EOS: a Parallel, Self-Adaptive, Multi-Population Evolutionary Algorithm for Constrained Global Optimization

Lorenzo Federici
Department of Mechanical and Aerospace Engineering
Sapienza University of Rome
Rome, Italy
lorenzo.federici@uniroma1.it

Boris Benedikter
Department of Mechanical and Aerospace Engineering
Sapienza University of Rome
Rome, Italy
boris.benedikter@uniroma1.it

Alessandro Zavoli
Department of Mechanical and Aerospace Engineering
Sapienza University of Rome
Rome, Italy
alessandro.zavoli@uniroma1.it

Abstract—**This paper presents the main characteristics of the evolutionary optimization code named *EOS*, Evolutionary Optimization at Sapienza, and its successful application to challenging, real-world space trajectory optimization problems. *EOS* is a global optimization algorithm for constrained and unconstrained problems of real-valued variables. It implements a number of improvements to the well-known Differential Evolution (DE) algorithm, namely, a self-adaptation of the control parameters, an epidemic mechanism, a clustering technique, an ε-constrained method to deal with nonlinear constraints, and a synchronous island-model to handle multiple populations in parallel. The results reported prove that *EOS* is capable of achieving increased performance compared to state-of-the-art single-population self-adaptive DE algorithms when applied to high-dimensional or highly-constrained space trajectory optimization problems.**

Index Terms—**global optimization, evolutionary optimization, constrained optimization, differential evolution, self-adaptation, parallel computing, island-model, space trajectory optimization**

I. INTRODUCTION

Evolutionary Optimization at Sapienza, or *EOS*, is an evolutionary optimization algorithm for continuous-variable problems developed at the Department of Mechanical and Aerospace Engineering of Sapienza University of Rome. Its origin dates back to 2015, when a first version of the algorithm was coded with the aim of tackling the "nearly-impossibile" interplanetary trajectory optimization problems proposed in the Global Trajectory Optimization Competitions [1]. Since then, *EOS* has been continuously updated and improved, and applied with success to a broad range of unconstrained and constrained space trajectory optimization problems, as multiple gravity-assist trajectories [2], rocket ascent trajectories [3], [4], and multi-rendezvous missions [5], [6].

EOS implements a multi-population, self-adaptive, ε-constrained Differential Evolution (DE) algorithm, with a synchronous island-model for parallel computation. DE is a well-known population-based evolutionary algorithm, devised by R. Storn and K. Price in 1997 [7] to find the global optimum of nonlinear, non-differentiable functions of real-valued variables.

Despite its simplicity, DE exhibits much better performance in comparison with several other meta-heuristic algorithms on a wide range of benchmark and real-world optimization problems, defined over a continuous parameter space [8]. As other Evolutionary Algorithms (EAs) and Genetic Algorithms (GAs), DE exploits the crossover, mutation, and selection operators to generate new candidate solutions, or individuals, and to decide on their survival in successive generations. Unlike traditional EAs and GAs, each mutated solution is generated as a scaled difference of a number of distinct individuals of the current population. This self-referential mutation has the desirable property to automatically adapt the different variables of the problem to their natural scale in the solution landscape, boosting the search potential of the algorithm [9]. All these evidences contributed to selecting DE as the optimization core of *EOS*.

The standard DE algorithm is neither a globally nor a locally convergent algorithm, that is, there exist instances for which it is not theoretically able to identify locally optimal solutions [10]. Nevertheless, DE has proven to be capable of attaining high quality results in many practical optimization problems. However, its performance may drop considerably in complex optimization environments, characterized by nonlinear constraints, numerous isolated local minima, and a high-dimensional search space, all typical characteristics of space trajectory optimization problems, which are the kind of problems *EOS* has been designed to deal with. For this reason, many researchers focused on modifying the standard DE algorithm in order to improve its effectiveness when applied to hard, constrained, global optimization problems. Several DE variants have thus been proposed, most of which are collected in [9], [11]. Four weaknesses of the classical DE algorithm have been particularly targeted by researchers: (i) (nonlinear) constraint handling [12], (ii) the tuning of the few DE control parameters that drive the evolution process [13], (iii) the lack in diversity among individuals in the population along generations [14], and (iv) the need to find a balance between

a wide exploration of the solution space and a quick refinement (or exploitation) of the previously obtained solutions [15].

All these aspects have been addressed in *EOS* by combining some of the most successful ideas found in the literature. Specifically, five major add-ons have been implemented in *EOS* to improve the performance of the standard DE algorithm: (i) a ε-constrained method, to deal with (possibly nonlinear) constraints, (ii) a self-adaptation of the control parameters, (iii) an epidemic mechanism, to maintain diversity within the population during the evolution, (iv) a pruning of the worst sections of the solution space, to speed up the convergence process, and (v) a synchronous, multi-mutation, island-model, to achieve a proper balance between exploration and exploitation of the search space. Moreover, the code has been made parallel by using a hybrid MPI-OpenMP programming, to obtain reasonable computation times even in presence of high dimensional problems and/or computationally expensive cost functions.

The paper is organized as follows. After the basic notation of optimization problems is introduced in Sec. II, the standard DE algorithm and its operators are briefly described in Sec. III. The following sections give a detailed description of the improvements made in *EOS* to DE, i.e., the self-adaptive scheme (Sec. IV), the epidemic mechanism (Sec V), the pruning technique (Sec VI), constraint handling (Sec. VII) and the parallel island-model (Sec. VIII). Finally, successful applications of *EOS* to real-world problems borrowed from the space sector, that is, the optimization of a Multiple Gravity-Assist (MGA) trajectory, of the ascent trajectory of a multi-stage launcher and of an Active Debris Removal (ADR) trajectory, are reported in Sec. IX.

II. OPTIMIZATION PROBLEM

Given a problem described by D real-valued variables:

$$\mathbf{x} = \left[x^{(1)}, \ldots, x^{(D)} \right] \tag{1}$$

and a cost function $f(\mathbf{x})$, the aim of an optimization process is to find the vector $\mathbf{x}^*$ that minimize $f(\mathbf{x})$:

$$\mathbf{x}^* = \underset{\mathbf{x} \in \Omega}{\operatorname{argmin}} f(\mathbf{x}) \tag{2}$$

where Ω represents the solution space. In *unconstrained optimization problems*, Ω is a D-dimensional hyperrectangle, defined as the Cartesian product of the variable bounding intervals $\mathbf{x}_L = \left[x_L^{(1)}, \ldots, x_L^{(D)} \right]$ and $\mathbf{x}_U = \left[x_U^{(1)}, \ldots, x_U^{(D)} \right]$:

$$\Omega = \Omega_b = \left\{ \mathbf{x} \in \mathbb{R}^D : \mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U \right\} \subset \mathbb{R}^D \tag{3}$$

