# On Testing Global Optimization Algorithms for Space Trajectory Design

M. Vasile* and E. Minisci †

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

M. Locatelli‡

*Universitá degli Studi di Torino, Turin, 10149, Italy*

**In this paper we discuss the procedures to test a global search algorithm applied to a space trajectory design problem. Then, we present some performance indexes that can be used to evaluate the effectiveness of global optimization algorithms. The performance indexes are then compared highlighting the actual significance of each one of them. A number of global optimization algorithms are tested on four typical space trajectory design problems. From the results of the proposed testing procedure we infer for each pair algorithm-problem the relation between the heuristics implemented in the solution algorithm and the main characteristics of the problem under investigation. From this analysis we derive a novel interpretation of some evolutionary heuristics, based on dynamical system theory and we significantly improve the performance of one of the tested algorithms.**

## 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 (such as 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 based techniques are used. In fact, if, on one hand, a stochastic search provides a non-zero probability to find an optimal solution even with a small number of function evaluations, 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 studies by the University of Reading[7] and by the University of Glasgow.[6] The former presented a small set of test problems mainly focusing on multiple gravity assist trajectories, while the latter 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, belonging to a subclass of Evolutionary Algorithms, performed particularly well on most of the problems, compared to other methods. In both studies, the indexes of performance for stochastic methods were: the average value of the best solution found for each run over a number of independent runs, the corresponding variance and the best value from all the runs. For deterministic methods, the index of performance was the best value for a given number of function evaluations. It should be noted that the application of global methods to space trajectory problems has often considered the problem as a black-box with limited exploitation of problem characteristics. On the other hand, ad hoc techniques exploiting problem characteristics[7] provide a sensible improvement over the simple enumerative search. In this paper, we propose a testing methodology for global optimization methods addressing specifically black-box problems in space trajectory design. In particular, we focus our attention on stochastic based approaches. The paper discusses the actual significance of some performance indexes and proposes some criteria to evaluate the actual usefulness of an algorithm. Furthermore, the paper presents a benchmark of

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

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

‡Associate Professor, Dipartimento di Informatica, Corso Svizzera, 189.

test cases, and an analysis of the relationship between the heuristics implemented in some global optimization algorithms and the main characteristics of the test cases under investigation. The identification of common patterns in the relation between solution method and problem features represents a useful guideline for the selection of the most appropriate approach to a problem. From this analysis we derive a novel formulation of some evolutionary heuristics, based on dynamical system theory and we propose a modified version of Differential Evolution that improves significantly over all the standard DE. It is worth to underline that this paper does not intend to propose any particular benchmark for testing global optimization algorithms, nor it wants to prove that one approach is better than the others. The goal of this paper is instead to propose a method of assessment and analysis of stochastic methods for global optimization applied to space trajectory design problems. A correct analysis of the performance of a particular method applied to a specific class of problem can be useful to identify which heuristic is most effective.

## II. Problem Description

We consider a benchmark made of four different test-cases, with increasing complexity: a direct bi-impulsive transfer from the Earth to an asteroid, a transfer to Mars via a gravity assist of Venus, a multi-gravity assist transfer to Saturn with no mid-course manoeuvres and the same transfer but with mid-course manoeuvres. In all of these cases the objective will be to minimize the total variation of the velocity of the spacecraft due to all propelled maneuvers, or total $\Delta v$.

### A. Bi-impulsive Earth-Apophis Transfer

A simple, but already significant, application 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. The transfer is computed as the solution of a Lambert's problem.[11] The objective function for this problem is the sum of the departure velocity change $\Delta v_0$ and the arrival velocity change $\Delta v_f$:

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

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 of the solution vector. In particular, the launch date from the Earth was taken in the interval [3653, 10958]MJD2000 (i.e. number of elapsed days since January 1st 2000), while the time of flight was taken in the interval [50, 900] days. A representation of the search space can be seen in Fig.1 where the value of the objective function was plotted with level curves against $t_0$ and $T$: dark regions represent local minima.
The known best solution in $D$ is $f_{best}$=4.3745658 km/s, $x_{best}$=[10027.6215924826, 305.12163547522].

### B. Earth-Venus-Mars Transfer with DSM

The second test-case consists of a transfer from Earth to Mars with the use of 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).

#### *1. Trajectory Model*

A general MGA-DSM trajectory can be modeled through a sequence of $N_P - 1$ legs connecting $N_P$ celestial bodies (Fig. 2).[9] In particular if all celestial bodies are planets, each leg begins and ends with an encounter with a planet. Each leg $i$ is made of two conic arcs: the first, propagated analytically forward in time, ends where the second, solution of a Lambert's problem, begins. The two arcs have a discontinuity in the absolute heliocentric velocity at their matching point $M$. Each DSM is computed as the vector difference between the velocities along the two conic arcs at the matching point. Given the transfer time $T_i$ and the variable $\alpha_i \in [0, 1]$ relative to each leg $i$, the matching point is at time $t_{DSM,i} = t_{f,i-1} + \alpha_i T_i$, where $t_{f,i-1}$ is the final time of the leg $i-1$. The relative velocity vector $\mathbf{v}_0$ at the departure planet can be a design parameter and is expressed as:

$$\mathbf{v}_0 = v_0[\sin\delta\cos\theta, \sin\delta\sin\theta, \cos\delta]^T \tag{3}$$

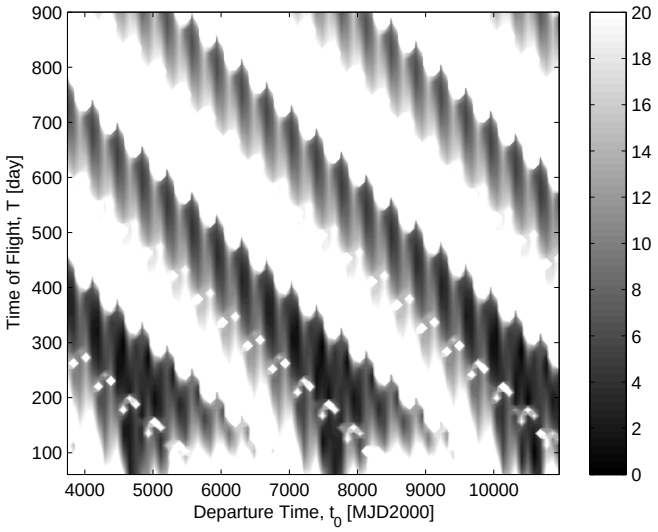



**Figure 1. Earth-Apophis search space.**

with the angles $\delta$ and $\theta$ respectively representing the declination and the right ascension with respect to a local reference frame with 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. This choice allows easily constraining the escape velocity and asymptote direction while adding the possibility of having a deep space maneuver in the first arc after the launch. This is often the case when escape velocity must be fixed due to the launcher capability or to the requirement of a resonant swing-by of the Earth (Earth-Earth transfers). In order to have a uniform distribution of random points on the surface of the sphere defining all the possible launch directions, the following transformation has been applied:

$$\bar{\theta} = \frac{\theta}{2\pi}; \bar{\delta} = \frac{\cos(\delta + \pi/2) + 1}{2} \tag{4}$$

It results that the sphere surface is uniformly sampled when a uniform distribution of points for $\bar{\theta}, \bar{\delta} \in [0, 1]$ is chosen. Once the heliocentric velocity at the beginning of leg $i$, which can be the result of a swing-by maneuver or the asymptotic velocity after launch, is computed, the trajectory is analytically propagated until time $t_{DSM,i}$. The second arc of leg $i$ is then solved through a Lambert's algorithm, from $M_i$, the Cartesian position of the deep space maneuver, to $P_i$, the position of the target planet of phase $i$, for a time of flight $(1 - \alpha_i)T_i$. Two subsequent legs are then joined together with a gravity assist manoeuvre. The effect of the gravity of a planet is to instantaneously change the velocity vector of the spacecraft. The relative incoming velocity vector and the outgoing velocity vector, at the planet swing-by, have the same modulus but different directions; therefore the heliocentric outgoing velocity results to be different from the heliocentric incoming one. In the linked conic model the spacecraft is assumed to follow a hyperbolic trajectory with respect to the swing-by planet. The angular difference between the incoming relative velocity $\tilde{\mathbf{v}}_i$ and the outgoing one $\tilde{\mathbf{v}}_o$ depends on the modulus of the incoming velocity and on the pericenter radius $r_i$. Both the relative incoming and outgoing velocities belong to the plane of the hyperbola. However, in the linked-conic approximation, the maneuver is assumed to occur at the planet, where the planet is a point mass coinciding with its center of mass. Therefore, given the incoming velocity vector, one angle is required to define the attitude of the plane of the hyperbola $\Pi$. There are different possible choices for the attitude angle $\gamma$; the one proposed in Ref. 7 has been adopted (Fig. 3): $\gamma$ is the angle between the vector $\mathbf{n}_\Pi$, normal to the hyperbola plane $\Pi$, and the reference vector $\mathbf{n}_r$, that is normal to the plane containing the incoming relative velocity and the velocity of the planet $\mathbf{v}_P$.
Given the number of legs of the trajectory $N_L = N_P - 1$, the complete solution vector for this model is:

$$\mathbf{x} = [v_0, \bar{\theta}, \bar{\delta}, t_0, \alpha_1, T_1, \gamma_1, r_{p,1}, \alpha_2, T_2, ..., \gamma_i, r_{p,i}, T_{i-1}, \alpha_{i-1}, ..., \gamma_{N_L-1}, r_{p,N_L-1}, \alpha_{N_L}, T_{N_L}] \tag{5}$$

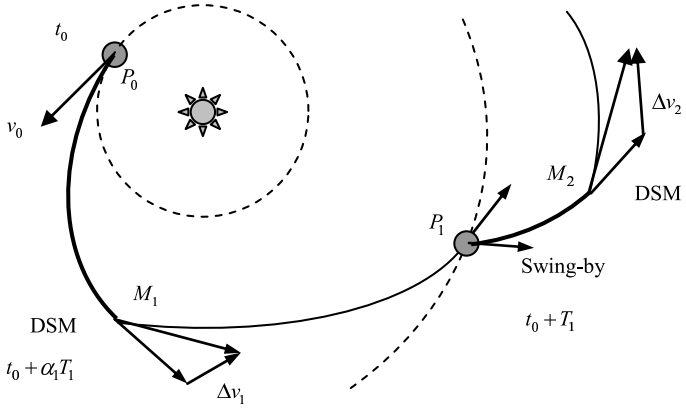



**Figure 2. Schematic representation of a multiple gravity assist trajectory**

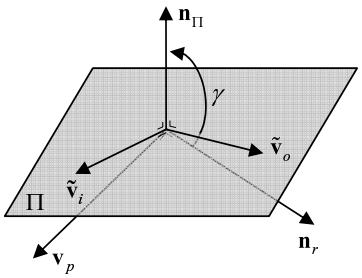



**Figure 3. Schematic representation of a multiple gravity assist trajectory**

where $t_0$ is the departure date. Now, the design of a multi-gravity assist transfer can be transcribed into a general nonlinear programming problem, with simple box constraints, of the form:

$$\min_{\mathbf{x}\in D} f(\mathbf{x}) \tag{6}$$

One of the appealing aspects of this formulation is its solvability through a general global search method for box constrained problems. Depending on the kind of problem under study, the objective function can be defined in different ways. Here we choose to focus on the problem of minimizing the total $\Delta v$ of the mission, therefore the objective function $f(\mathbf{x})$ is:

$$f(\mathbf{x}) = v_0 + \sum_{i=1}^{N_p} \Delta v_i + \Delta v_f \tag{7}$$

where $\Delta v_i$ is the velocity change due to the DSM in the $i-th$ leg, and $\Delta v_f$ is the maneuver needed to inject the spacecraft into the final orbit.
For a transfer to Mars via Venus, the solution vector in Eq. (5) has six dimensions. In particular the initial velocity 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:

$$\mathbf{x} = [t_0, T_1, \gamma_1, r_{p,1}, \alpha_2, T_2] \tag{8}$$

Since the initial velocity is not a free parameter $v_0$ is the modulus of the vector difference between the velocity of the Earth at time $t_0$ and the velocity of the spacecraft at the same time. Note that the final $\Delta v_f$

the $\Delta v$ needed to inject the spacecraft into an ideal operative orbit around Mars with 3950 km of pericenter radius and 0.98 of eccentricity. 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, $x_{best}$=[4472.01334656364, 172.289324250300, 2.97843388136061, 1, 0.509432880679500, 697.610012389372].

