w_4 &= 0.0005[1 - \text{sign}(r_{p,4} - r_{pmin,4})] \end{aligned} \tag{9}$$

with the minimum normalized pericenter radii $r_{pmin,1} = 1.0496$, $r_{pmin,2} = 1.0496$, $r_{pmin,3} = 1.0627$ and $r_{pmin,4} = 9.3925$. For this case the dimensionality of the problem is six, with the search space D defined by the following intervals: $t_0 \in [-1000, 0]$MJD2000, $T_1 \in [30, 400]$d,

$T_2 \in [100, 470]$d, $T_3 \in [30, 400]$d, $T_4 \in [400, 2000]$d, $T_5 \in [1000, 6000]$d. The best known solution is $f_{best} = 4.9307$ km/s, with $\mathbf{x}_{best} = [-789.753, 158.2993, 449.3859, 54.7060, 1024.5896, 4552.7054]^T$.

D. Earth-Saturn Transfer with DSM's

The fourth test case is also a multi gravity assist trajectory from the Earth to Saturn following the sequence Earth-Venus-Venus-Earth-Jupiter-Saturn (EVVEJS), but with a deep space manoeuvre allowed during the transfer from one planet to another. Although from a trajectory design point of view, this problem is similar to the third test case, the model is substantially different and therefore represents a different problem from a global optimization point of view. Since the transcription of the same problem into different mathematical models can affect the search for the global optimum, it is interesting to analyze the behavior of the same set of global optimization algorithms applied to two different transcriptions of the same trajectory design problem.

The trajectory model is essentially the same as that presented for the second test case save that Δv_f here, is the modulus of the vector difference between the velocity of the spacecraft and the velocity of Saturn at arrival. For this case the dimensionality of the problem is 22, with the search space D defined by the following intervals: $t_0 \in [-1000, 0]$MJD2000, $v_0 \in [3, 5]$km/s, $\bar{\theta} \in [0, 1]$, $\bar{\delta} \in [0, 1]$, $T_1 \in [100, 400]$d, $T_2 \in [100, 500]$d, $T_3 \in [30, 300]$d, $T_4 \in [400, 1600]$d, $T_5 \in [800, 2200]$d, $\alpha_1 \in [0.01, 0.9]$, $\alpha_2 \in [0.01, 0.9]$, $\alpha_3 \in [0.01, 0.9]$, $\alpha_4 \in [0.01, 0.9]$, $\alpha_5 \in [0.01, 0.9]$, $r_{p,1} \in [1.05, 6]$, $r_{p,2} \in [1.05, 6]$, $r_{p,3} \in [1.15, 6.5]$, $r_{p,4} \in [1.7, 291]$, $\gamma_1 \in [0, 2\pi]$, $\gamma_2 \in [0, 2\pi]$, $\gamma_3 \in [0, 2\pi]$, $\gamma_4 \in [0, 2\pi]$,. The best known solution is $f_{best} = 8.3889$ km/s, $\mathbf{x}_{best}$=[-780.9179, 3.2753, 0.7825, 0.3787, 169.1319, 424.1324, 53.2965, 589.7383, 2199.9865, 0.7958, 0.5301, 0.1260, 0.0106, 0.0382, 1.3556, 1.0500, 1.3070, 71.3749, 3.1584, 3.5304, 3.1256, 3.0842]^T.

Note that prior to running each test, we normalized the search space for each of the trajectory models so that D is a unit hypercube with each component of the solution vector belonging to the interval [0,1].

III. Optimization Algorithm Description

We tested five stochastic global search algorithms: Differential Evolution (DE) and Genetic Algorithms (GA) that belong to the generic class of Evolutionary Algorithms (EA), Particle Swarm Optimization (PSO) that belongs to the class of agent-based algorithms, and Multi-start (MS) and Monotonic Basin Hopping (MBH) that are based on multiple local searches with a gradient method.

1. Genetic Algorithms

Genetic Algorithms[14] (GA) are stochastic search methods that take their inspiration from natural selection and 'survival of the fittest' in the biological world. Each iteration of a GA involves a competitive selection that eliminates poor solutions. The solutions with a high level of fitness are recombined with other solutions by swapping parts of one solution with those from another. Solutions are also mutated by making a small change to a single element, or a small number of elements, of the solution. Recombination and mutation are used to generate new solutions that are biased towards regions of the search space for which good solutions have already been discovered.
In the following tests, we use the Matlab Genetic Algorithm Toolbox GATBX.[24] Only the influence of the population size was considered; specifically 100, 200 and 400 individuals for the bi-impulse test case and 200, 400 and 600 individuals for the other three cases. Single values were used for the crossover and mutation probability, $C_r = 0.5$ and $M_p = 1/d$ respectively, where d denotes the dimension of the problem. In the following, GA are identified by the population size, for example GA100 stands for Genetic Algorithms with 100 individuals. The crossover and mutation probabilities are the default values suggested for the use of the toolbox.

2. Differential Evolution

The main idea behind Differential Evolution[13] (DE) is to generate the variation vector $\mathbf{v}_{i,k+1}$ of a solution vector $\mathbf{x}_{i,k+1}$ by taking the weighted difference between two additional solution vectors, randomly chosen within a population of solutions, and to add that difference to the vector difference between $\mathbf{x}_{i,k}$ and a third solution vector:

$$\mathbf{v}_{i,k+1} = \mathbf{e}\left[(\mathbf{x}_{i_3,k} - \mathbf{x}_{i,k}) + F(\mathbf{x}_{i_2,k} - \mathbf{x}_{i_1,k})\right] \tag{10}$$

where i_1 and i_2 are integer numbers randomly chosen within the interval $[1,\ n_{pop}] \subset \mathbb{N}$ of indices of the population, and $\mathbf{e}$ is a mask containing a random set of 0's and 1's according to:

$$e(j) = \begin{cases} 1 \Rightarrow r \leq C_R \\ 0 \Rightarrow r > C_R \end{cases} \tag{11}$$

where $j = 1, ..., n$. The value for r is taken from a random uniform distribution $r \in U[0, 1]$ and C_R a constant. The index i_3 can be chosen at random (exploration strategy) or can be the index of the best solution vector $\mathbf{x}_{best}$ (convergence strategy). Selecting the best solution vector versus a random one significantly changes the convergence speed of the algorithm. The selection process is generally deterministic and simply preserves the variation of $\mathbf{x}_{i,k}$

only if $f(\mathbf{x}_{i,k} + \mathbf{v}_{i,k+1}) < f(\mathbf{x}_{i,k})$.
We considered six different settings for the DE resulting from combining: three sets of populations $[5\,d, 10\,d, 20\,d]$ where d is the dimensionality of the problem, and two strategies (convergence and explore). The step-size and crossover probability, $F = 0.75$ and $C_R = 0.8$ respectively, were taken based on those in common use. In the following, the six settings will be denoted by DE5c, DE10c, DE20c for the ones using the convergence strategy and by DE5e, DE10e, DE20e for the ones using the exploration strategy.

3. Particle Swarm Optimization

Particle swarm optimization[12] (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995, and was inspired by the social behavior of bird flocking or fish schooling. In PSO the potential solutions, called particles, 'fly' through the problem space by following the current optimum particles. Each particle keeps track of its coordinates in the problem space, which are associated with the best solution it has achieved so far. The particle swarm optimization concept consists of changing the velocity of each particle i at each iteration according to the close-loop control mechanism:

$$\mathbf{v}_{i,k+1} = w\mathbf{v}_{i,k} + \mathbf{u}_{i,k} \tag{12}$$

where w is a weighting function, that in this implementation is proportional to the number of iterations k,

$$w = [0.4 + 0.8(k_{max} - k)/(k_{max} - 1)] \tag{13}$$

The control $\mathbf{u}_{i,k}$ has the form:

$$\mathbf{u}_{i,k} = c_1 r_1(\mathbf{x}_{gi,k} - \mathbf{x}_{i,k}) + c_2 r_2(\mathbf{x}_{go,k} - \mathbf{x}_{i,k}) \tag{14}$$

where $\mathbf{x}_{gi,k}$ is the position of the best solution found by particle i (individualistic component) and $\mathbf{x}_{go,k}$ is the position of the best particle in the swarm (social component), with random numbers r_1, r_2 and coefficients c_1, c_2 used to weight the social and individualistic components. The position of a particle in the search space is then:

$$\mathbf{x}_{i,k+1} = \mathbf{x}_{i,k} + \nu\mathbf{v}_{i,k+1} \tag{15}$$

with

$$\nu = \min\left([v_{max}, v_{i,k+1}]\right)/v_{i,k+1} \tag{16}$$

Eq. (16) represents a limit sphere around the point $\mathbf{x}_{i,k}$ at stage k of the search process. The search is continued until a stopping criterion is satisfied.

