MHACO: a Multi-Objective Hypervolume-Based Ant Colony Optimizer for Space Trajectory Optimization

1st Giacomo Acciarini
Department of Mechanical and Aerospace Engineering
University of Strathclyde
75 Montrose Street, G1 1XJ Glasgow, United Kingdom
giacomo.acciarini@strath.ac.uk

2nd Dario Izzo
Advanced Concepts Team
European Space Research and Technology Center
Keplerlaan 1, 2201 AZ, Noordwijk, The Netherlands
dario.izzo@esa.int

3rd Erwin Mooij
Faculty of Aerospace Engineering
Delft University of Technology
Kluyverweg 1, 2629 HS Delft, The Netherlands
e.mooij@tudelft.nl

***Abstract*—In this paper, we combine the concepts of hypervolume, ant colony optimization and nondominated sorting to develop a novel multi-objective ant colony optimizer for global space trajectory optimization. In particular, this algorithm is first tested on three space trajectory bi-objective test problems: an Earth-Mars transfer, an Earth-Venus transfer and a bi-objective version of the Jupiter Icy Moons Explorer mission (the first large-class mission of the European Space Agency's Cosmic Vision 2015-2025 programme). Finally, the algorithm is applied to a four-objectives low-thrust problem that describes the journey of a solar sail towards a polar orbit around the Sun. The results on both the test cases and the more complex problem are reported by comparing the novel algorithm performances with those of two popular multi-objective optimizers (i.e., a nondominated sorting genetic algorithm and a multi-objective evolutionary algorithm with decomposition) in terms of hypervolume metric. The numerical results of this study show that the multi-objective hypervolume-based ant colony optimization algorithm is not only competitive with the standard multi-objective algorithms when applied to the space trajectory test cases, but it can also provide better Pareto fronts in terms of hypervolume values when applied to the complex solar sailing mission.**

***Index Terms*—global optimization, space trajectory optimization, ant colony optimization, hypervolume metric**

I. Introduction

Global optimization techniques are usually critical in terms of computation time, however, thanks to the fast developments in computer power in recent years, the interest in global optimization methods have been constantly increasing. Generally, two different kinds of global optimization algorithms are distinguished: deterministic and stochastic. Deterministic optimization focuses on providing theoretical proofs that the found solution is indeed the global best one, within a defined tolerance interval. Deterministic methods do not include any randomness, and they typically guarantee a finite amount of work for reaching the solution within certain tolerances. However, they also require physical insight into the problem, which can allow to formulate them rigorously from a mathematical point of view.

On the other hand, stochastic methods use randomly generated variables, and they basically provide optimization methods without the need for any particular insight into the problem. This means that the optimization problem can be treated as a black box. Furthermore, most of the stochastic methods are heuristic (i.e., it cannot be proved that they can find a global optimal solution within a certain amount of time). These heuristic methods seem very suitable for trajectory optimization problems, where the domain is typically too complex and large to be represented in a closed mathematical form or fully explored. Some of the most important stochastic algorithms are evolutionary algorithms (EA) (among which we distinguish: genetic algorithms (GA), differential evolution (DE), and many other forms), ant colony optimization (ACO), and particle swarm optimization (PSO). Also, many other variants of these algorithms have been derived (e.g., extended ant colony optimization (EACO), self-adaptive differential evolution (SADE), etc.). The European Space Agency (ESA), during the first years of the 21st century, has extensively studied GA, PSO, MPSO, DE, and ASA, to find the best algorithms for trajectory optimization using certain benchmark problems. It has been shown in those years (see for instance [1] and [2]) that MPSO, DE, ASA seem to be most promising for trajectory optimization. However, recently it has also been suggested that ACO might even outperform these algorithms for those types problems [3], [4], [5]. However, the extended ant colony optimizer that currently shows such performance is not available in an open-source fashion. Moreover, its multi-objective (MO) extension makes use of a Utopia-Nadir