### 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 $\Delta v$ maneuver at the pericenter of the gravity assist hyperbola for each planet. No deep-space maneuvers are possible and each planet-to-planet transfer is computed as the solution of a Lambert's problem. Therefore, the whole trajectory is completely defined by the departure time $t_0$ and by the transfer time for each leg $T_i$, with $i = 1, ..., N_P - 1$. The 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. The trajectory model for this problem can be downloaded from the ESA/ACT website [a]. In order to take into account the constraints on the altitude of the pericenters the objective function is augmented with the weighted violation of the constraints:

$$f(\mathbf{x}) = \Delta v_0 + \sum_{i=1}^{N_p-2} \Delta v_i + \Delta v_f + \sum_{i=1}^{N_p-2} w_i (r_{p,i} - r_{pmin,i})^2 \tag{9}$$

with the solution vector:

$$\mathbf{x} = [t_0, T_1, T_2, T_3, T_4, T_5] \tag{10}$$

The weighting functions $w_i$ are defined as follows:

$$\begin{aligned} w_i &= 0.005[1 - sign(r_{p,i} - r_{pmin,i})], i = 1, ..., 3 \\ w_4 &= 0.0005[1 - sign(r_{p,4} - r_{pmin,4})] \end{aligned} \tag{11}$$

with the minimum pericenter radii $r_{pmin,1} = 6351.8$, $r_{pmin,2} = 6351.8$, $r_{pmin,3} = 6778.1$ and $r_{pmin,4} = 671492$. For this case the dimensionality of the problem is six, and the search space $D$ is 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, $x_{best}$=[-789.8055, 158.33942, 449.38588, 54.720136, 1024.6563, 4552.7531].

### D. Earth-Saturn Transfer with DSMs

The forth test case is again a multi gravity assist trajectory from the Earth to Saturn following the sequence Earth-Venus-Venus-Earth-Jupiter-Saturn (EVVEJS) but a deep space manoeuvre is allowed along the transfer arc from one planet to the other. Although from a trajectory design point of view this problem is similar to problem three, the model is substantially different and therefore it 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 for this test case can be downloaded from the ESA/ACT web site [b]. This model is very similar to the one for problem two, the only differences are in the definition of the attitude angle $\gamma$ of the plane of the hyperbola, which is at 90 degrees with respect to the one of problem two, and in the computation of $\Delta v_f$ that is now the modulus of the vector difference between the velocity of Saturn at arrival and the velocity of the spacecraft at the same time. Although the difference is minimal we preferred to use the ESA/ACT version since for this problem some reference solutions are already available and

[a] *http://www.esa.int/gsp/ACT/inf/op/globopt/evvejs.htm*

[b] *http://www.esa.int/gsp/ACT/inf/op/globopt/edvdvdedjds.htm*

therefore a comparison is easier and not affected by any difference in the implementation of the trajectory model. The best known solution is $f_{best}$=8.4091810440 km/s, $x_{best}$=[-779.060197373242, 3.32046443745595, 0.531333503613675, 0.376218447342955, 168.685775870437, 422.672656805198, 53.3360098337041, 589.777827855018, 2200, 0.718720247401635, 0.532962541494841,
0.159170896444411,0.470495109020601, 0.0986526263521857, 1.46946051297954, 1.05138706406598,
1.30594027188689, 69.8194077461197, -1.60160853231321, -1.9600386515463,
-1.55445003054861, -1.51343200828766].

Note that, prior to run each test we normalized the search space for each one 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 Multistart (MS) and Monotonic Basin Hopping (MBH) that are based on multiple local searches with a gradient method. In a previous paper by the authors[27] we showed how deterministic algorithms, such as DIRECT,[18] might work better then stochastic ones on simple problems, such as the bi-impulsive case. On the other hand, for more complex problems, stochastic algorithms provide a better solution with a lower number of function evaluations.
In general, given a solution vector $\mathbf{x}_i$ in the solution space $D$, the heuristics implemented in each one of the global search methods listed above, aim at performing the following three tasks:

- at iteration $k$ take samples in the solution space by generating a variation $\mathbf{v}_{i,k+1}$ of $\mathbf{x}_{i,k}$:

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

- select a subset of all the samples
- decide when to stop the search

Regarding the stopping rule, in order to make a fair comparison, we will employ the same for all the tested algorithms, namely we stop the search when the total number of function evaluations $n_{feval}$ performed by the algorithm exceeds a predefined value $n_{fevalmax}$. In the following we will give a brief algorithmic description of all the algorithms.

### *1. Genetic Algorithms*

Genetic Algorithms (GAs)[15] are stochastic search methods that take their inspiration from natural selection and survival of the fittest in the biological world. Each iteration of an GA involves a competitive selection that eliminates poor solutions. The solutions with high fitness are recombined with other solutions by swapping parts of a solution with 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 space for which good solutions have already been seen. The GA search process is summarized in Algorithm 1.
Regarding the GA application, only the influence of the population size was considered ([100, 200, 400] for the bi-impulse test case and [200, 400, 600] for the other three cases, with single values for crossover and mutation probability, $C_r = 0.5$ and $M_p = 1/d$ respectively, where $d$ denotes the dimension of the problem. $C_r = 0.5$ is the probability of transferring one component of a parent solution vector to the child solution vector. The adopted code uses a single value for generation gap, $GGAP = 1$, thus $n_{pop} \times GGAP$ new individuals are produced at each generation, and a single value for the insertion rate, $INSR = 0.5$, which decides how many of the offsprings are inserted in the new population, as well. Therefore, in the following GAs are identified by the population size, for example GA100 stands for Genetic Algorithms with 100 individuals.

### *2. Differential Evolution*

Differential Evolution (DE)[13] is a method of mathematical optimization of multidimensional multimodal (i.e. exhibiting more than one minimum) functions and belongs to the class of Evolution Strategy optimizers.

**Algorithm 1** GA

1: Set values for $n_{pop}$, $GGAP$, $C_r$, $M_p$ and $INSR$, set $n_{feval} = 0$ and $k = 1$
2: Initialize a population $P_k$ of individuals $\mathbf{x}_{i,k}$ for all $i \in [1, ..., n_{pop}]$
3: Rank $P_k$ according to objective function value
4: Select individuals from $P_k$
5: Recombine selected individuals
6: Mutate offspring with probability $M_p$
7: Calculate objective function for offsprings, update $n_{feval}$
8: Insert $INSR$ best offspring to replace worst parents
9: $k = k + 1$
10: **Termination** Unless $n_{feval} \geq n_{fevalmax}$, *goto* Step 3

The main idea is to generate the variation vector $\mathbf{v}_{i,k+1}$ by taking the weighted difference between two other solution vectors randomly chosen within a population of solution vectors and to add that difference to the vector difference between $\mathbf{x}_{i,k}$ and a third solution vector:

$$\mathbf{v}_{i,k+1} = \mathbf{e}[(\mathbf{x}_{i_3,k} - \mathbf{x}_{i,k}) + F(\mathbf{x}_{i_2,k} - \mathbf{x}_{i_1,k})] \tag{13}$$

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

$$e(j) = \begin{cases} 1 \Rightarrow r \leq C_R \\ 0 \Rightarrow r > C_R \end{cases} \tag{14}$$

with $j = 1, ..., n$, $r$ is taken from a random uniform distribution $r \in U[0, 1]$ and $C_R$ is 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 or a random one changes significantly the convergence speed of the algorithm. The selection process is generally deterministic and simply preserves the variation of $\mathbf{x}_{i,k}$ only if $f(\mathbf{x}_{i,k} + \mathbf{v}_{i,k+1}) < f(\mathbf{x}_{i,k})$. It is worth noting that the only stochastic components in the DE process sits in the choice of the indexes $i_1, i_2$ and $i_3$. Since the selection is deterministic, the process tend to preserve only the elements of $\mathbf{v}_{i,k+1}$ that yield to an improvement of the population. Therefore, the whole population evolves toward a similar behavior for all the solution vectors, i.e. vectors with similar elements. The DE search process is summarized in Algorithm 2.

**Algorithm 2** Differential Evolution

1: Set values for $n_{pop}$, $C_R$ and $F$
2: Set $n_{feval} = 0$ and $k = 1$
3: Initialize $\mathbf{x}_{i,k}$ and $\mathbf{u}_{i,k}$ for all $i \in [1, ..., n_{pop}]$
4: Create the vector of random values $\mathbf{r} \in U[0, 1]$ and the mask $\mathbf{e} = \mathbf{r} < C_R$
5: **for all** $i \in [1, ..., n_{pop}]$ **do**
6: Select three individuals $\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \mathbf{x}_{i_3}$
7: Create the vector $\mathbf{v}_{i,k+1} = \mathbf{e}[(\mathbf{x}_{i_3,k} - \mathbf{x}_{i,k}) + F(\mathbf{x}_{i_2,k} - \mathbf{x}_{i_1,k})]$
8: If $f(\mathbf{x}_{i,k} + \mathbf{v}_{i,k+1}) < f(\mathbf{x}_{i,k}) \Rightarrow \mathbf{x}_{i,k+1} = \mathbf{x}_{i,k} + \mathbf{v}_{i,k+1}$
9: If $f(\mathbf{x}_{i,k} + \mathbf{v}_{i,k+1}) \geq f(\mathbf{x}_{i,k}) \Rightarrow \mathbf{x}_{i,k+1} = \mathbf{x}_{i,k}$
10: $n_{feval} = n_{feval} + 1$
11: **end for**
12: $k = k + 1$
13: **Termination** Unless $n_{feval} \geq n_{fevalmax}$, *goto* Step 4

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, two strategies, convergence and explore, and single values of step-size and crossover probability $F = 0.75$ and $C_R = 0.8$ respectively, on the basis of common use. In the following the six settings will be denominated with DE5c,DE10c,DE20c for the ones using the convergence strategy and with DE5e,DE10e,DE20e for the ones using the exploration strategy.