The process has two stochastic components given by the two random numbers r_1 and r_2. The term $c_1 r_1(\mathbf{x}_{gi,k} - \mathbf{x}_{i,k})$ is an elastic component that tends to recall the particle back to its old position. The term $c_2 r_1(\mathbf{x}_{go,k} - \mathbf{x}_{i,k})$ instead is driving the whole population toward convergence. There is no selection mechanism.
For the PSO algorithm, nine different settings were considered, resulting from the combination of: three sets of population $[5\,d, 10\,d, 20\,d]$, three values for the maximum velocity bound $v_{max} \in [0.5, 0.7, 0.9]$, and single values for weights $c_1 = 1$ and $c_2 = 2$. In the following, the three population sets will be denoted by PSO5, PSO10, PSO20, with two additional digits appended to identify the value of the v_{max}, for example PSO505 is the PSO algorithm with $5d$ particles and a limit on the max velocity $v_{max} = 0.5$.

4. Multi-Start

The simple idea behind multi-start (MS) algorithms is to pick a number of points in the search space and start a local search from each one of them. The local search can be performed with a gradient method. For the following tests, points were selected randomly following a Latin Hypercube distribution.

5. Monotonic Basin Hopping

Monotonic Basin Hopping (MBH) was first applied to molecular conformation problems,[16,17] and later extended to more general global optimization problems.[18,19] In its basic version it is quite similar to MS in that it is based on multiple local searches. The only difference is the distribution of the starting points for local searches: while in MS these are randomly generated over the whole feasible region, in MBH they are generated in a neighborhood $N_\rho(\mathbf{x})$ of the current best local minimizer $\mathbf{x}$. The parameter ρ controls the size of the neighborhood; its choice is essential to the performance of the algorithm. Too low a value causes the points to be generated only within the basin of attraction of the current local minimizer; too large a value would causes MBH to behave like MS. A careful choice of ρ can lead to results which strongly outperform those of MS in spite of the strong similarity between the two algorithms. In this work, $N_\rho(\mathbf{x})$ is a hypercube with edge length 2ρ centered at $\mathbf{x}$. All test cases were performed with $\rho = 0.1$.
The MBH scheme used to compute the results in this paper is summarized in Algorithm 1.

IV. Testing Procedure

In this section we describe a rigorous testing procedure that can be used to assess the performance of a given algorithm.

Algorithm 1 MBH

1: Select a point $\mathbf{y}$ in the solution space D;
2: Run a local optimizer A_l from $\mathbf{y}$ and let $\mathbf{x}$ be the detected local minimum.
3: Set $n_{feval} = n_{feval} + eval$, where $eval$ denotes the number of function evaluations required by the local search.
4: Select a candidate point $\mathbf{x}_c \in N_\rho(\mathbf{x})$
5: Run a local optimizer A_l from $\mathbf{x}_c$ and let $\mathbf{x}_l$ be the local minimum found by A_l
6: Update $n_{feval} = n_{feval} + eval$
7: **if** $f(\mathbf{x}_l) < f(\mathbf{x})$ **then**
8: $\quad \mathbf{x} \leftarrow \mathbf{x}_l$
9: **end if**
10: **Termination** Unless $n_{feval} \geq n_{fevalmax}$, *goto* Step 4

In theory, given a limited computational effort, we would like to know the probability that an optimization algorithm A finds the global optimum of a given problem p. However, such a probability does not necessarily converge to 1, even for an infinite computational effort, if the algorithm is not globally convergent. Furthermore, the global optimum might be unknown. Thus, in practice, what we would like to know is the probability that, for a given computational effort, an algorithm finds solutions with a value of the objective function below an arbitrary threshold. In the specific case of space trajectory design, the threshold can be derived from the mission requirements, e.g., do not exceed a given total Δv. Finding the set of solutions below a given threshold is practically more useful than finding the global optimum, as the set defines one or more launch windows rather than a single launch opportunity.
If the number of function evaluations is used to quantify the computational effort, and an algorithm can generate, at each iteration, a solution with strictly positive probability within:

$$D_\epsilon = \{\mathbf{x} \in D | f(\mathbf{x}) < f(\mathbf{x}_{global}) + \epsilon\} \tag{17}$$

for any $\epsilon > 0$, then, for the number of function evaluations that goes to infinity, the algorithm converges to the global optimum with probability 1 (e.g., see the works of Rudolph [22] and Pinter [23] on the convergence of stochastic algorithms).
We can now define a procedure, summarized in Algorithm 2, to assess the ability of an algorithm to generate solutions in D_ϵ.
Given an algorithm A and a problem p, apply A to p, n times for a fixed number of function evaluations N_k; register the quantity $\phi(N_k, i)$, which is the minimum function value found by the i^{th} run of algorithm A applied to p for the number of function evaluations N_k; register the number of times, $j_{s,k}$, the quantity $\phi(N_k, i)$ is below the threshold tol_f; repeat for $N_k = N_1, ..., N_{max}$.
In the following tests, the threshold tol_f is defined as $tol_f = f(\mathbf{x}_{best}) + \epsilon$ where $\mathbf{x}_{best}$ is the

Algorithm 2 Testing Procedure

define the set $\{N_1, N_2, ..., N_k, ..., N_{max}\}$
for all $N_k \in \{N_1, ..., N_{max}\}$ **do**
    apply A to p for n times
    set $j_{s,k} = 0$
    for all $i \in \{1, ..., n\}$ **do**
        compute $\phi(N_k, i)$
        if $(\phi(N_k, i) < tol_f)$ **then** $j_{s,k} = j_{s,k} + 1$
        end if
    end for
end for

best known solution to a given problem. If during the tests a better solution is found, $\mathbf{x}_{best}$ is updated. Therefore, if an algorithm is globally convergent, $\mathbf{x}_{best}$ will eventually become the global optimum for $N_{max} \to \infty$. If an algorithm is not globally convergent, we assess its ability to generate solutions with a function value at an arbitrary distance $tol_f - f(\mathbf{x}_{global})$ from the global optimum.

The procedure described in Algorithm 2 only considers the computational cost to evaluate f, not the intrinsic computational cost of A. The intrinsic cost of A is related to its complexity and the number of pieces of information A is handling, and varies from algorithm to algorithm. Here we assume that the computational effort of the algorithms is dominated by the function evaluation and, therefore, we do not take into account the intrinsic cost.

The quantity j_s is the number of successes, and is used to define the success rate $p_s = j_s/n$. Properly defining the value of n is a key point, for example too low a value of n means there are an insufficient number of samples to allow for proper statistics. A proper value for n should produce little or no fluctuations on the estimated value of p_s, i.e., by increasing n the value of p_s should remain constant, or with only a small variation. The fluctuation of the success rate can be represented by a binomial probability density function (pdf), independent of the number of function evaluations, problem or type of optimization algorithm. This last characteristic implies that the test can be designed fixing a priori the number of runs n on the basis of the acceptable error on the estimation of the success rate. A typical starting point to determine the sample size for a binomial distribution is to assume that: the sample proportion p_s of successes (the success rate for a given n in our case) can be approximated with a normal distribution, i.e. $p_s \sim N_p\{\theta_p, \theta_p(1-\theta_p)/n\}$, where θ_p is the unknown true proportion of successes, and that the probability of p_s to be at distance d_{err} from θ_p is at least $1 - \alpha_b$. This leads to the expression:[10]

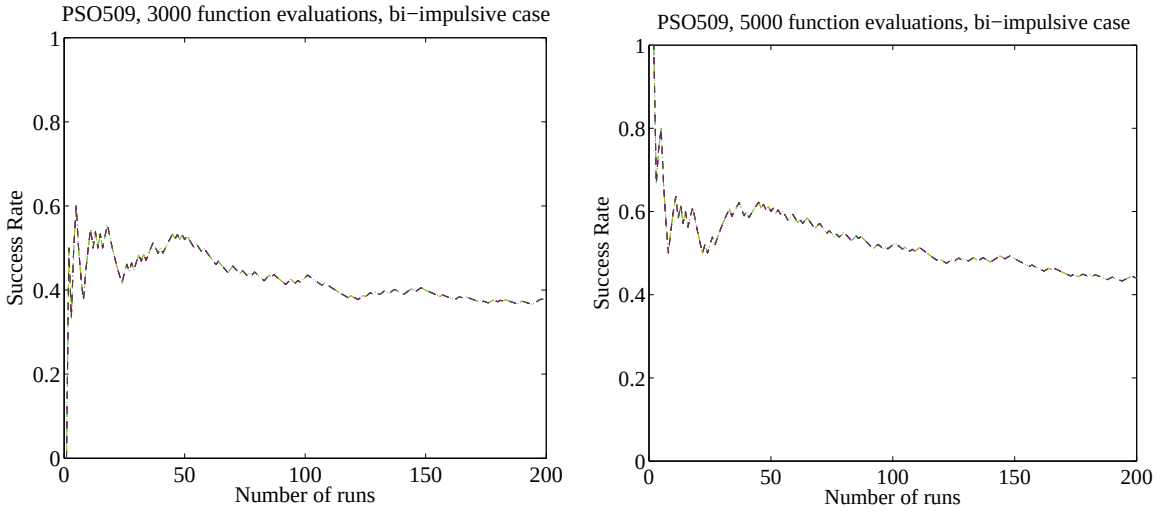
$$n \geq \theta_p(1-\theta_p)\chi^2_{(1),\alpha_b}/d^2_{err} \tag{18}$$