balance decomposition method [6], whose performance is highly dependent on the available information about the nadir and utopia points for each objective function, and on the established weights for constructing the target functions. For overcoming these difficulties, we propose an extended ant colony optimizer whose evolutionary strategy is inspired by [3], but it implements a new technique for handling multi-objective problems by using the hypervolume metric and the nondominated sorting strategy for assessing the quality of the solutions. To sum up, on the one hand, such algorithm is intended to exploit the suggested good performances of single-objective ACO on trajectory optimization types of problems, on the other, it is meant to retain these capabilities to MO problems, thanks to the fact that the hypervolume metric combines both the diversity and convergence information in one single value. Furthermore, we make this algorithm available in *pagmo2* [7]: the parallel global optimization software developed and maintained by the European Space Agency, so that everybody can access and improve the algorithm or test its performances, in an open-source fashion[1]. The algorithm can handle mixed-integer variables as well as concave, convex and discontinuous Pareto fronts as verified from several tests made on the ZDT, DTLZ and WFG test suites (which are also reported as test cases in the software implementation of the algorithm).

II. Methods

A. Multi-Objective Optimization

Assuming that there are M objective functions, the problem can be mathematically formulated as follows:

$$\min_{\mathbf{x}\in\Sigma}(\mathbf{f}(\mathbf{x})) = \min_{\mathbf{x}\in\Sigma}[f_1(\mathbf{x}), ..., f_M(\mathbf{x})]^T \tag{1}$$

where $\mathbf{x}$ is a vector of N variables, which can be both integer and continuous, defined in the search space Σ. The problem is generally subjected to equality and inequality constraints, which can be expressed, respectively, as:

$$h_i(\mathbf{x}) = 0 \qquad i = 1, ..., m \tag{2}$$

$$g_i(\mathbf{x}) \leq 0 \qquad i = m+1, ..., p \tag{3}$$

For multi-objective optimization, the problem is thus to simultaneously minimize the components of a vector of functions ($\mathbf{f} = (f_1, ..., f_M)^T$), which is subject to several constraints. The problem is typically not uniquely solvable, but a set of equally efficient alternative solutions is possible. These solutions, among all the found ones, constitute the Pareto-optimal front, when they are plotted in the objective space. When comparing individuals i and j, with objective vector functions $\mathbf{f}_i$ and $\mathbf{f}_j$ we affirm that i dominates j in the Pareto sense if and only if:

$$\forall k \in \{1, .., M\} : f_{i,k} \leq f_{j,k},\ \exists k \in \{1, .., M\} : f_{i,k} \neq f_{j,k}$$

[1]https://esa.github.io/pagmo2/docs/cpp/algorithms/maco.html, date of access: January 2020.

The Pareto-optimal set is thus a set of individuals that has the best performance taking all the objectives into account. It is clear that, among the population, it is possible to remove the Pareto front and identify the second best Pareto front, and so on. In this way, one is able to rank the various fronts. For establishing the Pareto front rank number of each solution, the concept of dominance is used. Indeed, the individuals that dominate all the others are checked so that the first and all the other fronts are established. Then these individuals are excluded and the same operation is repeated. Therefore, it is possible to obtain an integer number of Pareto fronts. Many multi-objective optimization techniques leverage this concept for evolving the population throughout generations. The methodology and implementation of MO algorithms are thus completely different from the single-objective ones. In this study, we have only applied these algorithms in the framework of space trajectory optimization. Moreover, we have employed and benchmarked three different multi-objective optimizers: a nondominated sorting genetic algorithm (NSGA-II) [8], a multi-objective evolutionary algorithm with decomposition (MOEA/D) [9], and a multi-objective ant colony optimizer (MHACO), which has entirely been developed in this study, and whose detailed description is presented in Section II-C. Furthermore, all these algorithms have only been applied in the framework of space trajectory optimization.

B. Hypervolume Metric

Before going into details in the algorithm description, it is first necessary to introduce the concept of hypervolume, as this has not only been used for assessing the performance of the algorithms, but also in the design of the novel ant colony optimizer. As we have already mentioned, MO optimizers often return a Pareto-front of several individuals, rather than a single-best individual. Moreover, for general optimization problems, the returned Pareto fronts are often different when different algorithms are used. Therefore, there is the need for assessing the quality of each Pareto-front compared to the others. For doing this, several convergence and diversity metrics exist. These typically assess whether the found individuals belong to the Pareto front and how diverse the Pareto front is (i.e., how distant the individuals are within the same front). The hypervolume metric is a popular metric that combines both the diversity and convergence information in one single value. This metric was first introduced in [10] and it has soon become one of the most used performance metrics for MO algorithms. Indeed, not only does this metric evaluate both convergence and diversity using a single value, but it also has the property of being strictly Pareto compliant, meaning that the Pareto-optimal front has the key characteristic of maximizing the hypervolume value. This means that any dominated set will result in a lower hypervolume value.