In the case of *constrained optimization problems*, the solution space Ω is further reduced by the presence of K inequality constraints $\mathbf{\Psi}(\mathbf{x}) = \left[\Psi^{(1)}(\mathbf{x}), \ldots, \Psi^{(K)}(\mathbf{x}) \right]$, which in general are nonlinear functions of the design variables:

$$\Omega = \left\{ \mathbf{x} \in \Omega_b : \mathbf{\Psi}(\mathbf{x}) \leq \mathbf{0} \right\} \subset \mathbb{R}^D \tag{4}$$

The possible presence of equality constraints is already included in expression (4); indeed, any equality constraint of the form: $\Phi^{(k)}(\mathbf{x}) = C$ can be rewritten as an inequality constraint if an arbitrarily small tolerance δ is introduced:

$$\Psi^{(k)}(\mathbf{x}) := |\Phi^{(k)}(\mathbf{x}) - C| \leq \delta \tag{5}$$

III. STANDARD DIFFERENTIAL EVOLUTION

A brief description of the standard DE algorithm is here provided. Let us consider the unconstrained minimization problem defined by Eqs. (2) and (3). An initial collection (or *population*) pop^0 of N_p candidate solution vectors (or *individuals*) is generated by randomly sampling the solutions, as evenly as possible, in the solution space:

$$pop^0 = \{\mathbf{x}_i \in \Omega\}_{i=1,\ldots,N_p} \tag{6}$$

where:

$$x_i^{(j)} = x_L^{(j)} + p_i^{(j)} \left(x_U^{(j)} - x_L^{(j)} \right) \tag{7}$$

for $j = 1, \ldots, D$, where $p_i^{(j)}$ indicates a random number with uniform distribution in $[0, 1]$. The value of the cost function (or *fitness*) $f(\mathbf{x}_i)$ is evaluated for each individual (or *agent*) $\mathbf{x}_i$ composing the initial population.

At any iteration G (or *generation*) of the algorithm, a new population pop^{G+1} is created by applying to each vector $\mathbf{x}_i \in pop^G$ a sequence of three operations, named mutation, crossover, and selection, defined as follows.

A. Mutation

During mutation, a mutated or *donor* vector $\mathbf{v}_i$ is created as a linear combination of a few population members. Several mutation rules were proposed in the original paper by Storn and Price [7] in order to attain either a wider exploration of the search space or a faster convergence to the optimum (i.e., exploitation). Since then, many other rules have been devised, with the same purpose, by a number of researchers [16]. In the current version of *EOS*, four strategies, among those available in the literature, have been adopted:

$$\begin{aligned}
1) \quad & \mathbf{v}_i = \mathbf{x}_{r_1} + F \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3} \right) \\
2) \quad & \mathbf{v}_i = \mathbf{x}_{best} + F \left(\mathbf{x}_{r_1} - \mathbf{x}_{r_2} \right) \\
3) \quad & \mathbf{v}_i = \mathbf{x}_i + F \left(\mathbf{x}_{r_3} - \mathbf{x}_i \right) + F \left(\mathbf{x}_{r_1} - \mathbf{x}_{r_2} \right) \\
4) \quad & \mathbf{v}_i = \mathbf{x}_{best} + F \left(\mathbf{x}_{r_1} - \mathbf{x}_{r_2} \right) + F \left(\mathbf{x}_{r_3} - \mathbf{x}_{r_4} \right)
\end{aligned} \tag{8}$$

where $F \in \mathbb{R}$ is a parameter driving the mutation (*scale factor*), $\mathbf{x}_{best}$ is the best individual in the current population, and $r_1, \ldots, r_4$ represent randomly chosen, non-repeated, indexes belonging to $[1, N_p] \setminus \{i\}$. Each strategy has its own weaknesses and strengths: strategies based on the mutation of the best individual, such as strategies 2 and 4 (referred to as *DE/best/1* and *DE/best/2*, respectively) typically show a faster rate of convergence toward an (often local) minimum, whereas strategies based on randomly chosen individuals, such as strategies 1 and 3 (referred to as *DE/rand/1* and *DE/current-to-rand/1*, respectively) explore to a greater extent (yet more slowly) the whole search space.

B. Crossover

During crossover, a *trial* vector $\mathbf{u}_i$ is obtained by mixing the components of the agent vector $\mathbf{x}_i$ and the donor vector $\mathbf{v}_i$. By relying on empirical comparisons [17], the *binomial* crossover was preferred to the *exponential* one in *EOS*, and implemented as:

$$u_i^{(j)} = \begin{cases} v_i^{(j)} & \text{if } p_i^{(j)} \leq C_r \text{ or } j = j_r \\ x_i^{(j)} & \text{otherwise} \end{cases} \qquad (9)$$

for $j = 1, \ldots, D$, where $p_i^{(j)}$ is a random number with uniform distribution in $[0, 1]$, $C_r \in [0, 1]$ is another control parameter of the algorithm (named *crossover probability*) and j_r is a random index chosen once per individual in the range $[1, D]$ that ensures that at least one element in $\mathbf{u}_i$ is inherited from $\mathbf{v}_i$. At this point, the bounding intervals are enforced on the design variables:

$$u_i^{(j)} = \begin{cases} x_L^{(j)} & \text{if } u_i^{(j)} < x_L^{(j)} \\ x_U^{(j)} & \text{if } u_i^{(j)} > x_U^{(j)} \\ u_i^{(j)} & \text{otherwise} \end{cases} \qquad (10)$$

for $j = 1, \ldots, D$.

C. Selection

Eventually, the agent and trial vectors are compared. In the case of an unconstrained optimization problem, the one that is characterized by the best fitness value is retained and inserted in the new population pop^{G+1}:

$$\mathbf{x}_i^{G+1} = \begin{cases} \mathbf{u}_i^G & \text{if } f(\mathbf{u}_i^G) \leq f(\mathbf{x}_i^G) \\ \mathbf{x}_i^G & \text{otherwise} \end{cases} \qquad (11)$$

The corresponding selection step in the case of constrained optimization is discussed in Sec. VII.

D. Termination Criteria

This three-step process is repeated iteratively, creating at each generation a new population that replaces the previous one. Several termination criteria can be adopted, either alone or in conjunction, to stop the optimization procedure. The most common are:

- maximum number of fitness evaluations (FES), N_F;
- maximum number of generations, N_G;
- maximum number of consecutive generations without any improvement of the best solution found, $N_{G,best}$.

In typical *EOS* applications, the termination criterion is mainly based on the maximum number of generations N_G. This parameter strongly depends on the complexity of the analyzed problem, and it is generally selected in such a way that the outcomes of independent runs of the algorithm bring, in almost all cases, to similar results. So, a few preliminary runs of the code are necessary to identify a suitable value for N_G.