*3. Particle Swarm Optimization*

Particle swarm optimization (PSO)[12] is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995, inspired by 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, at each iteration, changing the velocity of each particle $i$ according to a close-loop control mechanism:

$$\mathbf{v}_{i,k+1} = w\mathbf{v}_{i,k} + \mathbf{u}_{i,k} \tag{15}$$

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

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

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

$$\mathbf{u}_{i,k} = c_1 r_1(\mathbf{x}_{gi,k} - \mathbf{x}_{i,k}) + c_2 r_2(\mathbf{x}_{go,k} - \mathbf{x}_{i,k}) \tag{16}$$

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

$$\mathbf{x}_{i,k+1} = \mathbf{x}_{i,k} + \nu\mathbf{v}_{i,k+1} \tag{17}$$

with

$$\nu = \min([v_{max}, v_{i,k+1}])/v_{i,k+1} \tag{18}$$

The search is continued till the decision to stop is taken. 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 tend 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. The basic scheme of PSO is summarized in Algorithm 3.

**Algorithm 3** PSO

1: Set values for $c_1, c_2$, $n_{pop}$, $n_{fevalmax}$, set $k = 1$, compute $w$
2: Initialize $\mathbf{x}_{i,k}$ and $\mathbf{v}_{i,k}$ for all $i \in [1, ..., n_{pop}]$
3: $\mathbf{x}_{go,k} = \arg\min_{\mathbf{x}_{i,k}} f(\mathbf{x}_{i,k}), i = 1, ..., n_{pop}$, $n_{feval} = n_{pop}$
4: **for all** $i \in [1, ..., n_{pop}]$ **do**
5: Create random values $r_1, r_2 \in U[0\ 1]$
6: Update particle velocity $\mathbf{v}_{i,k+1} = w\mathbf{v}_{i,k} + c_1 r_1(\mathbf{x}_{gi,k} - \mathbf{x}_{i,k}) + c_2 r_2(\mathbf{x}_{go,k} - \mathbf{x}_{i,k})$
7: Check constraint on max velocity and compute $\nu$
8: Update particle position $\mathbf{x}_{i,k+1} = \mathbf{x}_{i,k} + \nu\mathbf{v}_{i,k+1}$
9: Update local best $f(\mathbf{x}_{i,k+1}) < f(\mathbf{x}_{gi,k}) \Rightarrow \mathbf{x}_{gi,k+1} = \mathbf{x}_{i,k+1}$
10: Update global best $f(\mathbf{x}_{i,k+1}) < f(\mathbf{x}_{go,k}) \Rightarrow \mathbf{x}_{go,k+1} = \mathbf{x}_{i,k+1}$
11: $n_{feval} = n_{feval} + 1$
12: **end for**
13: $k = k + 1$, update $w$
14: **Termination** Unless $n_{feval} \geq n_{fevalmax}$, *goto* Step 4

For the PSO algorithm, nine different settings were considered, resulting from the combination of three sets of population, again $[5\,d, 10\,d, 20\,d]$, 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 denominated with PSO5,PSO10,PSO20, then we add two digits 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 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 with a Latin Hypercube distribution (we call this algorithm MS). The basic scheme for MS is described in Algorithm 4.

**Algorithm 4** MS

1: Set $k = 1$, $f_{best} = +\infty$
2: Select point $\mathbf{y}_k$ according to a Latin Hypecube distribution
3: Run a local optimizer $a_l$ from $\mathbf{y}_k$ and let $\mathbf{x}_k$ be the detected local minimum.
4: Evaluate the objective function $f(\mathbf{x}_k)$
5: $f(\mathbf{x}_k) < f_{best} \Rightarrow \mathbf{x}_{best} = \mathbf{x}_k$, $f_{best} = f(\mathbf{x}_k)$
6: Run a local optimizer $a_l$ from each $f(\mathbf{x}_i)$
7: Set $n_{eval} = n_{eval} + eval_k$, where $eval_k$ denotes the number of function evaluations required by the $k$-th local search.
8: **Termination** Unless $n_{feval} \geq n_{fevalmax}$, *goto* Step 2

*5. Monotonic Basin Hopping*

Monotonic Basin Hopping (MBH) was first applied to special global optimization problems, the molecular conformation ones,[19,20] and later extended to general global optimization problems.[21,22] In its basic version it is quite similar to MS. It is also based on multiple local searches and the only difference is represented by 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 local minimizer $\mathbf{x}$. The parameter $\rho$ controls the size of the neighborhood. Its choice is essential for the performance. Too low a value would cause to generate points only within the basin of attraction of the current local minimizer; too large a value would basicall cause MBH to behave like MS. A careful choice of $\rho$ may lead to results which strongly outperform those of MS, in spite of the apparently mild difference between the two algorithms. In this work $N_\rho(\mathbf{x})$ will be a hypercube with edge length $2\rho$ centered at $\mathbf{x}$. The effectiveness fo the MBH can be improved with a global resampling. When the value of the global solution does not changes consecutively for $iun_{max}$ iterations, the search restarts from a point sampled in the whole search space.

**Algorithm 5** MBH

1: Select a point $\mathbf{y}$ in the solution space $D$; initialize $iun = 0$
2: Run a local optimizer $a_l$ from $\mathbf{y}$ and let $\mathbf{x}$ be the detected local minimum.
3: Set $n_{eval} = n_{eval} + eval$, where $eval$ denotes the number of function evaluations required by the local search.
4: Evaluate the objective function $f(\mathbf{x})$
5: Select a candidate point $\mathbf{x}_c \in N_\rho(\mathbf{x})$
6: Run a local optimizer $a_l$ from $\mathbf{x}_c$ and let $\mathbf{x}_l$ be the local minimum in $N_\rho(\mathbf{x})$ found by $a_l$
7: Set $n_{eval} = n_{eval} + eval$, where $eval$ denotes the number of function evaluations required by the local search.
8: **if** $f(\mathbf{x}_l) \leq f(\mathbf{x})$ **then**
9: $\quad \mathbf{x} \leftarrow \mathbf{x}_l$
10: $\quad iun = 0$
11: **else**
12: $\quad iun = iun + 1$
13: **end if**
14: **if** $iun \geq iun_{max}$ **then**
15: $\quad$ *goto* Step 1
16: **end if**
17: **Termination** Unless $n_{feval} \geq n_{fevalmax}$, *goto* Step 5

## IV. Testing Procedure

In this section we describe testing procedures that can be used to investigate the complexity of the problem and to derive performance indexes to compare different algorithms. If we call $A$ a generic solution algorithm and $p$ a generic problem, we can define the procedure in Algorithm 6. Now and in the following we say that

**Algorithm 6** Convergence Test

1: set the max number of function evaluations for $A$ equal to $N$
2: apply $A$ to $p$ for $n$ times
3: **for all** $i \in [1, ..., n]$ **do** $\phi(N, i) = \min f(A(N), p, i)$
4: **end for**
5: compute: $\phi_{min}(N) = \min_{i\in[1,...,n]} \phi(N, i)$, $\phi_{max}(N) = \max_{i\in[1,...,n]} \phi(N, i)$

an algorithm $A$ is globally convergent, when for a number of function evaluations $N$ that goes to infinity the two functions $\phi_{min}$ and $\phi_{max}$ converge to the same value, which is the global minimum value denoted as $f_{global}$. An algorithm $A$ is simply convergent, instead, if for $N$ that goes to infinity the two functions $\phi_{min}$ and $\phi_{max}$ converge to the same value, which is not necessarily a global or a local minimum for $f$.
If we fix a tolerance value $tol_f$, we could consider the following random variable as a possible quality measure of a globally convergent algorithm

$$N^* = \min\{N : \ \phi_{max}(N') - f_{global} \leq tol_f \ : \ \forall \ N' \geq N\}.$$

The larger (the expected value of) $N^*$ is, the slower is the global convergence of $A$. Figures 4a-b show the convergence profile for the bi-impulsive problem, 50 repeated independent runs Latin hypercube sampling, local optimization from each sample. Slightly more than 1000 initial samples are required to have a 100% convergence to the global minimum. However, the procedure in Algorithm 6 can be unpractical since, though

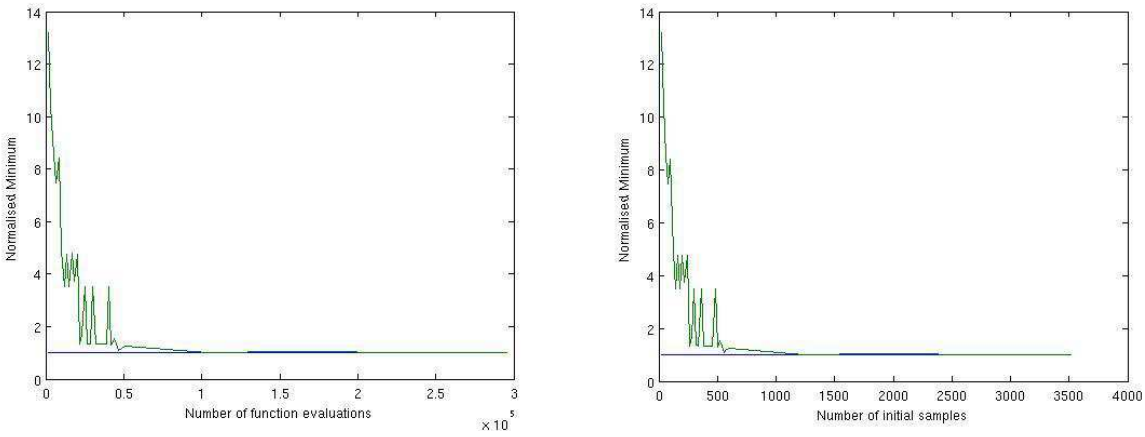



**Figure 4. Convergence of a bi-impulsive direct transfer to Apophis as a function of the total number of function evaluations a) and number of initial samples b).**

finite, the number $N^*$ could be very large. In practice, what we would like is not to choose $N$ large enough so that a success is always guaranteed, but rather, for a fixed $N$ value, we would like to maximize the probability of hitting a global minimizer. Now, let us define the following quantities:

$$\delta_f(\mathbf{x}) = \mid f_{global} - f(\mathbf{x}) \mid; \quad \delta_x(\mathbf{x}) = \|\mathbf{x}_{global} - \mathbf{x}\| \tag{19}$$

In case there is more than one global minimum point, $\delta_x(\mathbf{x})$ denotes the minimum distance between $\mathbf{x}$ and all global minima. Moreover, in case the global minimum point $\mathbf{x}_{global}$ is not known, we can substitute it with the best known point $\mathbf{x}_{best}$. We can now define a new procedure, summarized in Algorithm 7. A key point is setting properly the value of $n$. In fact a value of $n$ too small would correspond to an insufficient number of samples to have a proper statistics. The number $n$ is problem dependent and is related to the complexity of the problem and to the heuristics implemented in the solution algorithm. A proper value for