where $\chi^2_{(1),\alpha_b}$ is the χ^2 distribution function with 1 degree of freedom and computed at α_b. From Eq. (18) it is possible to derive the conservative rule:

$$n \geq 0.25\chi^2_{(1),\alpha_b}/d^2_{err} \tag{19}$$

obtained if $\theta_p = 0.5$. For our tests we required an error $d_{err} = 0.05$ with a 95% confidence ($\alpha_b = 0.05$), which, according to Eq. (19), yields $n \geq 176$. This was extended to 200 for all the tests in order to have a higher confidence in the result.

Fig. 1 shows the variation of the success rate as a function of n for the case of PSO applied to the solution of the bi-impulsive case. For $n \leq 50$, the success rate is oscillating and the confidence in the estimated value is poor. Increasing the number of runs to 200 gives a more stable value within the required confidence interval.



(a) (b)

Figure 1. The influence of sample size. The success rate is shown as function of the number of runs for the PSO applied to the solution of the bi-impulsive case: a) 3000 function evaluations, b) 5000 function evaluations.

A recent example of the use of the success rate can be found in the work of Olds and Kluewer,[5] where the authors use the success rate to assess the performance of Differential Evolution. They propose using 1000 runs in order to have an acceptable sample. According to the theory presented in this section, the confidence interval for 1000 runs would be $d_{err} = 0.020857$, which is confirmed by the good reproducibility of the results, testified by Olds and Kluewer.

A. Test Results

The results of the tests are summarized in Table 1, where the success rate (in percentage) is given for each of the 20 settings of the five stochastic solvers. The test cases are numbered: 1) bi-impulsive Earth-Apophis case (EA), 2) Earth-Venus-Mars transfer (EVM), 3) Earth-Venus-Venus-Earth-Jupiter-Saturn case with no DSM's (EVVEJS), 4) Earth-Venus-Venus-Earth-Jupiter-Saturn case with DSM's (EVVEJSDSM). The results in the table were computed for the maximum number of function evaluations for each of the tested cases. In particular, $n = 5000$ for the bi-impulsive case, 10^5 for the EVM case, 4×10^5 for the EVVEJS case with no DSM's and 1.25×10^6 for the EVVEJS case with DSM's. To compute the success rate, the following tol_f values were used: 4.3756 km/s for EA, 3 km/s for EVM, 5 km/s for EVVEJS and 8.5 km/s for EVVEJSDSM.

For both the EA and EVM cases, the algorithms DE10e and DE20e perform better than the other evolutionary algorithms, while algorithms GA100/200-GA200/400-GA400/600 appear to be the worst performing ones. Due to the binomial nature of the success and the adopted sample size, it is not possible to discriminate between algorithms for which difference in success rate is smaller than the expected error d_{err}. For example, algorithm DE20c and DE5c are considered to be equivalent to the algorithms PSO1007, PSO509. For the same reason, we can say that among the PSOs tested, PSO1007 and PSO509 perform better than PSO1005, while the remaining PSO algorithms all function at the same level.

Table 1. Success rate (in %) for the 20 algorithms on the four test cases.

Problem	DE5c	DE10c	DE20c	DE5e	DE10e	DE20e	PSO505	PSO1005	PSO2005	PSO507
1	14	30	35	45	77	85.5	35.5	34.5	41	39.5
2	5	5	5	15	25	37	4	3.5	8	4.5
3	2	0.5	1.5	0	0	0	0	0.5	0	0
4	1.5	1	0.5	0	0	0	0	0	0	0
	PSO1007	PSO2007	PSO509	PSO1009	PSO2009	GA100/200	GA200/400	GA400/600	MS	MBH
1	42.5	41	43.5	38.5	42	16	24	10.5	93	46.5
2	6	5.5	3.5	7	7.5	0.5	1	3.5	62.5	69.5
3	0	0	0	0	0	0	0	0.5	7	46.5
4	0	0	0	0	0	0	0	0	0.5	24

MS and MBH show remarkable performance in all the test cases, with the simple MS winning over all others in the EA case. For the EVVEJS case, all the algorithms but MBH appear practically unsuccessful. MBH is the only algorithm that can be practically used for this problem, with a success rate of 46.5%. The same happens for EVVEJSDSM case, when MBH is able to find the values below the threshold with a success rate of 24%.

In order to solve for uncertainty conditions, e.g., when the success rate appears uniformly

null, relaxing the tol_f value can be useful, as seen in Table 2.

Table 2. Successes rate (in %) for the 20 algorithms on the EVVEJSDSM test-case, with the threshold varying from 8.5 to 9.0 km/s

tol_f	DE5c	DE10c	DE20c	DE5e	DE10e	DE20e	PSO505	PSO1005	PSO2005	PSO507
8.5 km/s	1.5	1	0.5	0	0	0	0	0	0	0
8.6 km/s	2.5	1	2.5	0	0	0	0	0	0	0
8.7 km/s	7	5	6.5	0	0	0	0	0	0	0
8.8 km/s	8	6	2	6.5	0	0	0	0	0	0
8.9 km/s	8	6	5	7	0	0	0	0	0	0
9.0 km/s	8	6	6	7	0	0	0	0	0	0
	PSO1007	PSO2007	PSO509	PSO1009	PSO2009	GA100/200	GA200/400	GA300/600	MS	MBH
8.5 km/s	0	0	0	0	0	0	0	0	0.5	24
8.6 km/s	0	0	0	0	0	0	0	0	1.5	31.5
8.7 km/s	0	0	0	0	0	0	0	0	7	45
8.8 km/s	0	0	0	0	0	0	0	0	11	47
8.9 km/s	0	0	0	0	0	0	0	0	14	47.5
9.0 km/s	0	0	0	0	0	0	0	0	15.5	48.5

For the EVVEJSDSM case, raising the success threshold from 8.5 to 9 (see Table 2) identifies MBH and MS as the only algorithms practically able to find good solutions for this problem. However the results do not give any useful information in order to discriminate among the other algorithms. The DE's (particularly the DEc series) are able to find solutions with an objective value below 9 as well, but the success rate is so low that they cannot be considered useful to solve this problem.

B. Analysis of the Results

The collection of results from the tests can be used to deduce some properties of the problems within the benchmark and to predict the behavior of the solution algorithms. An understanding of the characteristics of the benchmark is required to generalize the result of the tests. In fact, any consideration on the performance of an algorithm is applicable only to problems with similar characteristics.

All local minima found in all the tests by the applied global methods were grouped according to the value of their objective function. Specifically, the range of values of the objective function for each test was divided in a finite number of levels, with each group of minima associated to a particular level.

Fig. 2 represents the distribution of the best function values returned by an optimization algorithm over 200 runs in a typical case: PSO applied to the EVM problem. The same figure represents also a Gaussian distribution with the mean and variance values equal to the