IV. SELF-ADAPTATION OF CONTROL PARAMETERS

A common practical issue for many stochastic algorithms concerns the selection of suitable values for the control parameters, i.e., the numerical quantities that drive the different phases of the optimization process. A fine tuning of the parameters is often mandatory in order to maximize the algorithm performance while solving a given problem.

The basic version of DE is characterized by only three control parameters. Apart from the population size N_p, the performance of the DE algorithm depends on an appropriate selection of the scale factor F, which controls the mutation phase, and the crossover probability C_r, which controls the crossover phase. However, as for other meta-heuristics, the best tuning of the control parameters, in terms of effectiveness and robustness of the algorithm, is usually related to the structure of the problem at hand [18].

In order to avoid a manual trial-and-error tuning of the DE control parameters prior to the solution of any problem, the jDE self-adaptive scheme proposed by Brest et al. [19] is implemented in *EOS* for automatically adjusting the values of both F and C_r during the optimization, without introducing any significant computational burden. In jDE, each individual owns its private copy of F and C_r, which are randomly initialized within the intervals $[F_{min}, F_{max}]$ and $[C_{r,min}, C_{r,max}]$, and thus different from individual to individual. Therefore, each individual evolves according to its own set of parameters. The hope is that good values of these control parameters would contribute to produce better individuals, which, being more prone to survive and produce offspring, will propagate their parameters into the population on following generations.

At the end of the current generation G, each individual $\mathbf{x}_i$ undergoes, with a probability p_τ, a random mutation of its control parameters:

$$F_i^{G+1} = \begin{cases} F_{min} + p_{i,1}\Delta F & \text{if } p_{i,2} \leq p_\tau \\ F_i^G & \text{otherwise} \end{cases}$$

$$(12)$$

$$C_{r,i}^{G+1} = \begin{cases} C_{r,min} + p_{i,3}\Delta C_r & \text{if } p_{i,4} \leq p_\tau \\ C_{r,i}^G & \text{otherwise} \end{cases}$$

where $p_{i,1}, \ldots, p_{i,4}$ are random numbers sampled from a uniform distribution in $[0, 1]$, $\Delta F = F_{max} - F_{min}$ and $\Delta C_r = C_{r,max} - C_{r,min}$. The hyper-parameter values suggested in [19] are used in *EOS*: $F_{min} = 0.1$, $F_{max} = 1$, $C_{r,min} = 0$, $C_{r,max} = 1$ and $p_\tau = 0.1$.

The population size N_p, instead, is kept fixed along generations. Its value is predetermined by looking at the problem dimension; a reasonable value for N_p generally lies between $5D$ and $10D$. For high-dimensional problems, e.g., $D > 100$, N_p is more often selected on the basis of the available computational budget.

V. EPIDEMIC MECHANISM

A partial restart mechanism, named "epidemic", is adopted in *EOS* in order to promote diversity among the individuals of the population over the generations. In fact, the use of (guided)

restart procedures in DE has demonstrated to be an effective method to reduce the chances of stagnation in isolated local minima [20], [21].

For this purpose, at each generation the population diversity score $\bar{d}$ is evaluated as the average Euclidean distance between any pair of solutions in the current population:

$$\bar{d} = \frac{2}{N_p(N_p - 1)} \sum_{i=1}^{N_p} \sum_{\substack{j=1 \\ j \neq i}}^{i} d_{ij} \tag{13}$$

where:

$$d_{ij} = \sqrt{\sum_{k=1}^{D} \left(\frac{x_i^{(k)} - x_j^{(k)}}{x_U^{(k)} - x_L^{(k)}} \right)^2} \tag{14}$$

Thus, d_{ij} would represent the actual (Euclidean) distance between the individuals $\mathbf{x}_i$ and $\mathbf{x}_j$ if the solution space were a D-dimensional unitary hypercube. For this reason, $d_{ij} \in [0, \sqrt{D}]$.

When the diversity score $\bar{d}$ falls under a given small threshold d_{tol}, an epidemic unleashes on the population. The best $\rho_{elite} N_p$ individuals are immune to the epidemic; instead, a large portion ρ_{ill} of the remaining individuals, randomly selected, contracts the "fatal disease", that is, is randomly reinitialized in the whole search space. This mechanism is illustrated in Fig. 1. The epidemic cannot happen twice within a number $N_{G,epid}$ of consecutive generations, so as not to compromise the search. Reasonable values for the newly introduced hyper-parameters can be chosen in the following ranges: $N_{G,epid} = 500 \div 2000$, $d_{tol} = 10^{-4} \div 10^{-2}$, $\rho_{elite} = 0.05 \div 0.25$, $\rho_{ill} = 0.75 \div 1$.
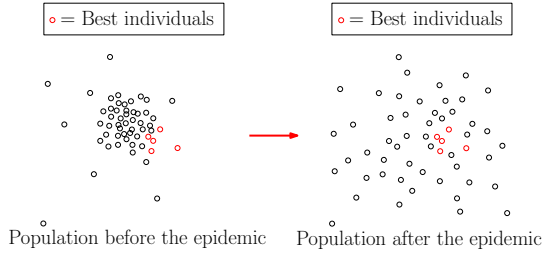

Fig. 1: Effect of the epidemic mechanism on the population.

The epidemic mechanism has proven to be particularly effective when an exploitative mutation rule (e.g., strategy 2 or 4) is chosen for DE. Figure 2 shows the effectiveness of the mechanism on two standard benchmark functions, that is, Rosenbrock function, with $D = 100$ and bounding intervals $[-50, 50]^D$, and Rastrigin function, with $D = 30$ and bounding intervals $[-5.12, 5.12]^D$. The plot reports the change in fitness trend (averaged on 20 independent runs) obtained when the epidemic mechanism is added to a single-population self-adaptive DE algorithm, where a self-adaption rule analogous to those in Eq. 12 was devised also to decide the mutation strategy among the four ones reported in Sec. III-A. The results were obtained with the following algorithm settings: $N_p = 64$, $N_G = 20000$, $N_{G,epid} = 1000$, $d_{tol} = 10^{-3}$, $\rho_{elite} = 0.1$, $\rho_{ill} = 1$.
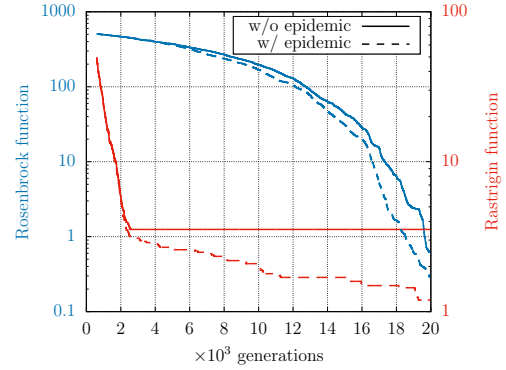

Fig. 2: Evolution plots, averaged on 20 independent runs, obtained with and without the epidemic mechanism on two benchmark functions.