**Algorithm 7** Convergence to the global optimum

set the max number of function evaluations for $A$ equal to $N$
apply $A$ to $p$ for $n$ times
set $j = 0$
**for all** $i \in [1, ..., n]$ **do**
$\phi(N, i) = \min f(A(N), p, i)$
$\mathbf{x} = arg\ \phi(N, i)$
compute $\delta_f(\mathbf{x})$ and $\delta_x(\mathbf{x})$
    **if** $(\delta_f(\mathbf{x}) < tol_f) \wedge (\delta_x(\mathbf{x}) < tol_x)$ **then** $j = j + 1$
    **end if**
**end for**

$n$ should give a little or null fluctuations on the value of $j/n$, i.e. by increasing $n$ the value of $j/n$ should remain constant or should have a small variation. The choice of $n$ will be discussed in Section IV.2. Note that the values of the tolerance parameter $tol_f$ and $tol_x$ depend on the problem at hand. Algorithm 7 is applicable to general problems either presenting a single solution with value function $f_{global}$ (or $f_{best}$) or presenting multiple solutions with value $f_{global}$ (or $f_{best}$). On the other hand, in the following we are not interested in distinguishing between solutions with equal $f$ and different $\mathbf{x}$ therefore we will use a reduced version of Algorithm 7 in which the condition $\delta_x(\mathbf{x}) \leq tol_x$ is not considered.
Finally, we remark that the two procedures described in Algorithms 6 and 7 only consider the computational cost to evaluate $f$ but not the intrinsic computational cost of $A$. The intrinsic cost of $A$ is related to its complexity and to the number of pieces of information $A$ is handling. For instance, for a simple grid search such intrinsic cost is represented by the cost of sweeping through all the $N$ points on the grid at which the objective function is evaluated. The intrinsic cost varies from algorithm to algorithm, but here we are assuming that the computational effort of the algorithms is dominated by function evaluations and, therefore, we do not take intrinsic costs into account.

## A. Performance Indexes

After the testing procedure have been defined, we move to the definition of performance indexes.
First we note that if the algorithm $A$ is deterministic, then we can set $n = 1$. Indeed, each time $A$ is applied to $p$, it always returns the same value. Then, for deterministic algorithms, given a value $N$, a reasonable performance index is simply $J_d(N) = \phi(N, 1)$, i.e. the best value returned by the algorithm.
Instead, for stochastic based algorithms different performance indexes can be defined. Such indexes are computed by running an algorithm over a problem a sufficiently high number $n$ of times. The indexes should reveal the ability of the algorithm to identify in a given computational time (or computational effort) the best solution to a problem by running an algorithm a single time or, alternatively, by running it several times.
Possible indexes are the best, the mean and the variance of all the results returned by the $n$ runs, or the probability of success of a single run. These will be discussed in what follows.

### *1. Best, Mean and Variance*

A common way to evaluate a stochastic algorithm is to collect the best value over a number of runs and to compute the mean and variance of the best values over the same number of runs. However, the use of best value, mean and variance presents some difficulties:

- The distribution of the best values is not Gaussian, as can be seen in Fig. 5 where the distribution of all the solutions found by PSO1009 over 200 runs for the EVM case is represented. Therefore the distance between the best and the mean values, or the value of the variance in general do not give an exact indication of the repeatability of the result. Moreover, it changes during the process, therefore we cannot define a priori the required number of runs to produce a correct estimation of mean and variance.
- The minimum number of samples that are required to have a sufficient statistics is not well defined for space problems

- The use of the best value could be misleading since statistically even a simple random sampling can converge to the global optimum, on the other hand an algorithm converging on average to a good value with small variance does not guarantee to be able to find the best possible solution.

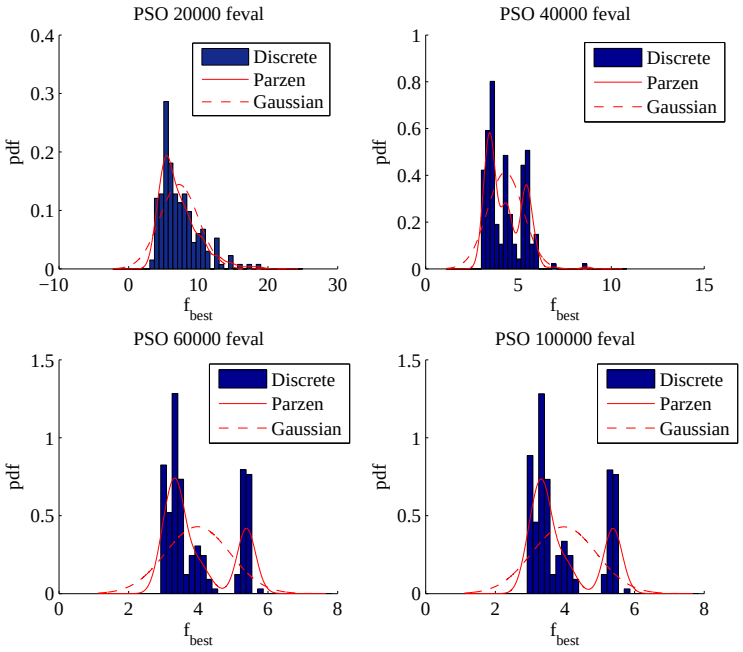



**Figure 5. Probability density function for PSO applied to the solution of the EVM case: discrete, gaussian (dashed) and kernel based approximation (continuous).**

### *2. Success Rate*

An alternative index that can be used to assess the effectiveness of a stochastic algorithm is the success rate $p_s$, which is related to $j$ in Algorithm 7 by $p_s = j/n$. Considering the success as the referring index for a comparative assessment implies two main advantages. First, it gives an immediate and unique indication of the algorithm effectiveness, addressing all the issues highlighted above, and, second, the success rate can be represented with a binomial probability density function (pdf), independently of the number of function evaluations, the problem and the type of optimization algorithm. This latter characteristic implies that the test can be designed fixing a priori the number of runs $n$, on the basis of the error we can accept on the estimation of the success rate. A usual 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 $\theta_p$ is the unknown true proportion of successes, and that the probability of being $p_s$ at distance $d_{err}$ from $\theta_p$, $Pr[|p_s - \theta_p| \leq d_{err}|\theta_p]$ is at least $1 - \alpha_b$ (see Ref.[10]). This leads to expression:

$$n \geq \theta_p(1-\theta_p)\chi^2_{(1),\alpha_b}/d^2_{err} \tag{20}$$

and to the conservative rule:

$$n \geq 0.25\chi^2_{(1),\alpha_b}/d^2_{err} \tag{21}$$

obtained if $\theta_p = 0.5$. For our tests we required an error $\leq 0.05$ ($d_{err} = 0.05$) with a 95% confidence ($\alpha_b = 0.05$), which according to Eq.(21) yields $n \geq 176$, which was extended to 200 for all the tests in the paper to have a higher confidence in the result.
Fig. 6 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 is extremely 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.

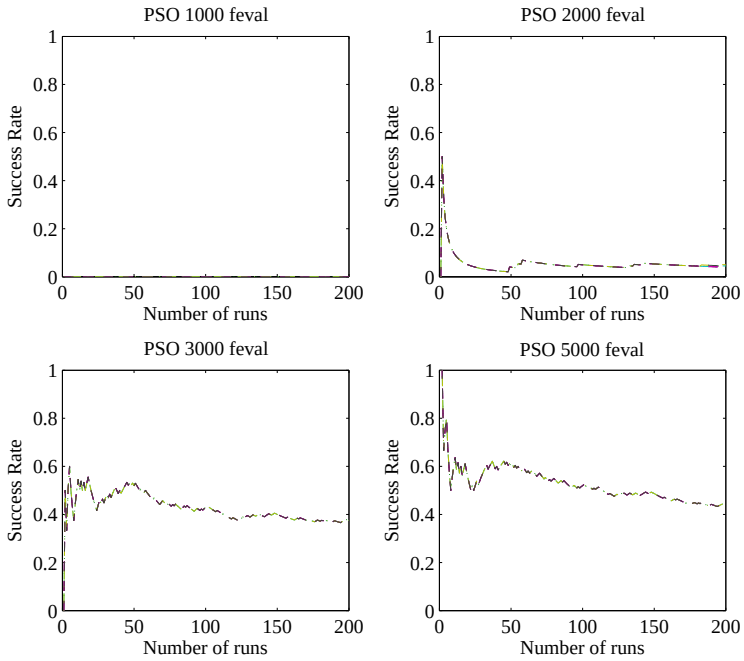



**Figure 6. The influence of sample size. The success rate is shown as function of the number of runs for the PSO applied to the solution of the bi-impulsive case.**

## B. Test Results

The results of the tests are summarized in Table 1 and 2, where success probability (Table 1) and best value, mean and variance of best results (Table 2) are given for each of the 20 settings of the five stochastic solvers. For both EA and EVM cases, success probability allows a fair classification and gives a clear indication of the best performing algorithms. If we look only at the set of evolutionary based algorithms (DE,GA and PSO) algorithms DE10e and DE20e perform undoubtedly better than the others and algorithms GA100/200-GA200/400-GA400/600 appear to be the worst performing ones. Algorithm DE5e wins a bronze medal, but if we can be confident in its third position for the EVM problem, we cannot have the same level of confidence regarding the third position for EA, because of the proximity of other algorithms. Actually, due to the binomial nature of the success and the adopted sample size, it is not possible to fairly discriminate between algorithms for which the success distance is smaller than the expected error (0.05 in our computations). Therefore, algorithm DE10e has to be considered at the same level of algorithms PSO1007, PSO509, for example, and other PSO settings. For the same reasons, we can say that, among the PSO settings, algorithms PSO1007 and PSO509 perform better than algorithm PSO1005 but the remaining PSO algorithms work at the same level.

An analogous vagueness is displayed by most of the evolutionary based algorithms when applied to EVM and almost all of them when applied to EVVEJS and EVVEJSdsm. On the other hand, MS and MBH shows remarkable performance in all the test cases with the simple MS winning over all the others in the EA case. For the EVVEJS case all the algorithms, but MBH, appear practically unsuccessful, because, even if the success probability cannot be considered really 0, due to the error margin, it is $\leq 0.12$, according to the expected error. MBH is the only algorithm that can be practically used to solve this problem with a success $\leq 0.34$. For EVVEJSdsm case even MBH is performing poorly, with a success $\leq 0.08$.