Given a set of $\mathbf{x} \in \mathbb{R}^n$ decision vectors and a set of $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^m$ objective functions, we have already discussed that we can define the Pareto-optimal set as: $\{\mathbf{x} \in \mathbb{R}^n | \nexists\, \mathbf{x}' \in \mathbb{R}^n : \mathbf{x}' \prec \mathbf{x}\}$. The corresponding image in the objective space $\mathbb{R}^m$ is called Pareto front.

Typical definitions of the hypervolume indicator are based on polytopes [11]. Nevertheless, without introducing this concept it is still possible to introduce the hypervolume concept (which is nothing more than a generalization of the area, and volume concepts in n-dimensions) in a simpler way. The computation of the hypervolume in 2-dimensions is shown in Fig. 1.

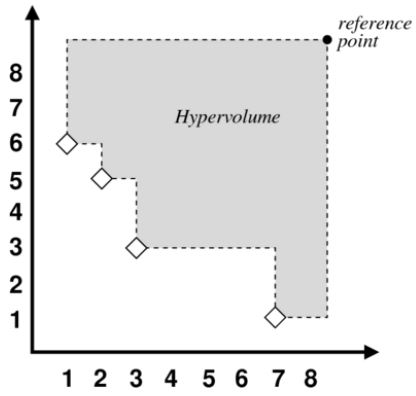

Fig. 1. Graphical representation of the hypervolume computation in two dimensions [3].

By taking a sub-set B of the objective functions space and letting Λ denote the Lebesgue measure (which is the common method to measure subsets of n-dimensional Euclidean space, and which corresponds to the concepts of length, area and volume for the case of n=1, 2 and 3, respectively), then the hypervolume (I_H) can be defined as:

$$I_H(B, \mathbf{y}_{ref}) = \Lambda\left(\bigcup_{\mathbf{y} \in B} \left\{\mathbf{y}' | \mathbf{y} \prec \mathbf{y}' \prec \mathbf{y}_{ref}\right\}\right),\ B \subset \mathbb{R}^m \quad (4)$$

where m is the objective space dimension, $\mathbf{y}$, $\mathbf{y}' \in B$ belong to a sub-set of the overall objective function vectors and where $\mathbf{y}_{ref} \in \mathbb{R}^m$ refers to a reference point that should be dominated by all Pareto-optimal solutions. It is important to notice that the hypervolume value can be computed both on a single individuals and on a set of individuals: we will exploit both these features. The former, in particular, will be leveraged to perform the internal ranking of the algorithm, whereas the latter will especially be used when the hypervolume is used as performance metric to evaluate and compare different algorithms performances.

Albeit recent improvements, the exact computation of the hypervolume for high dimensions is still a bottleneck and quickly becomes unfeasible for objective space dimensions higher than 10. For overcoming this problem, a lot of research has been carried out for approximating the hypervolume computation for high dimensions, while still using exact algorithms for low dimensions (such as 2, 3 or 4). In particular, in [12], a study is performed to benchmark the state-of-art hypervolume algorithms. From this study, it is concluded that for two and three dimensions, two exact algorithms shall be used [13], [14], due to their low computational time. However, for higher dimensions (i.e., from four to ten), the Walking Fish Group algorithm [15] seems to be the most suitable. This algorithm takes advantage of the bounding boxes to compute the exclusive contributions of points that are then used for the hypervolume computation. The asymptotic time does not seem to be very good, but experimental and theoretical evidence shows that is the fastest exact algorithm for high dimensions [16], [17]. For dimensions bigger than 10, however, the computation time increases toward unfeasible values. Typically, this forced researchers to scale down the problems before treating them. In [12], it is demonstrated that for the dimensions larger than 10 the best approximation algorithm in terms of accuracy of the approximation and runtime is the algorithm by Bringmann and Friedrich [18], which uses a Monte Carlo-like sampling method together with a racing approach to approximate the hypervolume computation.