VI. SPACE PRUNING BY CLUSTERING

The term "space pruning" refers to those techniques that aim at reducing the solution space during the search, focusing the optimization in a smaller area where, in accordance with some criteria, good solutions are expected to be found.

Different solution methods have been proposed in the past [22], [23] that combine branching, that is, the subdivision of the feasible region into smaller subdomains (with some convergence properties) and population-based stochastic algorithms, to search in the subdomains in order to evaluate them, showing higher average performance with respect to a number of stochastic and deterministic methods on the same problems.

The pruning method adopted in *EOS* is based on the *cluster pruning algorithm* developed by ESA's Advanced Concept Team [24] to solve Multiple Gravity-Assist (MGA) problems. The key idea is that in problems like MGA and related space trajectory optimization problems (as MGA-1DSM or multi-rendezvous ones), which are the kind of problems addressed by *EOS*, good solutions are often clustered in small regions instead of being densely distributed over the whole search space. So, during the optimization process, it is possible to progressively focus the search around the promising regions identified so far, increasing the quality of the solutions found, but accepting the risk that some possibly better solution may be potentially ruled out.

The *pruning-by-clustering* method implemented in *EOS* makes use of N_r independent, separate, partial runs. Each partial run corresponds to a run of the same DE algorithm with a (different) random initial population.

At each pre-assigned generation $N_{G,pr}^i$, the i-th pruning event occurs. For each separate run, the best found solution is collected in the set $pop^{best} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_r}\}$, which is sorted according to their fitness, so that $f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \ldots \leq f(\mathbf{x}_{N_r})$. A new (smaller) search space Ω_{pr}^i is defined as the convex hull containing the best $N_{p,pr}^i = \lfloor \rho_{pr}^i N_r \rfloor$ solutions in pop^{best}, up to some relaxation factor related to ρ_{pr}^i. More

precisely, for each design variable $x^{(j)}$, with $j \in [1, D]$, the new bounding interval $\left[x^{(j)}_{L,new}, x^{(j)}_{U,new} \right]$ is defined as:

$$x^{(j)}_{L,new} = \min_{k \in [1, N^i_{p,pr}]} x^{(j)}_k - 0.5(1 - \rho^i_{pr})\left(x^{(j)}_U - x^{(j)}_L \right) \quad (15)$$

$$x^{(j)}_{U,new} = \max_{k \in [1, N^i_{p,pr}]} x^{(j)}_k + 0.5(1 - \rho^i_{pr})\left(x^{(j)}_U - x^{(j)}_L \right) \quad (16)$$

where $\rho^i_{pr} = \rho^0_{pr} - i\Delta\rho_{pr}$ is a tuning parameter that controls the extent of the search space region to be removed. After the pruning event, the N_r independent, partial optimization runs are restarted, randomly initializing the population over the pruned search space Ω^i_{pr}. Nevertheless, the $N^i_{p,pr}$ best individuals of pop^{best} are copied into each new population, substituting the (randomly generated) worst individuals.

If N_{pr} pruning events are enforced, starting from generation $N^0_{G,pr}$, the subsequent pruning events are spaced from each other by $(N_G - N^0_{G,pr})/N_{pr}$ generations. By choosing $N^i_{p,pr} = \lfloor \rho^i_{pr} N_p \rfloor$, one makes the pruned search space iteratively smaller, as the convex hull is created on less points.

The hyper-parameter values that showed the best performance in EOS are the following: $\rho^0_{pr} = 0.3$, $\Delta\rho_{pr} = 0.1$, $N_{pr} = 3$ and $N^0_{G,pr} = 0.4N_G$.

VII. Constraint Handling

In order to tackle the constrained optimization problem defined by Eqs. (2) and (4) with DE, it is necessary to modify in some way the selection step (11) to take into account the presence of inequality constraints.

The simple, but promising, min-max approach proposed by Jimenez and Verdegay [25] is implemented in *EOS* to handle constraints. The key idea is to adopt a lexicographic order in the selection process, in which the constraint violation precedes the objective function:

1) between two feasible individuals, select on the basis of the minimum value of the cost function;
2) between a feasible and an infeasible individual, select the feasible one;
3) between two infeasible individuals, select on the basis of the lowest maximum constraint violation:

$$\Psi_{max} = \max_{j \in [1, K]} \Psi^{(j)}(\mathbf{x}) \quad (17)$$

However, as highlighted by Coello [12], this simple approach pushes the search to focus first only on the constraint satisfaction; so, if the feasible solution space is disjoint, the search could get trapped in a part of the feasible region that, in general, could be quite far from the global minimum of the problem, and, from which, it is impossible to escape.

An efficient way to overcome such drawback has been proposed by Takahama and Sakai [26], who applied an ε-constrained method to Differential Evolution ("εDE"). They suggested introducing a tolerance ε for constraint violation

Ψ_{max}, to be decreased along generations. The following rule has been adopted in *EOS* for evaluating ε at generation G:

$$\varepsilon^G = \begin{cases} \varepsilon^0 & \text{for } G \leq N^0 \\ \varepsilon^0 \left[\frac{\varepsilon^\infty}{\varepsilon^0} \right]^{\frac{G - N^0}{N^\infty - N^0}} & \text{for } N^0 < G < N^\infty \\ \varepsilon^\infty & \text{for } G \geq N^\infty \end{cases} \quad (18)$$

with $\varepsilon^0, \varepsilon^\infty$ the initial and final values of the tolerance. As for N^0 and N^∞, which define the interval in which ε decreases, the following values showed the best overall results with *EOS*: $N^0 = \frac{N_G}{6}$, $N^\infty = N_G$.

By using Eq. (18), moves toward infeasible solutions are allowed at the beginning of the search, when the tolerance ε is maximum and the entire solution space must be explored in order to identify promising regions. As the generation number increases, such moves become forbidden, and the search concentrates in the feasible part of the identified region, which, thanks to the initial search space exploration, is more likely to be closer to the optimum of the problem.

VIII. Island-Model

During the search process, a proper balance between two opposite needs, namely "Exploitation" and "Exploration", is paramount to the performance of any EA [15]. Here exploitation refers to the capability of the evolutionary algorithm to exploit the information already collected from the population to focus the search toward the goal; exploration, instead, refers to the ability to introduce new information about the solution space into the population. In DE, different mutation strategies, see Eq. (8) may privilege, to a certain degree, one tendency over the other [27]. It goes without saying that certain mutation strategies perform better on some optimization problems than others, but the opposite may be true for different problems. This leads to the idea of combining different strategies together within a single search process, in order to obtain a more robust and performing algorithm, capable of successfully tackling a wider range of problems [28]. In the same fashion as described in Sec. IV for the DE control parameters, a self-adaptation of the mutation strategy could be devised [29]; unfortunately, this approach suffers from the fact that greedy (i.e., exploitative) strategies tend to prevail over the others in just a few generations.