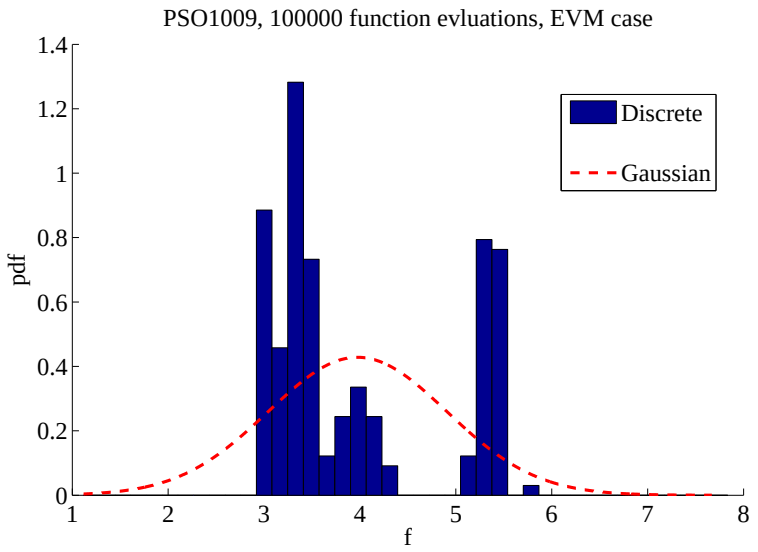
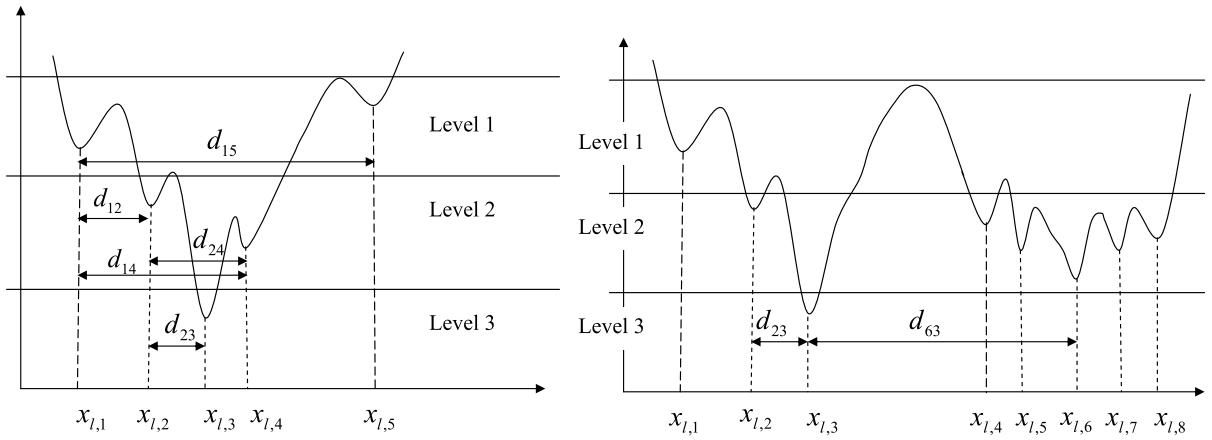


Figure 2. Probability density function for PSO applied to the solution of the EVM case: discrete vs. Gaussian (dashed) distribution.

mean and variance value of all the best function values. As it can be seen, the distribution is not Gaussian. Therefore, the average value can be far away from the results returned with a higher frequency from a given algorithm. In the same way, the variance is not a good indicator of the quality of the algorithm because a high variance together with a high mean value can correspond to the case in which 50% of the results are close to the global optimum with the other 50% far from it. Statistical tests, such as the t-test, that assume a Gaussian distribution of the sample can not be applied to correctly predict the behavior of an algorithm.

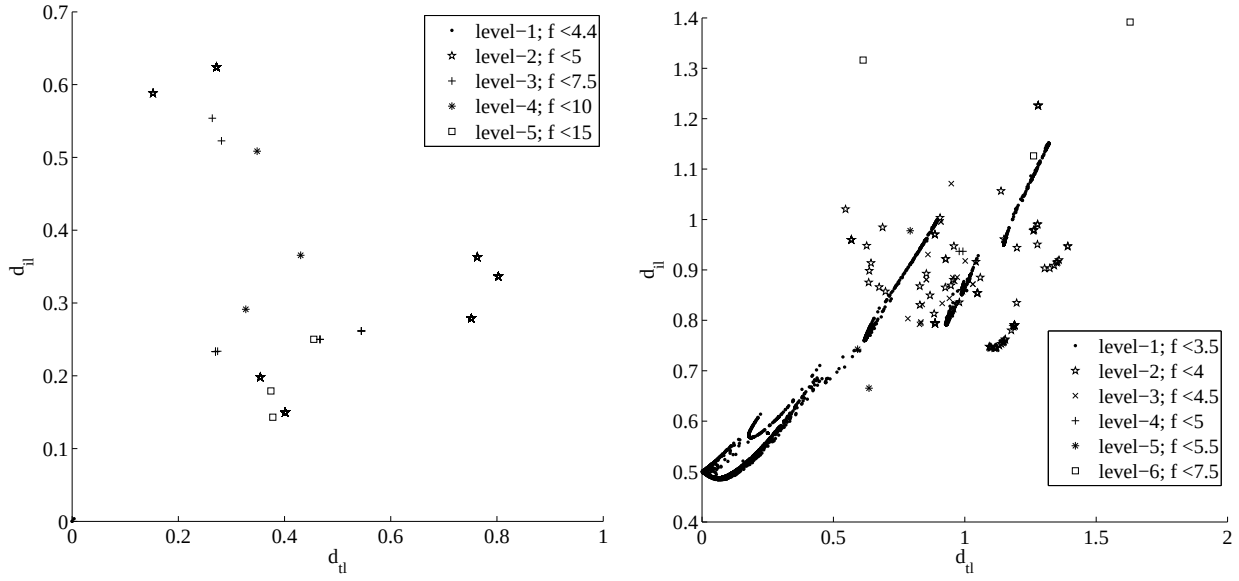
As a second step, we computed the average value of the relative distance of each local minimum with respect to all other local minima within the same level d_{il} (or intra-level distance), and the average value of the relative distance of each local minimum with respect to all other local minima in the lower level d_{tl} (or trans-level distance). The d_{tl} for the lowest level is the average distance with respect to the best known solution.

The values d_{il} and d_{tl} give an immediate representation of the diversity of the local minima and the probability of a transition from one level to another. More precisely, a cluster of minima with a large intra-level distance and a small trans-level distance suggests an easy transition to lower values of the objective function and a possible underlying funnel structure.[17] In particular in the case of funnel structures, the values of d_{tl} and d_{il} should progressively go to zero. A d_{il} that does not go to zero or clusters with different values of d_{tl}, are the cue to a possible underlying multi-funnel structure. Fig. 3 provide two illustrative examples. Fig. 3(a) represents a single funnel structure with five local minima $x_{l,i}$ where



(a) (b)

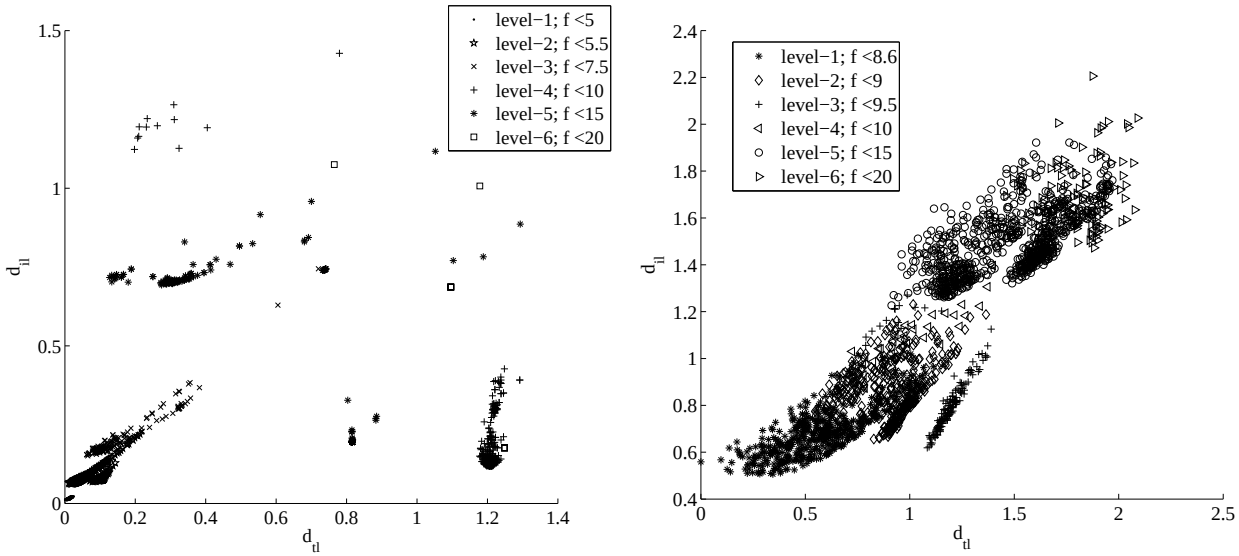
Figure 3. One dimensional example of a) single funnel structure and b) bi-funnel structure.



(a) (b)

Figure 4. Relative distance of the local minima for a) the bi-impulsive and b) the EVM case.

$i = 1, ..., 5$, and three levels. The intra-level distance at level 2, given by the distance $d_{24} = x_{l,2} - x_{l,4}$, is lower than $d_{15} = x_{l,1} - x_{l,5}$, the intra-level distance at level 1. The same is true for the trans-level distance at level 2, d_{23}, which is lower than the trans-level distance at level 1, $(d_{12} + d_{14})/2$, for minimum $x_{l,1}$. Fig. 3(b), instead, represents a bi-funnel structure.