For cases where the success probability cannot give practically useful information to classify the algorithms, the user could be tempted to use the values of mean and variance, but this practice is strongly heedless. Even if we suppose mean and variance are correct (in some way), looking at these two values can bring to incorrect conclusions. For instance, if we consider the values for the algorithms PSO1009 and GA400 applied to EVM, we could conclude that algorithm GA400 performs better than algorithm PSO1009, because of a smaller mean value and a smaller variance (regarded as an index of robustness). But, if we are interested in

**Table 1. Success for the 20 algorithms on the four test-cases. To compute the success, following $tol_f$ values were used: 0.001 for EA, $3 - f_{best}$ for EVM, $5 - f_{best}$ for EVVEJS and $8.5 - f_{best}$ for EVVEJSdsm**

| | DE5c | DE10c | DE20c | DE5e | DE10e | DE20e | PSO505 | PSO1005 | PSO2005 | PSO507 |
|---|---|---|---|---|---|---|---|---|---|---|
| EA | 0.140 | 0.300 | 0.355 | 0.450 | 0.770 | 0.855 | 0.355 | 0.345 | 0.410 | 0.395 |
| EVM | 0.050 | 0.050 | 0.050 | 0.150 | 0.250 | 0.370 | 0.040 | 0.035 | 0.080 | 0.045 |
| EVVEJS | 0.020 | 0.005 | 0.015 | 0.000 | 0.000 | 0.000 | 0.000 | 0.005 | 0.000 | 0.000 |
| EVVEJSdsm | 0.01 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| | PSO1007 | PSO2007 | PSO509 | PSO1009 | PSO2009 | GA100/200 | GA200/400 | GA400/600 | MS | MBH |
| EA | 0.425 | 0.410 | 0.435 | 0.385 | 0.420 | 0.160 | 0.240 | 0.105 | 0.93 | 0.82 |
| EVM | 0.060 | 0.055 | 0.035 | 0.070 | 0.075 | 0.005 | 0.010 | 0.035 | 0.625 | 0.765 |
| EVVEJS | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.005 | 0.07 | 0.29 |
| EVVEJSdsm | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.030 |

global optimal solutions, algorithm PSO1009 is noticeably better: it is able to find the global solution, even if it is less robust and gets stuck many times in a far basin (see Figs. 5 and 7).
In order to solve an uncertainty condition, for instance when the success probability appears uniformly null, relaxing the $tol_f$ value could be useful. Focusing on the EVVEJS case, there is no way to correctly discriminate among the algorithms on the basis of data in table 1, but if the success threshold is raised from 5 to 5.3, then a superior performance of GAs is revealed. Most likely, this behavior is due to a combination of different features, such as larger population, the mutation search operator and a non-deterministic selection operator, which in this complex case reduces the speed of local convergence, preserves diversity and allows for a better exploration of the search space.
For the EVVEJSdsm case we are not so "luky". Raising the success threshold from 8.5 to 9 (see table 3) allows identifying the MBH as the only one algorithm practically able to handle this problem, but does not give any useful information to discriminate among the other algorithms. Only DE "c" series and MS are able to find solutions with objective value under 9, but the success rate is so low and similar that discriminating between the two would be difficult.

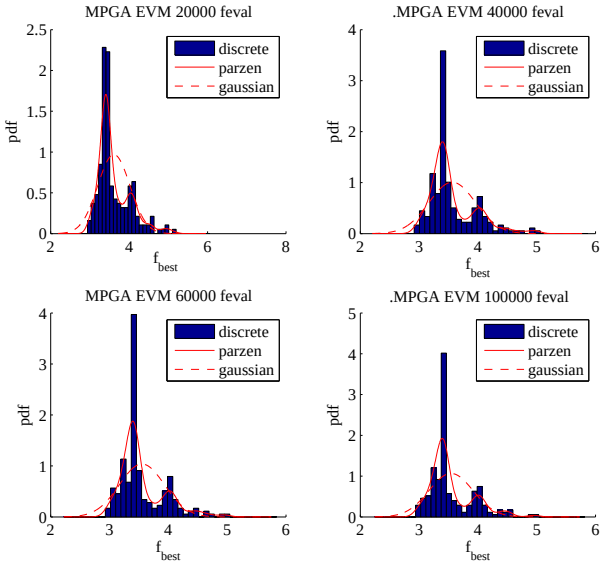



**Figure 7. Pdfs, for the EVM case with the MPGA algorithm.**

In Figs. 8 to 11 we present two interesting cases. Figs. 8 and 9 represent the pdf and the success rate for the MBH algorithm with $\rho = 0.1$ applied to the EVVEJS case with no deep space manoeuvres while Figs. 10 and 11 represent the pdf and the success rate of the DE5c algorithm applied to the same case. MBH distributes

**Table 2. Indexes: Best value, Mean Best, Variance Best.**

| | EA (N=5000) | | | EVM (N=100000) | | | EVVEJS (N=400000) | | | EVVEJSdsm (N=600000) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DE5c | 4.37 | 4.7 | 0.07 | 2.98 | 3.6 | 0.32 | 4.93 | 12.51 | 15.07 | 8.42 | 16.37 | 11.93 |
| DE10c | 4.37 | 4.57 | 0.03 | 2.98 | 3.51 | 0.09 | 4.93 | 11.37 | 15.75 | 8.62 | 16.07 | 11.68 |
| DE20c | 4.37 | 4.52 | 0.02 | 2.98 | 3.43 | 0.08 | 4.93 | 9.97 | 15.93 | 8.7 | 16.02 | 20.33 |
| DE5e | 4.37 | 4.51 | 0.02 | 2.98 | 3.29 | 0.03 | 5.3 | 8.15 | 9.75 | 14.41 | 26.87 | 6.06 |
| DE10e | 4.37 | 4.42 | 0.01 | 2.98 | 3.23 | 0.03 | 5.3 | 6.39 | 5.01 | 21.91 | 28.72 | 2.6 |
| DE20e | 4.37 | 4.39 | 0 | 2.98 | 3.17 | 0.03 | 5.3 | 5.56 | 1.4 | 24.86 | 30.08 | 2.03 |
| PSO505 | 4.37 | 4.51 | 0.01 | 2.98 | 3.82 | 0.68 | 5.03 | 12.68 | 16.49 | 12.42 | 23.13 | 8.52 |
| PSO1005 | 4.37 | 4.51 | 0.01 | 2.98 | 3.78 | 0.63 | 4.96 | 11.97 | 18.7 | 12.48 | 23.32 | 10.46 |
| PSO2005 | 4.37 | 4.5 | 0.01 | 2.98 | 3.65 | 0.52 | 5.3 | 11.19 | 17.92 | 15.32 | 23.74 | 6.67 |
| PSO507 | 4.37 | 4.51 | 0.01 | 2.98 | 4.04 | 1.01 | 5.01 | 11.74 | 17.73 | 11.37 | 22.06 | 9.54 |
| PSO1007 | 4.37 | 4.49 | 0.01 | 2.98 | 3.93 | 0.83 | 5.06 | 10.73 | 17.27 | 13.61 | 22.53 | 7.53 |
| PSO2007 | 4.37 | 4.5 | 0.01 | 2.98 | 3.73 | 0.6 | 5.02 | 10.47 | 18.43 | 10.19 | 22.17 | 8.55 |
| PSO509 | 4.37 | 4.5 | 0.01 | 2.98 | 4.22 | 1.06 | 5.25 | 11.83 | 21.71 | 11.83 | 22.08 | 8.83 |
| PSO1009 | 4.37 | 4.55 | 0.37 | 2.98 | 3.97 | 0.87 | 5.02 | 10.56 | 18.43 | 12.13 | 22.09 | 9.86 |
| PSO2009 | 4.37 | 4.5 | 0.01 | 2.98 | 3.81 | 0.75 | 5.03 | 10.53 | 15.13 | 12.08 | 22.07 | 8.59 |
| GA200 | 4.37 | 4.57 | 0.03 | 2.99 | 3.78 | 0.24 | 5.16 | 10.65 | 15.19 | 9.65 | 19.98 | 13.53 |
| GA400 | 4.37 | 4.5 | 0.01 | 2.99 | 3.54 | 0.14 | 5.02 | 8.31 | 9.91 | 9.1 | 18.6 | 13.35 |
| GA600 | 4.37 | 4.45 | 0.01 | 2.98 | 3.45 | 0.1 | 4.98 | 6.98 | 6.68 | 10.74 | 18.36 | 10.91 |
| MS | 4.37 | 4.39 | 0 | 2.98 | 3.02 | 0 | 4.94 | 5.28 | 0.03 | 8.62 | 14.52 | 4.92 |
| MBH | 4.37 | 4.41 | 0 | 2.98 | 3.01 | 0 | 4.93 | 5.19 | 0.39 | 8.41 | 12.64 | 9.74 |

**Table 3. Successes for the 20 algorithms on the EVVEJSdsm test-case, with the threshold varying from $8.5 - f_{best}$ to $9.0 - f_{best}$**

| | DE5c | DE10c | DE20c | DE5e | DE10e | DE20e | PSO505 | PSO1005 | PSO2005 | PSO507 |
|---|---|---|---|---|---|---|---|---|---|---|
| $8.5 - f_{best}$ | 0.01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $8.6 - f_{best}$ | 0.02 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $8.7 - f_{best}$ | 0.05 | 0.04 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $8.8 - f_{best}$ | 0.06 | 0.06 | 0.02 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $8.9 - f_{best}$ | 0.06 | 0.06 | 0.05 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $9.0 - f_{best}$ | 0.06 | 0.06 | 0.06 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | PSO1007 | PSO2007 | PSO509 | PSO1009 | PSO2009 | GA100/200 | GA200/400 | GA300/600 | MS | MBH |
| $8.5 - f_{best}$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.03 |
| $8.6 - f_{best}$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.07 |
| $8.7 - f_{best}$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.02 | 0.15 |
| $8.8 - f_{best}$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.02 | 0.19 |
| $8.9 - f_{best}$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.02 | 0.22 |
| $9.0 - f_{best}$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.02 | 0.23 |

the solutions over few distant minima for a low number of function evaluations but then quickly converge only to a neighborhood of the best known solution as soon as the number of function evaluations increases. DE5c instead maintain a practically unchanged distribution of the solutions. In fact DE5c converges very fast and then tend to remain trapped in local minima.

Figs.12(a) to 13(b) represent the distribution of all the local minima in the search space, found by all the search algorithms over all the runs. In particular, we defined some specific intervals (or levels) of values for the objective function and then we computed: the average value of the relative distance of a given local