EOS, instead, adopts a synchronous island-model paradigm as a way to concurrently handle different mutation strategies within one optimization run. An island-based EA relies on the definition of several sub-populations, or "tribes", each one evolving independently of the others, according to its own (preassigned) algorithm. Each tribe lives on a separate "island", and all the islands are arranged in a cluster, or "archipelago", of arbitrary topology. Information sharing among the islands occurs only in sporadic events called "migrations", during which the best individuals move from an island to the neighboring ones, in accordance with the predefined archipelago topology [30]. Numerical tests of multi-population DE-based algorithms on a broad range of low-to-high dimensional optimization problems [31], [32] showed

an improvement in performance with respect to sequential versions of the algorithm.

The island-model paradigm can be exploited to combine the convergence and search properties of different DE algorithms. The idea is to use heterogeneous mutation strategies among different islands, so as to achieve a correct balance between the search space exploration and exploitation, usually performing better than the best of the strategies involved in that particular problem [33]. As an example, in the radially-arranged 16-island archipelago in Fig. 3, inner rings favor exploration, featuring mutation strategies 1 and 3, while outer rings favor exploitation, featuring strategies 1 and 4.
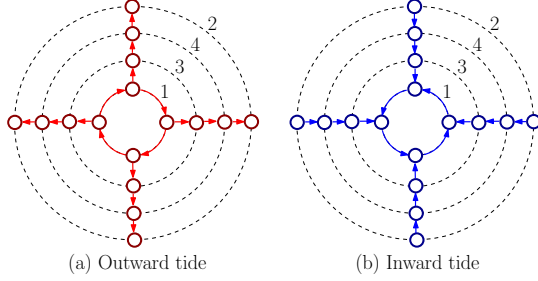

Fig. 3: Migration tide: forward (a) and backward (b), for the 16-island case.

The island-model optimization process, with its alternating phases of extended, isolated computation and occasional communications, also allows for an easy parallelization on a message-passing multi-processor environment. The MPI message-passing standard [34] is exploited in *EOS* for this purpose: each of the N_i islands corresponds to a process, and is assigned to a different node/CPU of a cluster. The evolution phase proceeds in parallel, until communications between processes are performed during migrations. More precisely, a synchronous migration of the best $N_b = \rho_{mig} N_p$ individuals occurs between the connected islands every N_{mig} generations, with a probability ϕ_{mig}. The N_b best individuals are copied in the destination islands, where they replace the N_b worst individuals. Typical values of these parameters are: $\rho_{mig} = 0.05 \div 0.1$, $N_{mig} = 100$, $\phi_{mig} = 0.5 \div 1$. Communications are easily handled through the point-to-point send/receive functions of the MPI library. In addition, MPI allows for handy implementation of Cartesian or graph process topologies, which define the "neighbors" of any process to/from which it can send/receive information; by exploiting such capability, it is possible to arrange the archipelago in any topology in a straightforward manner. Besides classical topologies, such as the ring topology or the Von Neumann grid topology [35], *EOS* implements as a default a peculiar radial topology, where migration tides alternate their direction at each event, as presented in Fig. 3.

This coarse-grained island-model, in which each tribe is mapped to a CPU, can be combined with a fine-grained parallelism, in which each individual (or group of individuals) of the tribe is assigned to a different core of the CPU [36]. The fine-grained parallelization of each sub-population is realized in *EOS* through OpenMP, which is a set of compiler directives and library routines used to perform shared-memory parallelism, e.g., between the cores of a single, multi-core CPU; so, if n_c cores are available for each CPU of the cluster, N_p/n_c individuals are mapped to each core through an OpenMP directive.

IX. APPLICATION TO REAL-WORLD SPACE TRAJECTORY OPTIMIZATION PROBLEMS

This section presents the main results attained by *EOS* when applied to challenging, real-world space trajectory optimization problems. More detailed information about how the three optimization problems that follow have been formulated (that is, their objective function, design variables, bounding boxes and constraints) are reported in the corresponding references [2], [3], [6].

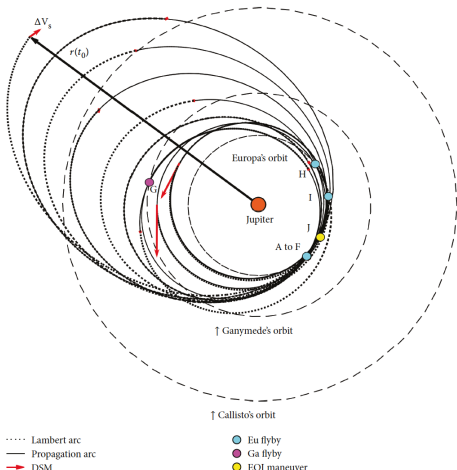### A. Europa Tomography Probe Mission Design

In 2015, a scientific and engineering team at Sapienza University of Rome, in collaboration with the Imperial College of London, carried out a feasibility study for a probe that could be launched as a piggyback payload for the NASA Europa Clipper mission, to enhance its scientific return [37]. An innovative mission concept was proposed, where a small Europa's orbiter, named Europa Tomography Probe (ETP), hosting just one magnetometer and a transponder required for the Inter-Satellite Link (ISL) with the mother spacecraft, is proved to be capable of providing crucial information on the moon's interior structure, such as depth and conductivity of the subsurface ocean. Also, ISL supports the reconstruction of the mother spacecraft orbit, significantly improving the accuracy of the topographic reconstruction of Europa's surface [38].

The optimization of ETP capture trajectory was crucial for the validity of the overall proposal, as a tight requirement on the available propellant mass was enforced by the need to stay within a total probe mass specified by NASA. A mission strategy based on the v_∞ leveraging concept [39] and the use of resonant orbits to exploit multiple gravity-assist from Europa was thus proposed [2]. Under the assumption of a patched-conic model, with radius of the sphere of influence of the secondary bodies and travel time inside these regions negligible, and an impulsive-thrust model, the problem can be posed as an unconstrained optimization problem, also known as Multiple Gravity-Assist with One Deep Space Maneuver (MGA-1DSM) [40], where the objective is to minimize the overall ΔV, which is the cumulative velocity variation performed by means of the on-board propulsive system.