(a) (b)

Figure 5. Relative distance of the local minima for the EVVEJS: a) without DSM's and b) with DSM's.

In this case, the minima around $x_{l,6}$ have an average intra-level distance lower than $x_{l,2}$ but a trans-level distance d_{63} much larger than d_{23}. Thus, the two minima $x_{l,2}$ and $x_{l,6}$ will appear on the d_{tl}-d_{il} graph with different values of d_{il} and d_{tl}. If the threshold of level 3 was increased above the objective value of $x_{l,6}$, then all minima of level 2 would have similar d_{tl}, but the d_{il} at level 3 would not go to zero.

This analysis method is an extension of the work of Reeves and Yamada,[21] and is used to concisely visualize the distribution of the local minima. The definition of the levels depends on the groups of minima of interest, and can be derived from mission constraints or by an arbitrary subdivision of the range of values of the objective function. Different subdivisions reveal different characteristics of the search space but give only equivalent cues on the transition probability.

Fig. 4(a) reveals that for the bi-impulsive case the minima are quite spread out. The EVM case in Fig. 4(b), instead, appears to have an almost continuous distribution of minima, although the minima of one level appear to be quite distant from the minima of the lower levels.

Fig. 5(a) seems to suggest that the problem has a structure similar to the one in Fig. 3(b), with the minima at level 5 belonging to two distinct clusters with substantially different d_{tl} and d_{il}. The clusters, corresponding to levels 1, 2 and 3, have values of d_{tl} and d_{il} both lower than 0.2, which suggests an easy transition from one level to an another. Thus, below an

objective function of 7.5 km/s there seems to be an underlying single funnel structure. Note that an easy transition among levels favors the search mechanism of MBH as demonstrated by the test results.

Fig. 5(b) shows that both d_{tl} and d_{il} progressively tend towards zero, up to a certain point, after which d_{tl} goes to zero while d_{il} remains almost unchanged. The figure suggests that, in the EVVESJ case with DSM, there is a single funnel structure for function values above 8.6 km/s, while below the minima are scattered and distant from each other.

V. A Dynamical System Perspective

The results in the previous sections suggest that the heuristics implemented in MBH are particularly effective on the set of tested problems compared to the evolutionary heuristics. MBH seems to exploit the funnel structures revealed by the d_{tl}-d_{il} graph. Therefore it is expected that on similar structures, the tested evolutionary algorithms will perform better if hybridized with the heuristics in MBH.

MBH is based on a Newton (or quasi-Newton) method for local minimization and on a restart of the search within a neighborhood N_ρ of a local minimum. We can view such local optimization methods as *dynamical systems*, where the evolution of the systems at each iteration is controlled by some *map*. Under suitable assumptions, the systems converge to a fixed point. For instance, if f is convex and C^2 is in a small enough domain D_k containing a local minimum which satisfies some regularity conditions, Newton's map converges quadratically to a single fixed point (the local minimum) in D_k.

Both DE and PSO can be rewritten in a compact form as a discrete dynamical system:

$$\begin{aligned} \mathbf{v}_{i,k+1} &= (1-c)\mathbf{v}_{i,k} + \mathbf{u}_{i,k} \\ \mathbf{x}_{i,k+1} &= \mathbf{x}_{i,k} + \nu S(\mathbf{x}_{i,k} + \mathbf{v}_{i,k+1})\mathbf{v}_{i,k+1} \end{aligned} \tag{20}$$

where the control $\mathbf{u}_{i,k}$ defines the next point that will be sampled for each one of the existing points in the solution space, the vectors $\mathbf{x}_{i,k}$ and $\mathbf{v}_{i,k}$ define the current state of a point in the solution space at stage k of the search process, and c is a viscosity, or dissipative coefficient, for the process. The value ν is given in Eq. (16).

In addition to Eq. (20), each optimization algorithm has heuristics responsible for selecting the new candidate points generated with $\mathbf{u}_{i,k}$. The binary selection operator is expressed through the function $S(\mathbf{x}_{i,k} + \mathbf{v}_{i,k+1})$ which can be either 1 if the candidate point is accepted or 0 if it is not accepted.

Differential Evolution, in its basic form, has $\mathbf{u}_{i,k}$ defined by Eq. (10), viscosity $c = 1$ and

$v_{max} = +\infty$. Therefore, we have the reduced map:

$$\mathbf{x}_{i,k+1} = \mathbf{x}_{i,k} + S(\mathbf{x}_{i,k} + \mathbf{u}_{i,k})\mathbf{u}_{i,k} \tag{21}$$

or in matrix form for the entire population, $\mathbf{x}_{k+1} = \mathbf{J}_k\mathbf{x}_k$.

Map (20) allows for a number of considerations on the evolution of the search process and therefore on the properties of the global optimization algorithm. In particular, the map can either: diverge to infinity, in this case the discrete dynamical system is unstable, the global optimization algorithm is not convergent; converge to a fixed point in D, in this case we can define a stopping criterion together with a restart procedure; converge to a limit cycle in which the same points in D are re-sampled periodically, even in this case we can define a stopping criterion and a restart procedure; or converge to a strange (chaotic) attractor, in this case a stopping criterion cannot be clearly defined because different points are sampled at different iterations.

Now, we can try to combine the properties of map (21) with MBH. In particular, the aim is to improve the performance of some of the tested evolutionary algorithms and at the same time to: drop the requirement for the continuity and differentiability of f required for MBH; automatically reduce the size of the region in which a candidate point is generated (the basic version of MBH has a constant size of N_ρ) and perform not only a local exploration of the neighborhood, but also a global one. In order to hybridize map (21) with MBH we need to demonstrate that it converges to a fixed point in D_k. We start by observing that if $S(\mathbf{x}_k + \mathbf{u}_k) = 1 \Leftrightarrow f(\mathbf{x}_k + \mathbf{u}_k) < f(\mathbf{x}_k)$, then the global minimizer $\mathbf{x}_g \in D$, if unique, is a fixed point for map (21) since every other point $\mathbf{x} \in D$ is such that $f(\mathbf{x}) > f(\mathbf{x}_g)$.

Then, let us assume that at every iteration k we can find two connected subsets of D, D_k and D_k^*, such that $f(\mathbf{x}_k) < f(\mathbf{x}_k^*), \forall \mathbf{x}_k \in D_k, \forall \mathbf{x}_k^* \in D_k^* \setminus D_k$, and let us also assume that $P_k \subseteq D_k$ while $P_{k+1} \subseteq D_k^*$ (recall that P_k and P_{k+1} denote the populations at iteration k and $k+1$ respectively). If $\mathbf{x}_l$ is the lowest local minimum in D_k, then $\mathbf{x}_l$ is a fixed point in D_k for (21). In fact, every point generated by (21) must be in D_k and $f(\mathbf{x}_l) < f(\mathbf{x}), \forall \mathbf{x} \in D_k$. Furthermore, if for every k, $P_k \subseteq D_k$ and $P_{k+1} \subseteq D_k^*$, then the reciprocal distance of the individuals cannot grow indefinitely because of the selection operator S, and the map cannot diverge.

Now, if we assume that the function f is strictly quasi-convex[20] in D_k (i.e. in a local region of D), we can prove that map (21) converges to a fixed point in D_k. It is worth underlining that, although the problem is not convex over the whole search space D, it can be assumed locally convex or quasi-convex under mild regularity assumptions which, according to our experiments, appear to be satisfied. The first step to proving the convergence of map (21) is to note that if f is continuous and strictly quasi-convex on a compact set D_k, the following

minimization problem with $F \in (0,1)$ has a strictly positive minimum value $\delta_r(\epsilon)$:

$$\begin{aligned} \delta_r(\epsilon) = \quad & \min \quad g(\mathbf{x}_1, \mathbf{x}_2) = f(\mathbf{x}_2) - f(F\mathbf{x}_1 + (1-F)\mathbf{x}_2) \\ & s.t. \quad \mathbf{x}_1, \mathbf{x}_2 \in D_k \\ & \quad\quad \|\mathbf{x}_1 - \mathbf{x}_2\| \geq \epsilon \\ & \quad\quad f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \end{aligned} \tag{22}$$

In fact, since f is strictly quasi-convex $g(\mathbf{x}_1,\ \mathbf{x}_2) > 0,\ \forall \mathbf{x}_1,\ \mathbf{x}_2 \in D_k$. Furthermore, the feasible region is compact and, therefore, according to Weierstrass' theorem, the function g attains its minimum value over the feasible region. If we denote the global minimum point of the problem by $(\mathbf{x}_1^*, \mathbf{x}_2^*)$, then we have

$$\delta_r(\epsilon) = g(\mathbf{x}_1^*, \mathbf{x}_2^*) > 0. \tag{23}$$

We can now say that, given a function f that is strictly quasi-convex over D_k and a population $P_k \in D_k$, then if $F \in (0,1)$ and $S(\mathbf{x}_k + \mathbf{u}_k) = 1 \Leftrightarrow f(\mathbf{x}_k + \mathbf{u}_k) < f(\mathbf{x}_k)$, the population P_k converges to a fixed point in D_k for $k \to \infty$.