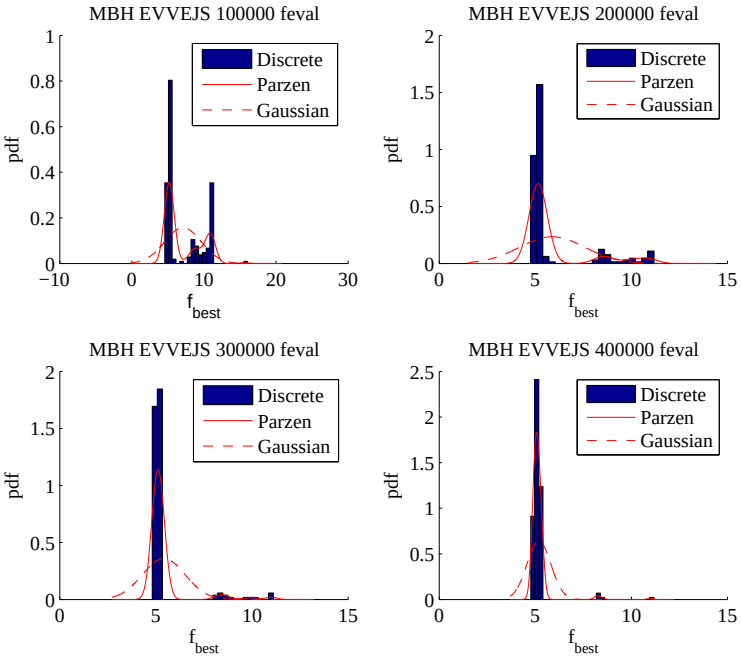



**Figure 8. Pdf for the MBH applied to the solution of the EVVEJS case with no DSMs.**

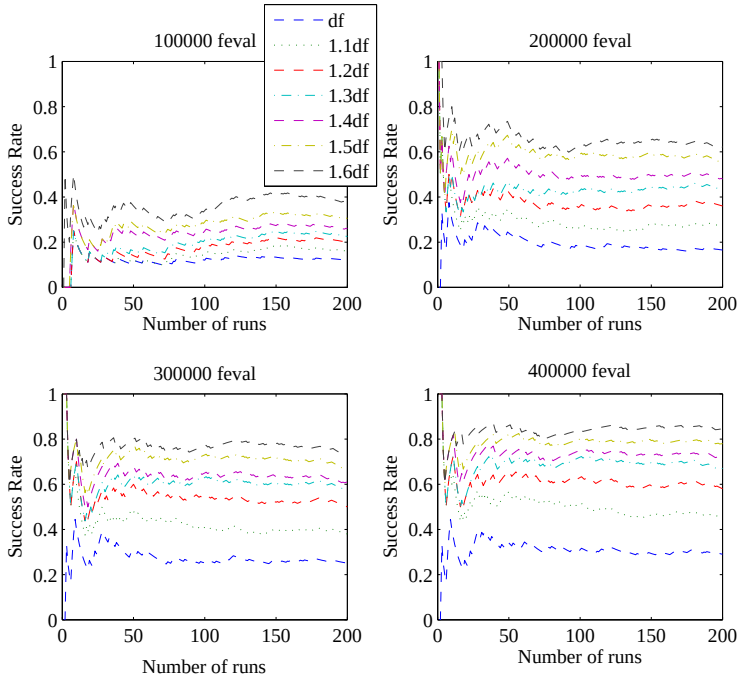



**Figure 9. Success rate of the MBH applied to the solution of the EVVEJS case with no DSMs.**

minimum with respect to all the local minima in the same interval of objective values $d_{il}$ (or intra-level distance) and the average value of the relative distance of a given local minimum with respect to all the local minima in the interval with lower values of the objective function $d_{tl}$ (or trans-level distance). The figures give an immediate representation of the diversity of the local minima in the search space (different colors

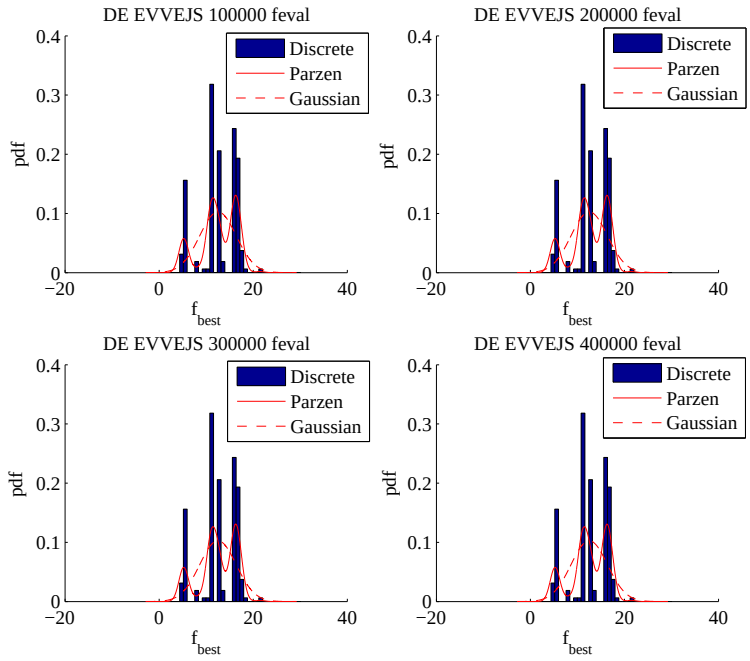



**Figure 10. Pdf for the DE applied to the solution of the EVVEJS case with no DSMs.**

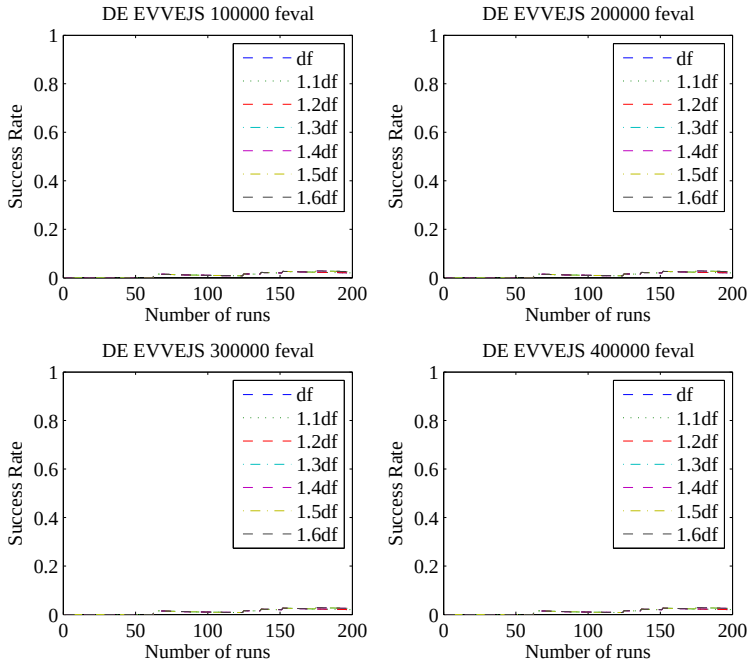
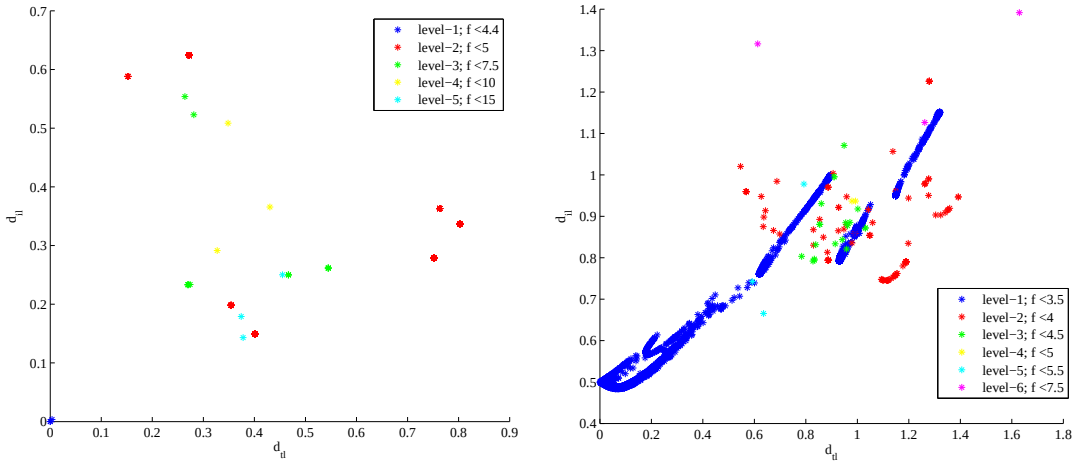


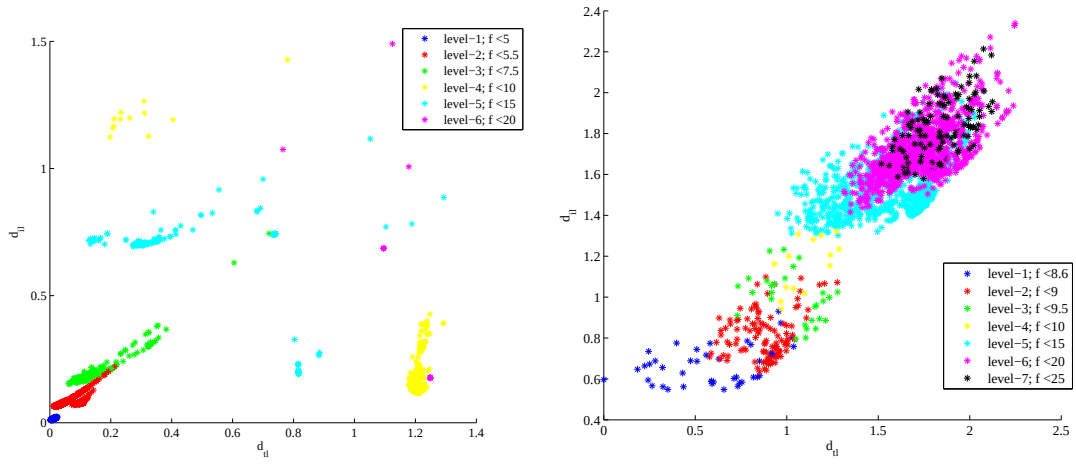

**Figure 11. Success rate of the DE applied to the solution of the EVVEJS case with no DSMs.**

correspond to different levels). Note that the two quantities $d_{il}$ and $d_{tl}$ can be related to the Shannon's diversity index.[23] In fact, if both $d_{tl}$ and $d_{il}$ are large then the solutions belonging to each species (the intervals of $f$) are well spread and distant from the solutions of the lower level. We can then redefine the species and associate a species to each solution and the value of the diversity index would be high. If both $d_{tl}$ and $d_{il}$ are small the solutions are clustered and close to each other, therefore species are made of large




(a) (b)

**Figure 12. Relative distance of the local minima for the bi-impulsive and EVM cases.**




(a) (b)

**Figure 13. Relative distance of the local minima for the EVVEJS with and without DSM's.**

groups of solutions and the diversity index would be low. The index would be low also for $d_{il}$ small and $d_{tl}$ large and would be large for $d_{il}$ large and $d_{tl}$ small. It should be noted that $d_{tl}$ and $d_{il}$ give a better representation of the actual diversity because they consider also the reciprocal distance in the solution space while the Shannon's index is generally associate only to the separation in the criteria space.[26]

More precisely, Fig.12(a) is telling us that for the bi-impulsive case the minima are quite spread, not clustered if not in pairs. The EVM case, in Fig.12(b), presents an almost continuous distribution of minima. However, the minima of one level appear to be quite distant from the minima of the lower level.