A velocity formulation [41] of the problem was exploited: the overall capture trajectory is made up of a series of body-to-body legs, each starting with a flyby and composed of two ballistic arcs, a *propagation arc* and a *Lambert arc*, joined by an impulsive maneuver. The k-th leg of the trajectory is parameterized by using four parameters: $\{r_{\pi,k}, \beta_k, \Delta T_k, \eta_k\}$ which represent, respectively, the flyby radius, the flyby plane orientation, the leg flight-time, and the fraction of the leg flight-time at which DSM occurs. By properly selecting the

bounds for variables ΔT_k and η_k it is possible to enforce a given resonance of the probe with a Jovian moon on the k-th leg. An initial Lambert arc, which moves the probe from the assigned initial condition (Jovian orbit in 4:1 resonance with Europa) to the first encounter with Europa, completes the formulation. This initial leg is described by three variables: the probe release epoch t_0, flight time ΔT_0 and flight angle $\Delta\theta$.

Fig. 4: Best fitness trend obtained with and without the use of the pruning procedure.

MGA-1DSM problems are typically characterized by a huge number of local optima [42]; in this respect, either the pruning-by-clustering method (Sec. VI) or the island-model (Sec. VIII) are paramount to the success of the optimization procedure through *EOS*. In particular, Figure 4 compares the results obtained on Mission C of Ref. [2] by performing 50 runs of a mono-population, self-adaptive DE algorithm with and without the pruning procedure. The evolution plots reported are obtained by considering, at any generation number, the best fitness among those found by the separate runs. The following algorithm settings were used: $N_i = 1$, $N_p = 64$, $N_G = 10000$, $N_r = 50$, $\rho_{pr}^0 = 0.3$, $\Delta\rho_{pr} = 0.1$, $N_{pr} = 3$ and $N_{G,pr}^0 = 0.4N_G$.

Fig. 5: ETP capture trajectory.

The overall-best capture trajectory obtained for ETP is reported in Fig. 5, and allows saving approximately $1900\,\mathrm{m\,s^{-1}}$ of ΔV with respect to a direct insertion maneuver. The trajectory exploits eight flybys of Europa and a flyby of Ganymede, for a total of 39 design variables. The result was obtained with the following algorithm settings: $N_i = 8$, $N_p = 512$, $N_G = 10000$.

B. Ascent Trajectory Optimization of VEGA Launch Vehicles

Since 2017, *EOS* is adopted as a solver for the optimization of the ascent trajectory of VEGA Launch Vehicle (LV) and its evolution VEGA-C in the framework of independent support and cross-check activity for ESA-ESRIN [43]. By applying a control discretization approach, the problem of optimizing the ascent trajectory of a LV from the launch pad to a target orbit can be posed as a global constrained optimization problem. This problem is highly sensitive to the value of the optimization variables and often the trajectory cannot be evaluated over the whole search space. Therefore, the use of a derivative-free optimization algorithm is mandatory. With respect to the previous application, here the constraint handling plays a major role in determining the success or failure of the optimization process. In particular, both the terminal constraints on the final LV orbital parameters and the path constraints limiting the dynamic pressure, the bending and axial loads experienced by the rocket, and the heat flux on the payload during the flight after the fairing jettisoning must be considered. As a result, the admissible portion of the search space is quite small. Traditional approaches of constraint handling for EAs based on penalty functions [44] are not very effective in this case, because they tend to create, and get the search trapped into, a number of spurious sub-optimal solutions. Moreover, they are strongly reliant on a proper choice of the constraint weighting factors. Barrier methods are not of any help too, as the feasible set in the search domain is quite limited and disconnected.

On the other hand, the ε-constrained method adopted in *EOS* is really effective on this kind of problems, since starting from a value of the tolerance ε sufficiently high, ε-feasible solutions (i.e., which meet the imposed constraints with a tolerance ε) can be obtained very soon during the search. Then, by decreasing in a slow and monotone way the parameter ε, the EA is able to maintain the solution feasible and, at the same time, attain better values of the fitness. As a result, at the end of the search, when ε is close to zero, a feasible, good quality solution is returned by the algorithm. This process is shown in Fig. 6 for VEGA LV, by using as objective function (to maximize) the payload mass M_u.

A typical run for a low/medium-fidelity ascent trajectory optimization with 22 variables and 9 nonlinear inequality constraints ($N_i = 1$, $N_p = 128$, $N_G = 3000$) requires about 5 minutes on a workstation with Intel Core i9-9900K @3.60 GHz, using a single MPI process and up to 16 OpenMP threads.

An integrated optimization of the ascent trajectory and first stage Solid Rocket Motor (SRM) design of a VEGA-like
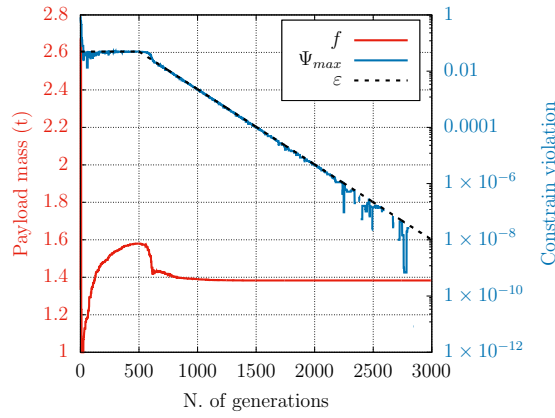
Fig. 6: Typical trend of fitness f, constraint violation Ψ_{max} and ε level during a run of *EOS* for an ascent trajectory optimization problem.

multi-stage launch vehicle was also carried out through *EOS*, where the objective was to determine the optimal internal pressure law during the first stage SRM operation, together with the optimal thrust direction and other relevant flight parameters of the entire ascent trajectory, so as to maximize the payload injected into a target orbit. Multiple design constraints involving the solid rocket motor or dependent on the actual flight trajectory were also enforced. Here the problem was even more complex, because of the large number of constraints and its multi-disciplinarity (the design variables represent very distant physical quantities); nevertheless, *EOS* was still able to attain competitive results [3].

In this case, a typical high-fidelity optimization ($N_i = 8$, $N_p = 136$, $N_G = 10000$), with 22 variables and 12 nonlinear inequality constraints, has a run time of about 30 minutes on KNL partition of CINECA's supercomputer Marconi (3600 compute nodes with 68-core Intel(R) Knights Landing @1.40GHz and 16 GBMCDRAM + 96 GB RAM each, connected through an IntelOmniPath network @100Gb/s), by using 8 nodes and 68 CPU cores per node.

C. Active Debris Removal Missions

An Active Debris Removal (ADR) mission can be seen as a peculiar instance of a multi-rendezvous (MRR) trajectory, where an active (chaser) spacecraft is asked to visit (that is, to perform a rendezvous with) a certain number of targets (space debris), making the best use of the on-board propellant.