To conclude, we will make use of these studies to establish the algorithm to be used for either computing the hypervolume exactly or for approximating it (depending on the objective function dimension). In particular, we will use the following algorithms:

- For $M = 2$: dimension-sweep exact algorithm [13].
- For $M = 3$: Beume exact algorithm [14].
- For $4 \leq M \leq 10$: Walking Fish Group exact algorithm [15].
- For $M > 10$: Bringmann and Friedrich approximation algorithm [18].

C. Algorithm Description

The original ACO, inspired by the biological process through which real ants forage food, was proposed by Dorigo in his PhD thesis [19]. However, we based our research on an extension of the original ACO version [20], [21]. This type of ACO is called extended ACO and its working principle consists in first randomly generating the initial population, with a uniform distribution, and then evolving the future generations of ants according to a solution archive, where the best and worst individuals are stored. In particular, the evolutionary operator used for evolving the individuals from the archive consists of a multi-kernel normal distribution that is the weighted sum of different normal probability density functions (PDF):

$$\mathcal{G}^h(x, \omega, \mu, \sigma) = \sum_{k=1}^{K} \omega_k^h \frac{1}{\sigma^h \sqrt{2\pi}} e^{-\frac{(x-\mu_k^h)^2}{2(\sigma^h)^2}} \quad (5)$$

where $h = 1, ..., n_{con}$ (in which n_{con} is the continuous dimension), whereas the triplet $(\omega_k^h, \mu_k^h, \sigma^h)$ represent the mean values, the standard deviations and the weights, of each normal distribution. These values are defined depending on the k^{th} position of the individual in the archive of size K (which is updated continuously throughout the evolution process) and on the integer or continuous variable considered (i.e., h). In particular, while the mean values μ_k^h and the weights ω_k^h are

[3]http://lopez-ibanez.eu/hypervolume, date of access: December 2019.

uniquely defined for each single ant in the archive, on the other hand, the standard deviations are the same across different ants, if the variable taken into account is the same. In other words, the standard deviations are only dependent on which variable h is being considered.

However, the practical strategy implemented for generating new individuals according to this weighted multi-kernel distribution did not make direct use of (5). Indeed, this strategy uses the same concept as the multi-kernel distribution, but employing a different method. Also, it allows the user to establish whether to focus more on the first individuals of the archive, for producing the offspring. This was achieved thanks to a peculiar formulation of the weights. In particular, for generating the weights, we have employed the following formulation [22]:

$$\omega_k^h = \frac{1}{qK\sqrt{2\pi}} e^{-\frac{(k-1)^2}{2q^2K^2}} \tag{6}$$

where k refers to the k^{th} individual in the solution archive, whose members range from 1 to K (which represents the solution archive size). Also, q is a user-defined parameter that substantially regulates the weights' definition. Indeed, if q is set to high values, then the resulting weights will be of similar magnitude and every individual in the solution archive will have a similar probability of being selected. Whereas if q is chosen to be very small, then the algorithm will focus more on the very first individuals of the solution archive (which are considered to be the best), assigning them higher values of the weights.

As a consequence, the sampling process is not done anymore by directly using the multi-kernel normal distribution function shown in (5), but it is performed by randomly selecting a single normal, taking into account that those with higher weights are more likely to be selected. This is carried out by only selecting one of the normal probability density functions of the multi-kernel PDF, whose selection is done by taking into account the probability p_k^h of choosing the k^{th} normal function, for each variables' component h. This probability is computed as:

$$p_k^h = \frac{\omega_k^h}{\sum_{k=1}^{K} \omega_k^h} \tag{7}$$

Of course, these probability functions have the property of having a sum equal to one (for each variable h). Mathematically, this means that:

$$\sum_{k=1}^{K} p_k^h = 1 \tag{8}$$

The process of sampling the chosen normal distribution is then executed as follows: from the probability values the cumulative probability is extracted, and each individual is assigned with that cumulative distribution function value. Afterwards, a random number between 0 and 1 is generated using a uniformly distributed random number generator, and the normal function is chosen accordingly. For how the weights are defined, the q parameter plays a pivotal role: if this parameter is set to low values, the chosen normal will more likely be among the top rated individuals, whereas if it is set to high values, a more uniformly distributed selection among all the members of the archive would be achieved. Usually, in the first phase of the algorithm optimization process, more evenly distributed weights are desirable, so that all the individuals in the archive are used for the generation of the offspring. Whereas when the generations' number is high and the evolution process is mature, we would like the algorithm to focus more its evolution process around the very first individuals in the archive, to possibly achieve convergence. For taking this into account another user-defined parameter was introduced, called the threshold ($\overline{T}$). This parameter is an integer that can range between 1 and the generation number. When the generation's value reaches the $\overline{T}$ value, then the q value is changed from the user-defined choice to a value of 0.01 (which makes sure that the algorithm focuses more on the very best individuals for generating new ones).

The parameters μ_k^h are simply the values of each variable component related to the k^{th} individual of the archive and they can be computed using the variables vector of each individual contained in the archive.

Finally, the standard deviations are computed as:

$$\sigma^h = \frac{D_{max}^h - D_{min}^h}{\overline{G}} \tag{9}$$

where $\overline{G}$ is a parameter, whereas D_{max} and D_{min} are the maximum and minimum distances between all the vectors in the archive (i.e., $(\mathbf{x})^{k=1,\dots,K}$), for each variable dimension (h). These distances can therefore be computed as:

$$D_{min}^h = min\{|x_h^p - x_h^q| : p, q \in \mathbb{N}, p \neq q \leq K\}$$
$$D_{max}^h = max\{|x_h^p - x_h^q| : p, q \in \mathbb{N}, p \neq q \leq K\}$$

Equation (9) resembles the one introduced in [20]. However, in that case, the number of generations were used instead of the parameter $\overline{G}$. The main reason why this modification was done in this work, is that by using the original formula, the standard deviations were soon driven to zero (as the number of generations increased), thus causing the algorithm not to generate offspring far enough from previous individuals, and to prematurely converge towards local optima during the evolution process. For avoiding this, a new user-defined parameter has been introduced: $\overline{NG}$. Basically, $\overline{G}$ plays the same role as the number of generation, with the difference that when $\overline{G}$ reaches the $\overline{NG}$ value, it is restarted again from 1 and it is increased again by one as the generation counter increases by one as well. For instance, let us assume that the user chooses $\overline{NG}$ = 7, then, this means that the $\overline{G}$ parameter will be increased until 7, when the population is evolved until the 7$^{\text{th}}$ generation. Afterwards, the $\overline{G}$ parameter is started again from 1 and then increased until 7, when the generation value reaches 14. This is repeated again with the same logic for the entire

evolution process. In this way, as long as the $D^h_{max} - D^h_{min}$ parameter is different than zero, σ^h is ensured not to reach very small values if the user decides so (by selecting a low value for the $\overline{NG}$ parameter). Of course, there is also the risk that $\overline{NG}$ is chosen so small that the consequent standard deviations are too large and the algorithm thus struggles to converge. For this reason, this parameter often requires to be adjusted according to the problem (especially when it is particularly cumbersome).

Two parameters are used to allow the user to monitor the spread of the individuals stored in the solution archive and thus establish how to set $\overline{NG}$. This aspect is strictly related to the standard deviation values since very spread out individuals might require a lower standard deviation value (hence a higher $\overline{NG}$), whereas very cluttered individuals might require a higher standard deviation value (hence a lower $\overline{NG}$), which may ensure a wider search in the variables' domain. For allowing the user to determine this more easily, the flatness in the variables (i.e., *dx*) and in the hypervolume (i.e., *dh*) of the individuals in the solution archive are saved for each generation. These are defined as:

$$dx = \sum_{i=1}^{n} \left(|x_{w,i} - x_{b,i}| \right) \tag{10}$$

$$dh = |h_w - h_b| \tag{11}$$

where h_w is the hypervolume value of the last (i.e., the worst) individual in the solution archive, whereas h_b of the first (i.e., the best). Also, n is the variables' dimension, i is the i^{th} variables' component, $x_{b,i}$ indicates the components of the first decision vector (i.e., the best in the solution archive) and $x_{w,i}$ the components of the last decision vector (i.e., the worst in the solution archive).