Fig.13(a) presents a quite different structure. There are a number of clusters and in particular three of them, corresponding to level 1,2 and 3, have values of $d_{tl}$ and $d_{il}$ lower than 0.2. The existence of clusters of minima with low $d_{tl}$ and low $d_{il}$ suggests an easy transition from one level to an another. An easy transition among

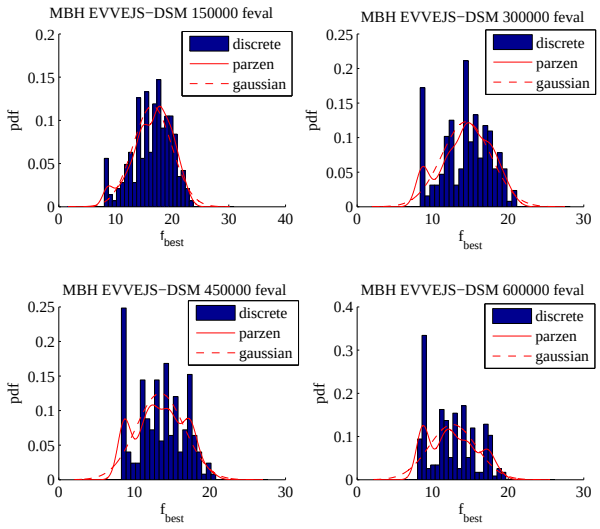



**Figure 14. Examples of pdfs, for the EVVEJS case with DSM and the MBH optimizer.**

levels favors the search mechanism of MBH and represent a clue of a possible underlying funnel structure.[20] Both Figs. 13(b) and (14) suggest, instead, a different landscape for the EVVEJSdsm case. The solutions are quite spread and with values of both $d_{tl}$ and $d_{il}$ higher than 0.5. This case appears to be highly multi-modal, with relatively distant minima. Gradient based methods (MS and MBH) converge to local minima and get stuck. Even if the re-sampling allows the MBH to better converge to the global solution, the variance of the best solutions is still very high. Note that in the case of the EVVEJS problem with no deep-space manoeuvre running the MBH with no re-sampling provides an increase in performance from 29% to 37.5%.

## V. A Dynamical System Prospective

In this section we look at some evolutionary heuristics from a different prospective in the attempt to better understand the dynamics of the search process regardless of the problem under investigation. This analysis will be used to improve the performance of one of the algorithms tested above. In particular, we note that both DE and PSO can be rewritten in a compact form as a discrete dynamical system:

$$\begin{aligned} \mathbf{v}_{i,k+1} &= (1-c)\mathbf{v}_{i,k} + \mathbf{u}_{i,k} \\ \mathbf{x}_{i,k+1} &= \mathbf{x}_{i,k} + \nu S(\mathbf{x}_{i,k} + \mathbf{v}_{i,k+1})\mathbf{v}_{i,k+1} \end{aligned} \tag{22}$$

with

$$\nu = \min([v_{max}, v_{i,k+1}])/v_{i,k+1} \tag{23}$$

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. Eq. (23) represents a limit sphere around the point $\mathbf{x}_{i,k}$ at stage $k$ of the search process.
In addition to Eq. (22) and Eq. (23) each optimization algorithm has heuristics responsible for selecting the new candidate points generated with $\mathbf{u}_{i,k}$. The 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 the $\mathbf{u}_{i,k}$ defined by Eq.(13), viscosity $c = 1$ and $v_{max} = +\infty$. The selection function $S$ can be either 1 or 0 depending on the relative value of the objective function of the new candidate individual generated with Eq.(13) with respect to the one of the current individual (see Algorithm 2).Particle swarm optimization has the $\mathbf{u}_{i,k}$ defined in Eq.(16) with the viscosity term $c = 1 - [0.4 + 0.8(k_{max} - k)/(k_{max} - 1)]$, no mask and no selection, therefore $S$ is always equal to 1 but $v$ is constrained by Eq. (23). Note that if we take $c_2\mathbf{r}_2 = F\mathbf{e}$, $c_1\mathbf{r}_1 = \mathbf{e}$ and replacing $\mathbf{x}_{gi,k}$ with one of the individuals in the population, then PSO translates into DE and vice versa we can go from DE to PSO just

by defining properly the selection of the individuals $\mathbf{x}_{i_1,k}, \mathbf{x}_{i_2,k}, \mathbf{x}_{i_3,k}$, the value of the coefficients $c, c_1, c_2, \nu$ and the selection function $S$.
The discrete dynamical system in Eqs. (22) can be rewritten in matrix form as follows:

$$\left[ \begin{array}{c} \mathbf{x}_{i,k+1} \\ \mathbf{v}_{i,k+1} \end{array} \right] = \mathbf{J}_{i,k} \left\{ \begin{array}{c} \mathbf{x}_{i,k} \\ \mathbf{v}_{i,k} \end{array} \right\} \tag{24}$$

and if we consider all the individuals in the population:

$$\left[ \begin{array}{c} \mathbf{x}_{k+1} \\ \mathbf{v}_{k+1} \end{array} \right] = \mathbf{J}_{k} \left\{ \begin{array}{c} \mathbf{x}_{k} \\ \mathbf{v}_{k} \end{array} \right\} \tag{25}$$

The map (25) 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:

- 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 the global optimization algorithm is simply convergent in $D$ and we can define a stopping criterion. Once the search is stopped we can define a restart procedure. Depending on the convergence profile, the use of a restart procedure can be more or less efficient.
- 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.
- Converge to a strange (chaotic) attractor. In this case a stopping criterion cannot be clearly defined because different points are sampled at different iterations.

We have seen in the previous sections that the heuristics implemented in MBH is particularly effective. MBH is based on a Newton (or quasi-Newton) method for local minimization and on a restart of the search within a neighborhood $N_\rho$ 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* $J_k$. Under suitable assumptions the systems converge to a fixed point. For instance, if $f$ is convex and $C^2$ 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 $D_k$.
The motivations for using different dynamical systems are: dropping the requirement for the continuity and differentiability of $f$; automatically reducing the size of the region in which a candidate point is generated (the basic version of MBH has a constant size of $N_\rho$); performing not only a local exploration of the neighborhood, but also a global one.
For instance, in the specific case of simple Differential Evolution, $c = 1$ and $\nu = 1$, therefore we have the reduced map:

$$\mathbf{x}_{i,k+1} = \mathbf{x}_{i,k} + S(\mathbf{x}_{i,k} + \mathbf{u}_{i,k})\mathbf{u}_{i,k} \tag{26}$$

or in matrix form for the entire population $\mathbf{x}_{k+1} = \mathbf{J}_k \mathbf{x}_k$. The interest is now in the properties of map (26). We start by observing that if $S(\mathbf{x}_k + \mathbf{u}_k) = 1 \Leftrightarrow f(\mathbf{x}_k + \mathbf{u}_k) < f(\mathbf{x}_k)$ the global minimizer $\mathbf{x}_g \in D$ is a fixed point for (26) since every 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 $D_k$ and $D_k^*$ of $D$ 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 $P_k \in D_k$ while $P_{k+1} \in D_k^*$. If $\mathbf{x}_l$ is the lowest local minimum in $D_k$, then $\mathbf{x}_l$ is a fixed point in $D_k$ for (26). In fact, every point generated by (26) (or (25)) must be in $D_k$ and $f(\mathbf{x}_l) < f(\mathbf{x}), \forall \mathbf{x} \in D_k$
Now we want to know if there are other fixed points for the dynamics (26) and if we can always have a simple convergence in $D_k$. First of all we note that if for every $k$, $P_k \in D_k$ and $P_{k+1} \in D_k^*$, then the reciprocal distance of the individuals cannot grow indefinitely because of the map (26), therefore the map cannot be divergent.
Then, if we assume that the function $f$ is strictly quasi-convex[25] in $D_k$ we can prove that (26) converge to a fixed point in $D_k$. We first need the following lemma.

LEMMA V.1 *If $f$ is continuous and strictly quasi-convex on a compact set $D_j$, the following minimization problem with $F \in (0,1)$ has a strictly positive minimum value $\delta_r(\epsilon)$:*

$$\begin{aligned} \delta_r(\epsilon) = \quad & \min \quad g(\boldsymbol{y}_1, \boldsymbol{y}_2) = f(\boldsymbol{y}_2) - f(F\boldsymbol{y}_1 + (1-F)\boldsymbol{y}_2) \\ & s.t. \quad \boldsymbol{y}_1, \boldsymbol{y}_2 \in D_k \\ & \quad\;\; \|\boldsymbol{y}_1 - \boldsymbol{y}_2\| \geq \epsilon \\ & \quad\;\; f(\boldsymbol{y}_1) \leq f(\boldsymbol{y}_2) \end{aligned} \tag{27}$$

**Proof** Since $f$ is strictly quasi-convex $g(\mathbf{y}_1,\ \mathbf{y}_2) > 0,\ \forall \mathbf{y}_1,\ \mathbf{y}_2 \in D$; 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 by $(\mathbf{y}_1^*, \mathbf{y}_2^*)$ a global minimum point of the problem, then we have

$$\delta_r(\epsilon) = g(\mathbf{y}_1^*, \mathbf{y}_2^*) > 0. \tag{28}$$

THEOREM V.2 *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 $S(\boldsymbol{x}_k + \boldsymbol{u}_k) = 1 \Leftrightarrow f(\boldsymbol{x}_k + \boldsymbol{u}_k) < f(\boldsymbol{x}_k)$ , the population $P_k$ converges to a fixed point in $D_k$ for $k \to \infty$.*

**Proof** We propose two distinct proofs for this result. The first one is simpler but requires an additional assumption. The second one is more complicated but also more general.

The first simpler proof requires the additional assumption that the population always has an individual $\mathbf{x}_{j,k}$ strictly better than the others, i.e. $f(\mathbf{x}_{j,k}) < f(\mathbf{x}_{i,k})$ for any $i \neq j$, then the map (26) at each iteration $k$ can generate with strictly positive probability a displacement $(\mathbf{x}_{j,k} - \mathbf{x}_{i,k})$ for all the members of the population. 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 probability one.

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

$$\inf_k \max\{\|\mathbf{x}_{i,k} - \mathbf{x}_{j,k}\|, i, j \in [1, ..., n_{pop}]\} \geq \epsilon > 0 \tag{29}$$

At every generation $k$ the map can generate with 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 (29) and of Lemma (V.1) we must have that

$$f(\mathbf{x}_{cand}) \leq f(\mathbf{x}_{j,k}) - \delta_r(\epsilon). \tag{30}$$

Such reduction will occur with probability one infinitely often, and consequently the function value of at least one individual will be, with probability one, infinitely often reduced by $\delta_r(\epsilon)$. But this way the value of the objective function of such 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, when the population of DE lies at each iteration 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 certainly converge to a fixed point. For general functions, we can not always guarantee that the population will converge to a fixed point, but we can show that the maximum difference between the objective function values in the population converges to 0, i.e. the points in the population tend to belong to the same level set.