We propose two distinct proofs for this statement. The first proof requires the additional assumption that the population always has an individual $\mathbf{x}_{j,k}$ that is strictly better than the others, i.e. $f(\mathbf{x}_{j,k}) < f(\mathbf{x}_{i,k})$ for any $i \neq j$. In this case at each iteration k, map (21) can generate, a displacement $(\mathbf{x}_{j,k} - \mathbf{x}_{i,k})$ for all members of the population with a strictly positive probability. This means that at each iteration we have a strictly positive probability that the whole population collapses into a single point. Then, for $k \to \infty$ the whole population collapses to a single point with a probability of one.

The second, more general, proof is the following. By contradiction, let us assume that we do not have convergence to a fixed point. Then, it must hold that:

$$\inf_k \max \left\{ \|\mathbf{x}_{i,k} - \mathbf{x}_{j,k}\|, i, j \in [1, ..., n_{pop}] \right\} \geq \epsilon > 0 \tag{24}$$

At every generation k the map can generate, with a strictly positive probability, a displacement $F(\mathbf{x}_{i^*,k} - \mathbf{x}_{j^*,k})$, where i^* and j^* identify the individuals with the maximal reciprocal distance, such that the candidate point is $\mathbf{x}_{cand} = F\mathbf{x}_{i^*,k} + (1-F)\mathbf{x}_{j^*,k}$ with $f(\mathbf{x}_{i^*,k}) \leq f(\mathbf{x}_{j^*,k})$. Since the function f is strictly quasi-convex, the candidate point is certainly better than $\mathbf{x}_{j^*,k}$ and, therefore, is accepted by S. Now, in view of Eq. (24) and the solution to problem (22) we must have that

$$f(\mathbf{x}_{cand}) \leq f(\mathbf{x}_{j,k}) - \delta_r(\epsilon). \tag{25}$$

Such a reduction will occur with a probability of one infinitely often, and consequently the

function value of at least one individual will be, with a probability of one, reduced by $\delta_r(\epsilon)$ infinitely often. In this way, however, the value of the objective function of such an individual would diverge to $-\infty$, which is a contradiction because f is bounded from below over the compact set D_k.

In particular, the above result shows that at each iteration, when the population of DE lies in the neighborhood of a local minimum satisfying some regularity assumption (e.g., the Hessian at the local minimum is definite positive, implying strict convexity in the neighborhood), then DE will converge to a fixed point. For general functions, we cannot always guarantee that the population will converge to a fixed point, but we can show that the maximum difference between objective function values in the population converges to 0, i.e., the points in the population tend to belong to the same level set. Given a function f and a population $P_k \in D_k$, then if $F \in (0, 1)$ and $S(\mathbf{x}_k + \mathbf{u}_k) = 1 \Leftrightarrow f(\mathbf{x}_k + \mathbf{u}_k) < f(\mathbf{x}_k)$, the following holds

$$\max_{i,j \in [1,\ldots,n_{pop}]} \left| f(\mathbf{x}_{j,k}) - f(\mathbf{x}_{i,k}) \right| \to 0 \tag{26}$$

as $k \to \infty$. Let S_k^* denote the set of best points in population P_k, i.e.

$$S_k^* = \{\mathbf{x}_{j,k} \ : \ f(\mathbf{x}_{j,k}) \leq f(\mathbf{x}_{i,k}) \ \ \forall \ i \in [1, \ldots, n_{pop}]\} \tag{27}$$

At each iteration k there is a strictly positive probability that the whole population will be reduced to S_k^* at the next iteration. In other words, there is a strictly positive probability for the event that the population, at a given iteration, will be composed of points with the same objective function value. Therefore, such event will occur infinitely often with a probability of one. Let us denote with $\{k_h\}_{h=1,\ldots}$ the infinite subsequence of iterations at which the event is true, and let

$$\Delta_h = f(\mathbf{x}_{i,k_h}) - f(\mathbf{x}_{i,k_{h+1}}) \tag{28}$$

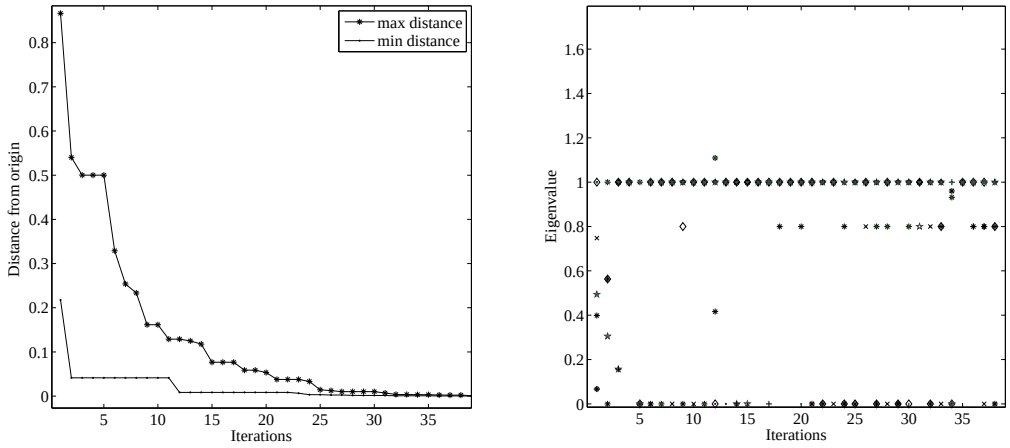
be the difference in objective function values at two consecutive iterations k_h and k_{h+1}. Note that, since at iterations k_h for $h = 1, \ldots$ the objective function values are equal, any i can be employed in the above definition.

It holds that for all $i, j \in [1, \ldots, n_{pop}]$

$$|f(\mathbf{x}_{j,k}) - f(\mathbf{x}_{i,k})| \leq \Delta_h \quad \forall \ k \in [k_h, k_{h+1}]. \tag{29}$$

Therefore, if we are able to prove that $\Delta_h \to 0$, as $h \to \infty$, then we can also prove that (26) is true. Let us assume, by contradiction, that $\Delta_h \not\to 0$. Then, there will exist a $\delta_r > 0$ such that $\Delta_h \geq \delta_r$ infinitely many times. But this would lead to function values diverging to $-\infty$ and, consequently, to a contradiction.

As a consequence of these results, a possible stopping criterion for DE would be to stop when the difference between the function values in the population drops below a given threshold. However, this could cause premature convergence. Indeed, even if at some iteration the value Δ_h drops to 0, this does not necessarily mean that the algorithm will be unable to make further progress. Therefore, since the most likely situation is convergence to a single point, we can use the fact that the maximum distance between points in the population drops below a threshold as the stopping criterion (using, as a safeguard, a maximum number of allocated generations as an alternative stopping criterion).



(a) (b)

Figure 6. Dissipative properties of Differential Evolution: a) max and min distance of the individuals in the population from the origin, b) eigenvalues with the number of evolutionary iterations.

To further verify the contraction property of the dynamics in Eq. (21) we can look at the eigenvalues of the matrix $\mathbf{J}_k$.

If the function f is strictly quasi-convex in D_k, the population converges to a fixed point in D_k, which implies that the map (21) is a contraction in D_k and therefore the eigenvalues[15] should have norms that, on average, are lower than 1.

This can be illustrated with the following test: Consider a population of 8 individuals and a D_k enclosing the minimum of a paraboloid with the minimum at the origin. We compute, for each step k, the distance of the closest and farthest individual from the local minimum and the eigenvalues of the matrix $\mathbf{J}$. Fig. 6 shows the behavior of the eigenvalues and the distance from the origin. From the figure we can see that for all iterations, the value of the norm of all the eigenvalues is in the interval $[0, 1]$, except for one eigenvalue at iteration 12. However, since on average the eigenvalues are lower than 1, the population contracts as represented in Fig. 6(a).