The optimization of a multi-target rendezvous trajectory can be posed as a mixed-integer NLP problem, involving the simultaneous optimization of both integer variables (defining the debris encounter sequence) and real-valued variables (describing the spacecraft trajectory from a debris to the next one and the encounter epochs). A bi-level approach can be pursued for solving this NP-hard problem, by isolating i) an outer-level, which concerns the definition of the encounter sequence based on a rough, but fast, evaluation of the encounter epochs (an *heuristic*), and ii) an inner-level, which deals with the

optimization of each body-to-body transfer with full details, assuming that departure and arrival bodies are assigned; encounter epochs are also adjusted.

Given a proper heuristic to estimate the cost of each transfer leg (without solving the full optimization problem), the two levels can be solved sequentially: the outer-level combinatorial problem is isolated and solved first, for example using either a Genetic Algorithm [5] or Simulated Annealing [6]; its solution is then used as initial guess for the inner-level NLP problem, that is the actual trajectory optimization problem.

If a Keplerian dynamical model is assumed for both the targets and the chaser, the inner-level problem results in an unconstrained problem of real-valued design variables with the total ΔV as cost function, which can be tackled by *EOS*. When the N targets to remove are supposed on circular and coplanar orbits, each debris-to-debris trajectory can be parameterized by 7 variables, one being the final encounter time and the other 6 identifying the trajectory arcs making up the transfer. So, to optimize the whole chaser trajectory, *EOS* must deal with a $7N$-variable problem. Because of the high problem dimensions (for typical values of N) the island-model parallelization of *EOS* is decisive to attain good quality results is a limited amount of time.
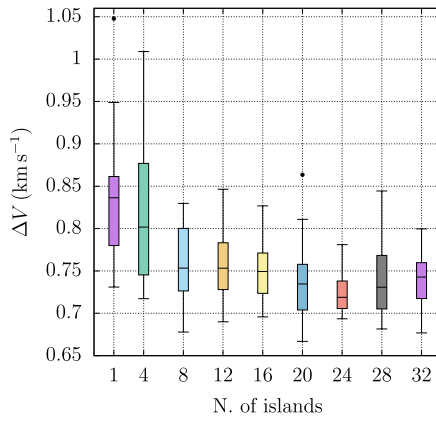
TABLE I: Results of the application of *EOS* to the 15-target ADR mission by varying the number of islands.

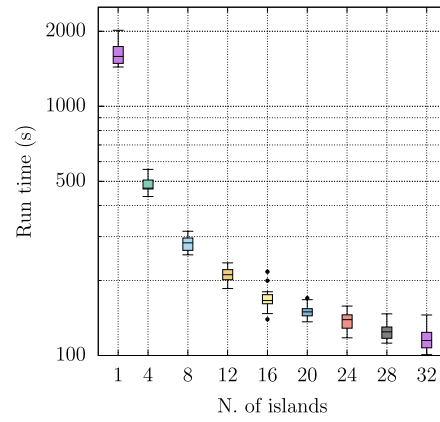| Algorithm parameters			Fitness stats, [km/s]		
N_i	N_p	N_G	*mean*	*std*	*best*
1	512	20000	0.8400	0.0755	0.7310
4	256	10000	0.8165	0.0842	0.7173
8	128	10000	0.7577	0.0444	0.6779
12	85	10000	0.7575	0.0408	0.6900
16	64	10000	0.7475	0.0343	0.6957
20	51	10000	0.7369	0.0458	0.6668
24	43	10000	0.7231	0.0228	0.6935
28	37	10000	0.7389	0.0437	0.6816
32	32	10000	0.7381	0.0318	0.6768

Fig. 7 shows a few results for a 15-debris ADR mission (105 variables) as a function of the number of islands N_i, always considering approximately the same maximum number of FES. Population size and number of generations are reported in Table I. A migration probability $\phi_{mig} = 0.5$ was used in these tests. For the sake of comparison, the results obtained through a single-island version of *EOS*, with self-adaptive mutation strategy, are also reported. All computations were performed on KNL partition of CINECA's supercomputer Marconi, whose characteristics are reported in Sec. IX-B. The reduction of the user run-time is apparent. The quality of the attained solutions improves with the number of islands, up to $N_i = 24$; from that point on, the population size of each island probably becomes too small for the problem at hand.

X. Conclusion

This paper presents the evolutionary optimization code *EOS* developed at Sapienza University of Rome, which represents a state-of-the-art DE-based algorithm. After recalling the traditional DE algorithm, the main features added to *EOS* to

(a) Fitness (total ΔV) vs number of islands.

(b) Run time (logarithmic scale) vs number of islands.

Fig. 7: Boxplots of run time and solution quality as a function of the number of islands N_i for a 15-target ADR mission. Statistics are averaged over 20 independent runs, with a maximum FES $N_F = 10240000$.

enhance the performance of DE were described in detail. These features include: the self-adaptation scheme for the DE control parameters, to avoid a manual tuning of such quantities and let them automatically evolve along the search; a partial restart ("epidemic") mechanism, to enhance population diversity and evade the danger to get the search trapped in local optima; a pruning technique based on clustering, to focus the search on the most promising regions of the solution space; a ε-constrained method to make the algorithm capable of efficiently tackling constrained problems; a parallel synchronous island-model paradigm, to properly balance the "explorative" and "exploitative" tendencies of the algorithm and, at the same time, being able to distribute the computational load on a highly parallel environment, with a reduction of computational times and an increase in performance as immediate results. *EOS* demonstrated to be successful when applied to hard, real-world (unconstrained and constrained) problems of space trajectory design, characterized by many variables and the presence of several local minima, such as multiple gravity-assist trajectories, the ascent trajectory of a rocket, and multi-target rendezvous missions.

REFERENCES

[1] L. Casalino, G. Colasurdo, A. Zavoli, and M. Berga, "Gtoc8: Results and methods of team 22," *Advances in the Astronautical Sciences*, vol. 158, pp. 4291–4300, 2016.

[2] L. Federici, A. Zavoli, and G. Colasurdo, "Preliminary capture trajectory design for europa tomography probe," *International Journal of Aerospace Engineering*, vol. 2018, 2018.

[3] L. Federici, A. Zavoli, G. Colasurdo, L. Mancini, and A. Neri, "Integrated optimization of ascent trajectory and srm design of multistage launch vehicles," *Advances in the Astronautical Sciences*, vol. 168, pp. 733–752, 2019.

[4] B. Benedikter, A. Zavoli, and G. Colasurdo, "A convex approach to rocket ascent trajectory optimization," in *8th European Conference for Aeronautics and Space Sciences (EUCASS)*, Madrid, Spain, 1-4 July 2019.

[5] L. Federici, A. Zavoli, and G. Colasurdo, "Impulsive multi-rendezvous trajectory design an optimization," in *8th European Conference for Aeronautics and Space Sciences (EUCASS)*, Madrid, Spain, 1-4 July 2019.