Once the triplet of pheromone values is defined, it is only necessary to establish how the solutions are ranked within the archive of size K. Indeed, having done the ranking, the evolutionary operator expressed in (5) is then capable of generating new ants from those in the archive, according to their positions. For ranking the individuals we have decided to implement an hypervolume-based strategy: meaning that the individuals with the highest hypervolume values will be placed at higher positions of the archive.

Having discussed the evolutionary operator, we needed to make sure to establish an effective strategy for ranking the individuals within the solution archive. As a novel approach, we propose to use a combination of the Pareto dominance concept and the hypervolume value, for ranking the individuals after they have been generated: this will allow us establish a ranking of the individuals and to have an update of the solution archive (in case new individuals that outperform those in the archive are generated). In particular, the population is first divided into nondomination levels, using the fast nondominated sorting strategy (the same as NSGA-II [8]), and then, the hypervolume metric is computed for each individual at each nondomination level. In this way, a sorting strategy is established: in case the individuals have different nondomination ranks, the lower nondomination level is preferred, whereas in case that the nondomination level is the same, then the hypervolume value of the single individual within the nondomination level is used for the ranking (where individuals with higher hypervolume values are preferred). Mathematically, this can be expressed using an hypervolume-comparison operator (i.e., $<_h$). According to this operator, an individual i is considered better than an individual j (and thus placed higher in the ranking) if the following is verified:

$$\begin{aligned} &i <_h j \text{ if } (i_{\text{rank}} < j_{\text{rank}}) \\ &\text{or } ((i_{\text{rank}} = j_{\text{rank}}) \text{ and } (hv(i) > hv(j))) \end{aligned}$$

where the $hv(i)$ represents the hypervolume contribution of the i^{th} individual. The hypervolume is computed by taking as reference point the nadir point of each nondomination level set of individuals, and by shifting it by 1% in each coordinate (in orther to ensure that the reference point is dominated by all the individuals in that level).

The algorithm strategy can be summarized as follows:

1) The initial population of size NP is randomly generated within the box-bounds of the variables. Also, a solution archive of size $K < NP$ is also generated using the individuals of the initial population.
2) If the generation number is higher than 1, then a merged list of $NP + K$ individuals is created, combining the archive and the newly generated offspring.
3) The hypervolume-comparison operator is used for ranking the individuals of the merged list, and establish which ones will be kept in the archive: this is updated only if at least one individual in the offspring outperforms the worst in the archive.
4) Once the archive is updated new offspring is generated using the evolutionary operator expressed in (5).
5) The algorithm goes back to Step 2 and repeats itself. It is thus clear that at the first iteration, only the initial population will be ranked, whereas, for all the others, both the solution archive and the population are sorted.

The user is thus left to choose the solution archive size (K) and the three parameters (i.e., $\overline{T}$, $\overline{NG}$ and q) to tune the pheromone values, and the optimization algorithm is capable of running. Furthermore, it is also possible, if needed, to make the search greedier by introducing a focus parameter ($focus$) (which will result in a contraction of the standard deviations), or to employ a stopping criterion ($evalstop$) that interrupts the evolution when the archive is not updated for a certain number of function evaluations.

Concerning the computation speed, MHACO has the possibility to parallelize the fitness computation of the individuals of the same generation on multiple threads (since the evaluation of each individual does not depend on the others), thus making the optimization process considerably faster for problems that require a high computation time of each individual fitness. However, for employing this strategy, it is required that the problem is thread-safe.