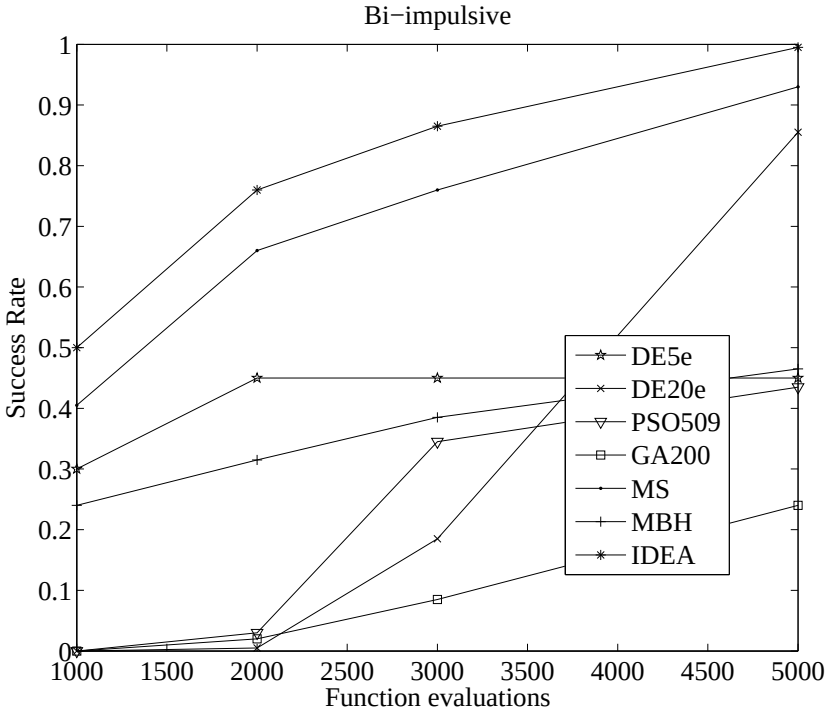


Figure 7. Successes rate of different optimizers on the bi-impulsive Earth-Apophis case.

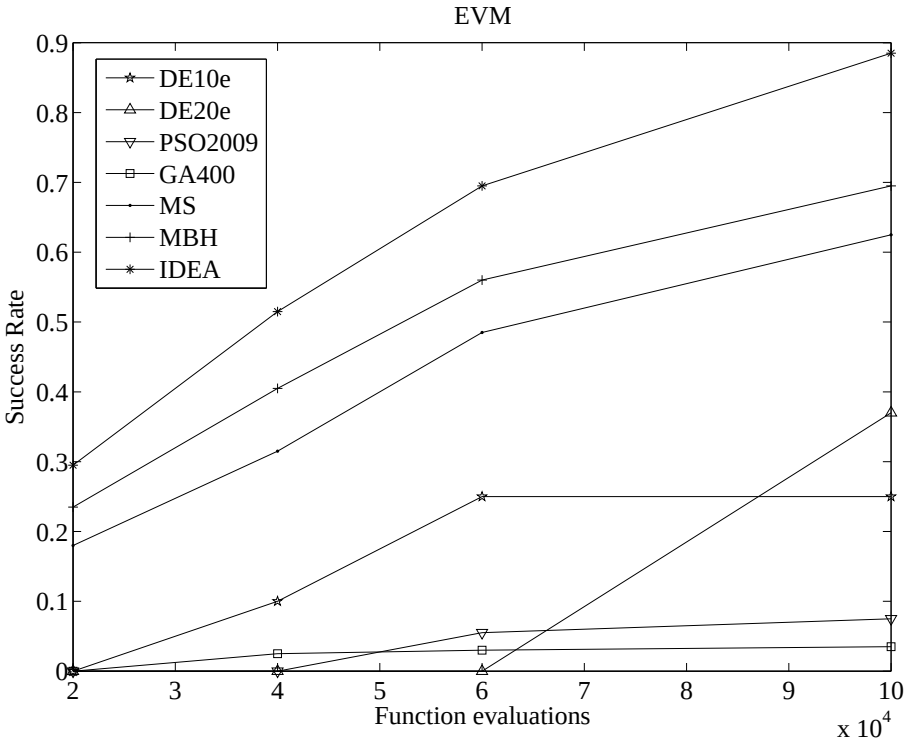


Figure 8. Successes rate of different optimizers on the EVM case.

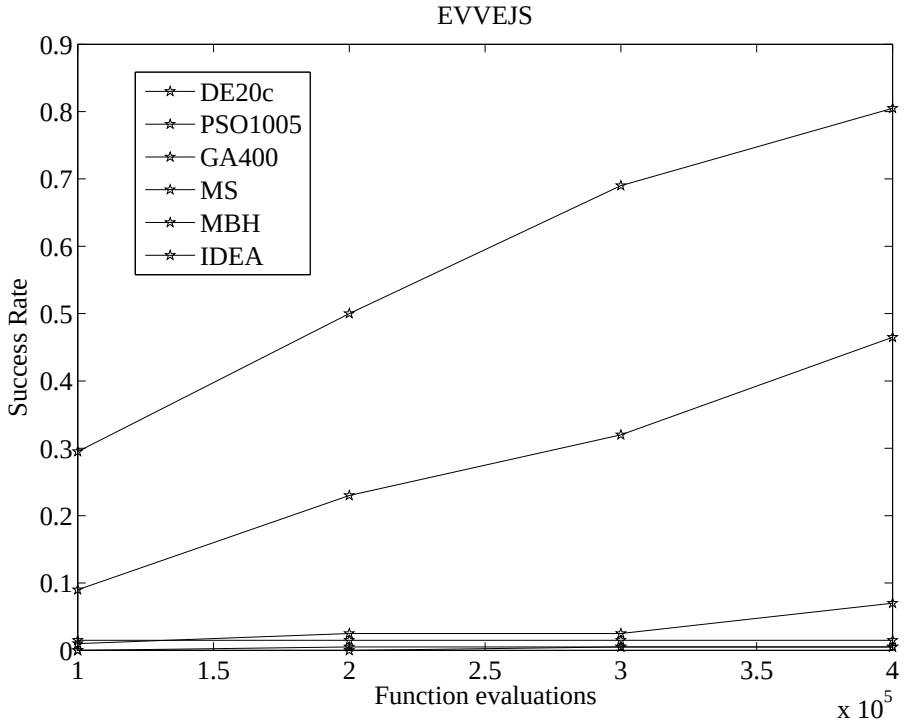


Figure 9. Successes rate of different optimizers on the EVVEJS case.

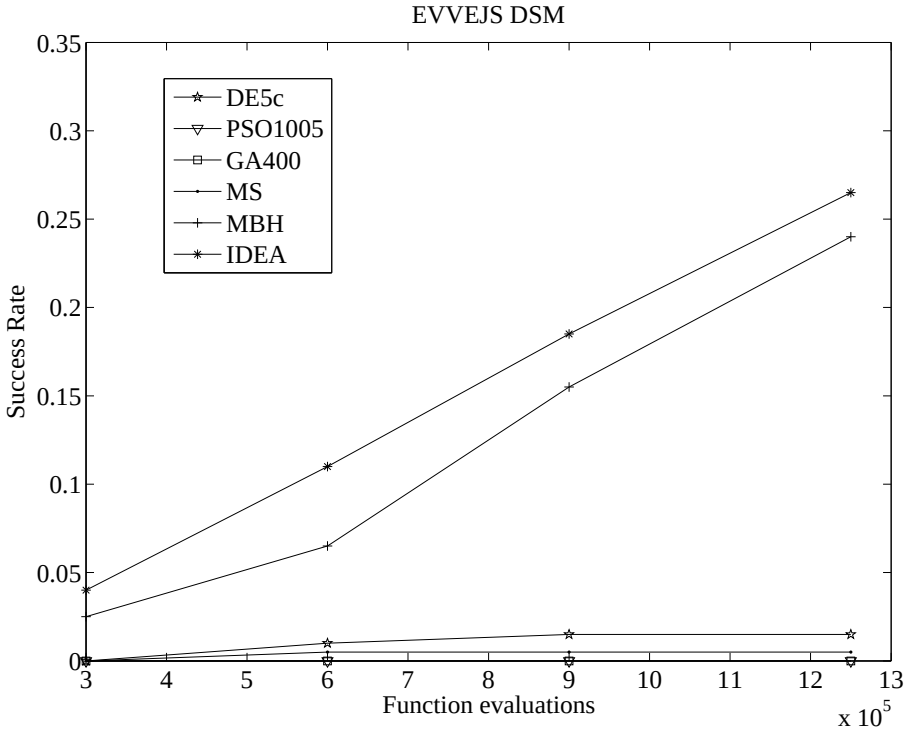


Figure 10. Successes rate of different optimizers on the EVVEJS with DSM's case.