[6] L. Federici, Z. Alessandro, and G. Colasurdo, "A time-dependent TSP formulation for the design of an active debris removal mission using simulated annealing," in *Astrodynamics Specialist Conference*, vol. AAS 19-701, Portland, Maine, 11-15 Aug. 2019.

[7] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[8] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*, vol. 2. IEEE, 2004, pp. 1980–1987.

[9] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution–an updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 2016.

[10] M. Locatelli and M. Vasile, "(non) convergence results for the differential evolution method," *Optimization Letters*, vol. 9, no. 3, pp. 413–425, 2015.

[11] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE transactions on evolutionary computation*, vol. 15, no. 1, pp. 4–31, 2011.

[12] C. A. C. Coello and A. Carlos, "A survey of constraint handling techniques used with evolutionary algorithms," *Lania-RI-99-04, Laboratorio Nacional de Informática Avanzada*, 1999.

[13] Á. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on evolutionary computation*, vol. 3, no. 2, pp. 124–141, 1999.

[14] J. Lampinen, I. Zelinka *et al.*, "On stagnation of the differential evolution algorithm," in *Proceedings of MENDEL*, 2000, pp. 76–83.

[15] A. E. Eiben and C. A. Schippers, "On evolutionary exploration and exploitation," *Fundamenta Informaticae*, vol. 35, no. 1-4, pp. 35–50, 1998.

[16] M. Ali, "Differential evolution with generalized differentials," *Journal of Computational and Applied Mathematics*, vol. 235, no. 8, pp. 2205–2216, 2011.

[17] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM, 2006, pp. 485–492.

[18] J. Liu, "On setting the control parameter of the differential evolution method," in *Proceedings of the 8th international conference on soft computing (MENDEL 2002)*, 2002, pp. 11–18.

[19] J. Brest, B. Bošković, S. Greiner, V. Žumer, and M. S. Maučec, "Performance comparison of self-adaptive and adaptive differential evolution algorithms," *Soft Computing*, vol. 11, no. 7, pp. 617–629, 2007.

[20] F. Peng, K. Tang, G. Chen, and X. Yao, "Multi-start jade with knowledge transfer for numerical optimization," in *2009 IEEE Congress on Evolutionary Computation*. IEEE, 2009, pp. 1889–1895.

[21] M. Vasile, E. Minisci, and M. Locatelli, "An inflationary differential evolution algorithm for space trajectory optimization," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, pp. 267–281, 2011.

[22] M. Vasile and M. Locatelli, "A hybrid multiagent approach for global trajectory optimization," *Journal of Global Optimization*, vol. 44, no. 4, pp. 461–479, 2009.

[23] B. Ghojogh, S. Sharifian, and H. Mohammadzade, "Tree-based optimization: A meta-algorithm for metaheuristic optimization," *arXiv preprint arXiv:1809.09284*, 2018.

[24] B. A. Conway, *Spacecraft trajectory optimization*. Cambridge University Press, 2010, vol. 29.

[25] F. Jiménez and J. L. Verdegay, "Evolutionary techniques for constrained optimization problems," 1999.

[26] T. Takahama and S. Sakai, "Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 2010, pp. 1–9.

[27] V. Feoktistov, *Differential evolution*. Springer, 2006.

[28] M. G. Epitropakis, V. P. Plagianakos, and M. N. Vrahatis, "Balancing the exploration and exploitation capabilities of the differential evolution algorithm," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. IEEE, 2008, pp. 2686–2693.

[29] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2008.

[30] W. Martin, J. Lienig, and J. P. Cohoon, "C6. 3 island (migration) models: evolutionary algorithms based on punctuated equilibria," *B ack et al. BFM97], Seiten C*, vol. 6, pp. 101–124, 1997.

[31] D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, "Parallel differential evolution," in *Proceedings of the 2004 congress on evolutionary computation (IEEE Cat. No. 04TH8753)*, vol. 2. IEEE, 2004, pp. 2023–2029.

[32] M. Di Carlo, M. Vasile, and E. Minisci, "Adaptive multi-population inflationary differential evolution," *Soft Computing*, pp. 1–31, 2019.

[33] D. Izzo, M. Rucinski, and C. Ampatzis, "Parallel global optimisation meta-heuristics using an asynchronous island-model," in *2009 IEEE Congress on Evolutionary Computation*. IEEE, 2009, pp. 2301–2308.

[34] L. Clarke, I. Glendinning, and R. Hempel, "The mpi message passing interface standard," in *Programming environments for massively parallel distributed systems*. Springer, 1994, pp. 213–218.

[35] N. Lynn, M. Z. Ali, and P. N. Suganthan, "Population topologies for particle swarm optimization and differential evolution," *Swarm and evolutionary computation*, vol. 39, pp. 24–35, 2018.

[36] E. Alba and M. Tomassini, "Parallelism and evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 443–462, Oct 2002.

[37] V. Notaro, M. Di Benedetto, G. Colasurdo, D. Durante, P. Gaudenzi, L. Imperi, M. Mariani, A. Marotta, G. Palermo, L. Pollice, P. Racioppa, A. Zavoli, and L. Iess, "Europa tomography probe (ETP) mission feasibility - spacecraft design," 2016.

[38] M. Di Benedetto, L. Imperi, D. Durante, M. Dougherty, L. Iess, V. Notaro, and P. Racioppa, "Augmenting nasa europa clipper by a small probe: Europa tomography probe (etp) mission concept," *Acta Astronautica*, vol. 165, pp. 211–218, 2019.

[39] J. A. Sims and J. M. Longuski, "Analysis of v_∞ leveraging for interplanetary missions," in *AIAA/AAS Astrodynamics Conference, Scottsdale, AZ*, 1994, pp. 505–513.

[40] M. Vasile and P. D. Pascale, "Preliminary design of multiple gravity-assist trajectories," *Journal of Spacecraft and Rockets*, vol. 43, no. 4, pp. 794–805, 2006.

[41] M. Ceriotti, "Global optimisation of multiple gravity assist trajectories," Ph.D. dissertation, 2010.

[42] D. Myatt, V. M. Becerra, S. J. Nasuto, and J. Bishop, "Advanced global optimisation for mission analysis and design," *Final Report. Ariadna id*, vol. 3, p. 4101, 2004.

[43] *Statement of Work for Support and Cross-check of VEGA LEAP and VEGA-C Solid Propulsion and System Activities (Period Q2 2015 - Q2 2017) Addendum, VG-SOW-0-D-30005-ESA*.

[44] J. T. Richardson, M. R. Palmer, G. E. Liepins, and M. R. Hilliard, "Some guidelines for genetic algorithms with penalty functions," in *Proceedings of the 3rd international conference on genetic algorithms*. Morgan Kaufmann Publishers Inc., 1989, pp. 191–197.