THEOREM V.3 *Given a function $f$ and a population $P_k \in D_k$, then if $F \in (0,1)$ and $S(\boldsymbol{x}_k + \boldsymbol{u}_k) = 1 \Leftrightarrow f(\boldsymbol{x}_k + \boldsymbol{u}_k) < f(\boldsymbol{x}_k)$ , the following holds*

$$\max_{i,j \in [1,...,n_{pop}]} \ | f(\boldsymbol{x}_{j,k}) - f(\boldsymbol{x}_{i,k}) | \to 0,$$

*as $k \to \infty$.*

**Proof** Let $S_k^*$ denote the set of best points in population $P_k$, i.e.

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

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 made up by points all with the same objective function value. Therefore, such event will occur infinitely often with probability one. Let us denote with $\{k_h\}_{h=1,\ldots}$ the infinite subsequence of iterations at which the event is true, and let

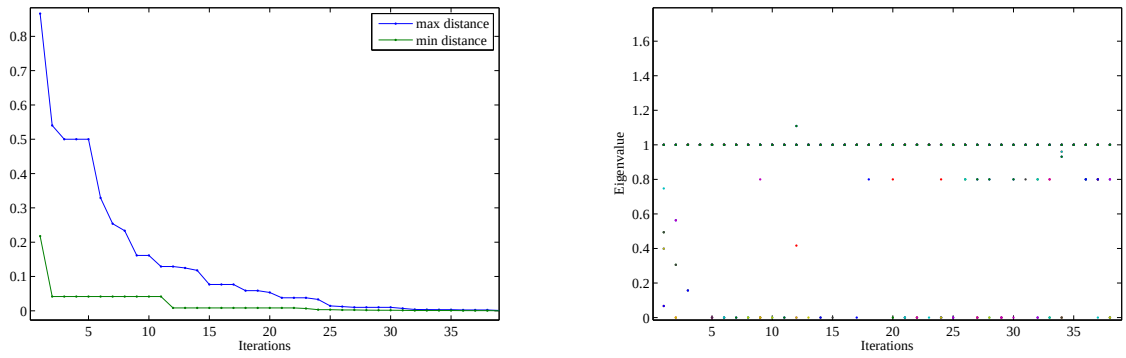
$$\Delta_h = f(\mathbf{x}_{i,k_h}) - f(\mathbf{x}_{i,k_{h+1}})$$

be the difference of the objective function values at two consecutive iterations $k_h$ and $k_{h+1}$ (note that, since at iterations $k_h$, $h = 1, \ldots$ the objective function values are all equal, any $i$ can be employed in the above definition). It holds that for all $i, j \in [1, \ldots, n_{pop}]$

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

Therefore, if we are able to prove that $\Delta_h \to 0$, as $h \to \infty$, we also prove the result of our theorem. Let us assume by contradiction that $\Delta_h \not\to 0$. Then, there will exist a $\delta > 0$ such that $\Delta_h \geq \delta$ 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, though it is true according to V.3 that we can converge to a level set with $\Delta_h = 0$ it is not true if $\Delta_h = 0$ the algorithm has converged. Therefore, since the most likely situation is the convergence to a single point, we can alternatively use as a stopping criterion the fact that the maximum distance between points in the population drops below a threshold.




(a) (b)

**Figure 15. 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. (26) we can look at the eigenvalues of the matrix $J_k$.
If the population cannot diverge the eigenvalues cannot have norm always $> 1$. Furthermore, according to theorem (V.2) 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 (26) is a contraction in $D_k$ and therefore the eigenvalues should have norm on average lower than 1. This can be illustrated with the following test. We consider a population of 8 individuals and a $D_k$ enclosing the minimum of a paraboloid with minimum in the origin. We compute, for each step $k$, the distance of the closest and farthest individual form the local minimum and the eigenvalues of the matrix $\mathbf{J}$. Fig.15 shows the behavior of the eigenvalues and of the distance from the origin. From

**Algorithm 8** 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 $\nu$
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: $\rho_A = \max(\|\mathbf{x}_{i,k} - \mathbf{x}_{j,k}\|)$ for $\forall \mathbf{x}_{i,k}, \mathbf{x}_{j,k} \in P_{sub} \subseteq P_k$
14: **if** $\rho_A < tol_{conv}$ **then**
15: Define a bubble $D_l$ such that $\mathbf{x}_{i,k} \in D_l$ for $\forall \mathbf{x}_{i,k} \in P_{sub}$ and $P_{sub} \subseteq P_k$
16: $A_g = A_g + \{\mathbf{x}_{best}\}$ where $\mathbf{x}_{best} = \arg\min_i f(\mathbf{x}_{i,k})$
17: Initialize $\mathbf{x}_{i,k}$ and $\mathbf{v}_{i,k}$ for all $i \in [1, ..., n_{pop}]$, in the bubble $D_l \subseteq D$
18: **end if**
19: **Termination** Unless $n_{feval} \geq n_{fevalmax}$, *goto* Step 3

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 eigenvalue is lower than 1 the population contracts as represented in Fig.15a.

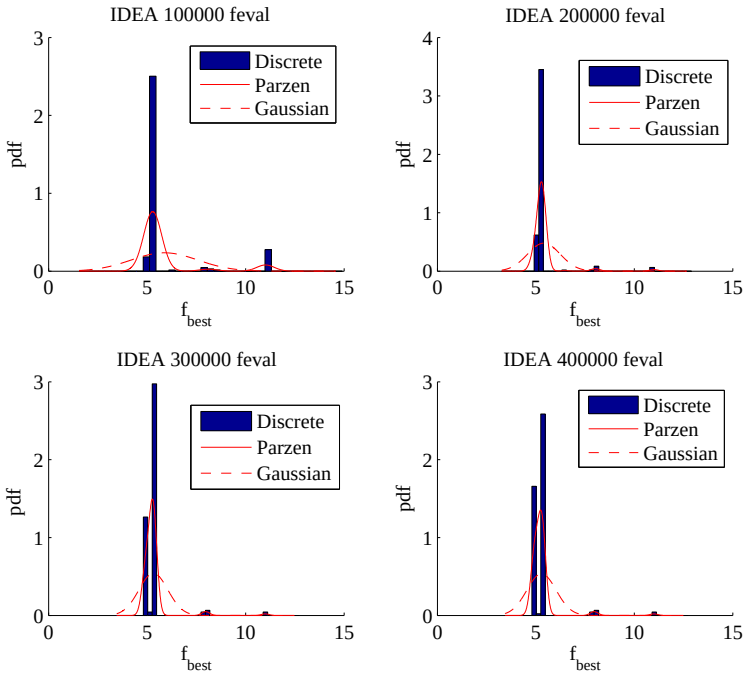



**Figure 16. Pdf for IDEA applied to the solution of the EVVEJS case with no DSMs.**

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 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

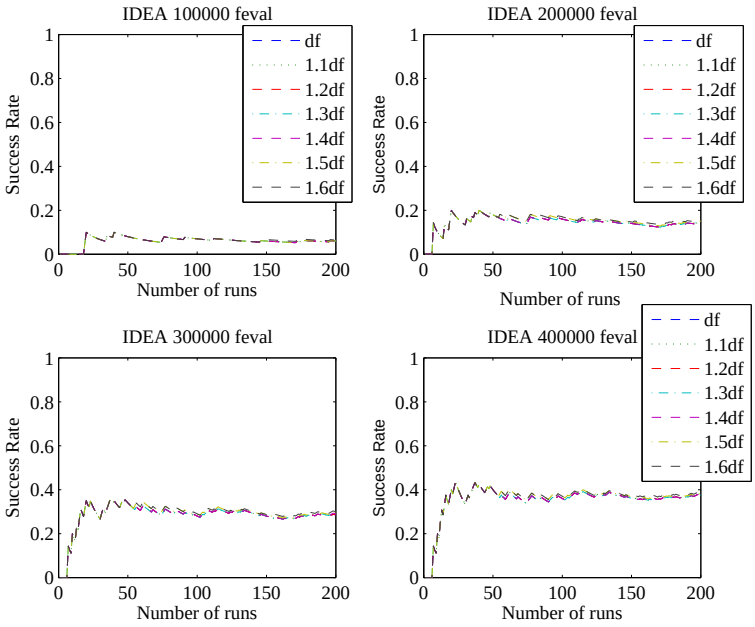



**Figure 17. Success rate of IDEA applied to the solution of the EVVEJS case with no DSMs.**

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 value $tol_{conv}$. Every time a subpopulation is re-initialized the best solution $\mathbf{x}_{best}$ of the cluster is saved in an archive $A_g$. This process leads to the modified DE algorithm 8. Note that the contraction of the population given for example by the metric $\rho_A$ is a stopping criterion that does not depend explicitly on the value of the objective function but on the contractive properties of the map in Eq.(26).

The new algorithm was tested again on the EVVEJS case without DSM. Instead of restarting each cluster we restarted the whole population when maximum distance among its individuals was below $tol_{conv}$. The result is represented in Figs. 16 and 17. With respect to the original version of DE (see Fig. 11) the improvement in the success rate is remarkable and reaches almost 40%. Unlike the MBH version in Fig. 8 for IDEA we did not operate a re-sampling over the entire $D$ when no improvement was registered. This could be the reason for the lower success rate when the tolerance $tol_f$ is increased. On the other hand, we should remark that IDEA manages to converge to the best known solution, while MBH remains slightly above that value (both with and without re-sampling). This is due to the fact that in a small neighborhood of the best known solution there is a large number of local minima, all with similar values but irregularly distributed. In this case the ability of the DE map to adapt automatically the size of the region in which it generates the candidate points, represents an advantage over the fixed neighborhood $N_\rho$ of MBH.

## VI. Conclusion

In this paper we presented an approach to test global optimization algorithms applied to the solution of space trajectory design problems. We discussed the relevance of some performance indexes, in particular mean, variance, best value and success rate. The last one offers a better and more precise estimation of the actual performance of an algorithm when applied to a standard benchmark. The selection of the test cases in the benchmark is quite important since the characteristics of the problem, and therefore the performance of a given algorithm, can change significantly even by changing the modeling of the same problem, as demonstrated by the two models for the same EVVEJS transfer. The use of the success rate offers a distinctive advantage of a precise a priori definition of the required number of runs to correctly estimate the success rate. The use of mean and variance, instead, showed to be misleading in some cases and inconclusive in others since the distribution of the results was demonstrated to be non-Gaussian.

Finally we presented a dynamical system interpretation of some evolutionary heuristics. By analyzing some

dynamical properties of differential evolution in relation to the characteristics of the problems in the benchmark we derived a new algorithm that outperformed the standard DE and all the tested EA, becoming competitive against the best performing tested stochastic method, the MBH.
An interesting result of the analysis presented in this paper is that even a single specific heuristic can improve significantly the performance of a search method applied to a particular class of problems. On the other hand, this fact highlights the importance of a proper coupling between heuristics and problem structure which is not always possible a priori.

## 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]D. R. Myatt, V.M. Becerra, S.J. Nasuto, and J.M. Bishop, "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]A.V. Labunsky, O.V. Papkov, K.G. Sukhanov. *Multiple Gravity Assist Interplanetary Trajectories*. ESI Book Series, 1998.

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

[16]S.H. Strogatz. *Nonlinear Dynamics and Chaos*. Westview Press, 2000.

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

[18]C. D. Perttunen and B. E. Stuckman, Lipschitzian Optimization Without the Lipschitz Constant, *JOTA*, 79(1), (1993) 157-181.

[19]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).

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

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

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

[23]Shannon, C.E. A mathematical theory of communication. July and October 1948, Bell System Technical Journal 27: 379-423 and 623-656

[24]Vasile I. Istratescu, Fixed Point Theory, An Introduction, D.Reidel, the Netherlands 1981. ISBN 90-277-1224-7 Chapter 7.

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

[26]Liu S-H, Mernik M., Bryant B.B. Entropy-Driven Parameter Control for Evolutionary Algorithms. Informatica 31 (2007) 4150.

[27]Vasile M., Locatelli M. A Hybrid MultiAgent Approach for Global Trajectory Optimization. Journal of Global Optimization, 2008, in press.