Algorithm 3 Inflationary Differential Evolution Algorithm (IDEA)

1: Set values for n_{pop}, C_R and F, set $n_{feval} = 0$ and $k = 1$, set threshold tol_{conv}
2: Initialize $\mathbf{x}_{i,k}$ and $\mathbf{v}_{i,k}$ for all $i \in [1, ..., n_{pop}]$
3: Create the vector of random values $\mathbf{r} \in U[0, 1]$ and the mask $\mathbf{e} = \mathbf{r} < C_R$
4: **for all** $i \in [1, ..., n_{pop}]$ **do**
5: Select three individuals $\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \mathbf{x}_{i_3}$
6: Create the vector $\mathbf{u}_{i,k} = \mathbf{e}[(\mathbf{x}_{i_3,k} - \mathbf{x}_{i,k}) + F(\mathbf{x}_{i_2,k} - \mathbf{x}_{i_1,k})]$
7: $\mathbf{v}_{i,k+1} = (1 - c)\mathbf{v}_{i,k} + \mathbf{u}_{i,k}$
8: Compute S and ν
9: $\mathbf{x}_{i,k+1} = \mathbf{x}_{i,k} + S\nu\mathbf{v}_{i,k+1}$
10: $n_{feval} = n_{feval} + 1$
11: **end for**
12: $k = k + 1$
13: $\rho_A = \max(\|\mathbf{x}_{i,k} - \mathbf{x}_{j,k}\|)$, $\forall \mathbf{x}_{i,k}, \mathbf{x}_{j,k} \in P_{sub} \subseteq P_k$
14: **if** $\rho_A < tol_{conv}$ **then**
15: Run a local search from $\mathbf{x}_{best}$ till local minimum $\mathbf{x}_l$, where $\mathbf{x}_{best} = \arg\min_i f(\mathbf{x}_{i,k})$
16: Define a bubble D_l such that $\mathbf{x}_{i,k} \in D_l$, $\forall \mathbf{x}_{i,k} \in P_{sub}$ and $P_{sub} \subseteq P_k$
17: $A_g = A_g + \{\mathbf{x}_l\}$
18: Initialize $\mathbf{x}_{i,k}$ and $\mathbf{v}_{i,k}$ for all $i \in [1, ..., n_{pop}]$, in the bubble $D_l \subseteq D$
19: **end if**
20: **Termination** Unless $n_{feval} \geq n_{fevalmax}$, *goto* Step 3

If multiple minima are contained in D_k then it can be experimentally verified that the population contracts to a number of clusters initially converging to a number of local minima and eventually to the lowest among all of the identified local minima. Now, if a cluster contracts we can define a bubble $D_l \subseteq D$, containing the cluster, and re-initialize a subpopulation P_{sub} in D_l when the maximum distance $\rho_A = \max(\|\mathbf{x}_i - \mathbf{x}_j\|)$ among the elements in the cluster collapses below a tolerance value tol_{conv}. Every time a subpopulation is re-initialized, a local search is run from the best solution $\mathbf{x}_{best}$ of the cluster, with the resulting local minimum x_l saved in an archive A_g. This process leads to the modified DE in Algorithm 3. Note that the contraction of the population given, for example by the metric ρ_A, is a stopping criterion that does not depend explicitly on the value of the objective function, but on the contractive properties of the map (21).

This new algorithm was tested on the benchmark. For the tests presented in this paper we did not try to identify the formation of clusters made of subsets of the population, i.e. $P_{sub} = P_k$. Instead, the population was restarted inside a single bubble, when the maximum distance among its individuals was below tol_{conv}. The results are represented in Figs. 7 to 10. The new modified version of DE (IDEA) is compared against the most effective algorithms within each class (GA, DE, PSO, MS, MBH), on each one of the test cases. The success rate is represented for an increasing number of function evaluations and a constant number of repeated runs (here set to 200). With respect to the original version of DE, the improvement

in the success rate is impressively high for the EVVEJS. On the same case, the improvement is remarkable also with respect to MBH. In all other cases, the improvement is noticeable with respect to the standard DE and marginally significant in the case of the EVVEJS case with DSM's, with respect to MBH. On average, the new heuristic outperforms all other tested algorithms.

VI. Conclusion

In this paper we presented an analysis of some global optimization algorithms applied to the solution of a benchmark of space trajectory design problems. A rigorous procedure was defined to assess the performance of each of the global methods.

The test results and the analysis of the distribution of the minima revealed a possible underlying structure for the problems in the benchmark. Such a structure seems to favor the search heuristics implemented in MBH, compared to the other global methods. This observation led to the development of a new search algorithm based on a dynamical system interpretation of some evolutionary heuristics. This new algorithm combines the properties of Differential Evolution with the heuristics of MBH. On all the test cases in the benchmark, the new algorithm outperformed the other tested global methods and provided a remarkable improvement with respect to both DE and MBH.

All the problems in this paper are characterized by a limited number of funnel structures and a locally quasi-convex objective function. It is therefore expected that on similar problems, the new hybrid will perform equally well.

References

[1]P.J. Gage, R.D. Braun, I.M. Kroo, Interplanetary trajectory optimization using a genetic algorithm, Journal of the Astronautical Sciences, 43(1) (1995) 59–75.

[2]G. Rauwolf, V. Coverstone-Carroll, Near-optimal low-thrust orbit transfers generated by a genetic algorithm, Journal of Spacecraft and Rockets, 33(6) (1996) 859–862.

[3]Kim, Y.H., and Spencer, D.B., "Optimal Spacecraft Rendezvous Using Genetic Algorithms", Journal of Spacecraft and Rockets, Vol. 39, No. 6, Nov.–Dec. 2002, pp. 859–865.

[4]Abdelkhalik O., Mortari D., "N-Impulse Orbit Transfer Using Genetic Algorithms", Journal of Spacecraft and Rockets, Vol. 44, No. 2, March-April 2007, pp. 456–459.

[5]Olds A. D., Kluever C. A., Cupples M.L. "Interplanetary Mission Design Using Differential Evolution", Journal of Spacecraft and Rockets, Vol. 44, No. 5, Sept.–Oct. 2007, pp. 1060–1070.

[6]Di Lizia P., Radice G., "Advanced Global Optimization Tools for Mission Analysis and Design", Final Report of ESA Ariadna ITT AO4532,Call 03/4101, 2004.

[7]Myatt D. R., Becerra V.M. , Nasuto S.J. , and Bishop J.M., "Advanced Global Optimization Tools for Mission Analysis and Design", Final Rept. ESA Ariadna ITT AO4532/18138/04/NL/MV,Call03/4101, 2004.

[8]Vasile M., Summerer L., De Pascale P., "Design of Earth-Mars Transfer Trajectories using Evolutionary Branching Techniques", Acta Astronautica, Vol. 56, pp. 705–720, 2005.

[9]Vasile M., De Pascale P., Preliminary Design of Multiple Gravity-Assist Trajectories, Journal of Spacecraft and Rockets, Vol. 43,No.4, July-August, 2006.

[10]C. J. Adcock. Sample size determination: a review. *The Statistician*, 46(2):261–283, 1997.

[11]H. Battin. *An Introduction to the Mathematics and Methods of Astrodynamics*. AIAA, 1999.

[12]M. Clerc. *Particle Swarm Optimization*. ISTE, 2006.

[13]K.V. Price, R.M. Storn and J.A. Lampinen. *Differential Evolution. A Practical Approach to Global Optimization*. Natural Computing Series, Springer, 2005.

[14]M. Mitchell. *An introduction to genetic algorithms*. MIT Press, 1998.

[15]Galor O. Introduction to Stability Analysis of Discrete Dynamical Systems. Macroeconomics, EconPapers, 2004.

[16]Wales D. J., and Doye J. P. K., Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms, *J. Phys. Chem. A*, 101, 5111–5116, (1997).

[17]Leary R. H., Global optimization on funneling landscapes, *J. Global Optim.*, 18, 367–383, (2000).

[18]B. Addis, M. Locatelli, F. Schoen, Local optima smoothing for global optimization, Optimization Methods and Software, 20, 417–437 (2005)

[19]M.Locatelli, "On the multilevel structure of global optimization problems", Computational Optimization and Applications, 30, 5–22 (2005)

[20]Rockafellar, R. T. *Convex analysis*. 1970, Princeton University Press.

[21]Reeves, C.R., Yamada, T., Genetic Algorithms, Path Re-linking and the Flowshop Sequencing Problem, Evolutionary Computation, Vol. 6, pp. 45–60, 1998.

[22]Rudolph G., Convergence of Evolutionary Algorithms in General Search Space. Proceedings of the IEEE International Conference on Evolutionary Computation 1996, Nagoya Japan, 20-22 May 1996, ISBN: 0-7803-2902-3.

[23]Pinter J., Convergence properties of stochastic optimization procedures. Optimization, 15:3, 405–427, DOI: 10.1080/02331938408842957.

[24]Chipperfield A., Fleming P., Pohlheim H., Fonseca C. Genetic Algorithm Toolbox. For Use with Matlab. User's Guide, version 1.2, University of Sheffield.