III. Space Trajectory Optimization Problems

The developed algorithm was tested on four space trajectory problems. Three can be found in *pykep* [23]: the scientific open-source library developed at the European Space Agency to provide basic tools for astrodynamics research[4]. The first two of these problems represent single leg transfers between two planets (in our case Mars and Venus) allowing up to three impulsive deep space maneuvers, with a time of flight between 20 and 400 days and with an infinite velocity between 0 and 4 km/s. On the other hand, the third of these problems is a more complex one and it represents a rendezvous mission to Jupiter modelled as a multi-gravity assist transfer with one deep space maneuver. The selected fly-by sequence is Earth-Venus-Earth-Mars-Earth and the spacecraft departs from the Earth and arrives at Jupiter. This problem is inspired by the ESA JUpiter ICy moons Explorer (JUICE) mission, which is a large-class mission within ESA's Cosmic Vision 2015-2025 programme[5]. It is planned to be launched in 2022 and to arrive at Jupiter in 2029. The spacecraft will then spend three years observing the gaseous planet and three of its moons: Ganymede, Callisto and Europa. A launcher model (i.e., Ariane 5) is also modeled within the problem, so that the final mass delivered at Jupiter is included in the objectives, together with the time of flight. Finally, the fourth problem we optimized, is a difficult low-thrust problem that simulates the journey of a solar sail towards a polar orbit around the Sun for exploring the Sun poles, starting from a GTO orbit. A thorough description of the general framework of this last problem can be found in [24]. The first three problems are bi-objective: in the case of the planet-to-planet transfers, 7 global variables are optimized over two objectives (an encoded version of the time and the total ΔV). Also in the case of the JUICE problem two objective are optimized (i.e., an encoded version of the final mass and the time of flight), but in this case with 27 variables. On the other hand, for the solar sailing mission, the problem has 12 variables and 4 objectives (i.e., the sail mass, the time of flight, and two penalties that regulate how precise the final orbit is both in terms of time of arrival and orbital elements). These problems have all been optimized with the same strategy and using the hypervolume metric as performance indicator (which encapsulates both the quality and diversity information of the Pareto front). The employed strategy for comparing the algorithms is the same as the one described in [25]. Each k algorithm (i.e., NSGA-II, MOEA/D and MHACO) was run t times with a controlled seed (for both the initial population and the internal evolutionary operators) for each run (so that not only each algorithm starts from the same initial population but the results are also reproducible when setting the same seeds) and over a G number of total generations. After the algorithms are run over all the generations, all their objective function vectors are normalized so that each component spans between 1 and 2. This mapping is applied to all the ktG populations (which size is 100 for the three pykep problems and 56 for the solar sailing problem). This means that all the coordinates of all the individuals across the algorithms are always bounded between [1,2]. Therefore, the reference point (for the hypervolume computation) can be chosen equal to 2.1 in each of its components (e.g., in the case of a bi-objective problem, it will be $a_{ref} = [2.1, 2.1]$). Furthermore, in this way, we are able to track the evolution of the hypervolume values of the populations throughout the generations, and compare those values (averaged through the number of trials t) across different algorithms (as both the mapping procedure of the objective function vectors and the reference point are the same for all of them).

[4]https://esa.github.io/pykep/, date of access: January 2020.

[5]https://sci.esa.int/web/juice, date of access: January 2020.

IV. Results

As already discussed in Section III, MOEA/D, NSGA-II and MHACO have been compared on the four space trajectory problems. In particular, besides using a population size of 100 and a generation size of 500 for all the problems except for the solar sail one (where a population size of 56 and a generation size of 40 was selected), NSGA-II was run with the following input parameters: a crossover probability of 0.9, a distribution index for crossover of 10, a mutation probability of 0.01 and a distribution index for mutation of 10. On the other hand, MOEA/D was run with the following input parameters: a grid weight generation method, a Chebyshev decomposition method, a size 20 of the neighbors, a crossover (CR) parameter equal to 1.0, an F parameter of 0.5, a distribution index for polynomial mutation of 20, a chance that the neighborhood is considered at each generation of 0.9 and a maximum number of copies reinserted in the population of 2 (where two diversity preservation mechanisms, discussed in [26], are used). Both NSGA-II and MOEA/D have not been tuned in their input parameters, as there is a wealth of knowledge and previous studies on the selection of these input parameters, that one can rely on, including the scientific publications where these algorithms have been introduced [8], [26], [9]. Finally, as far as MHACO is concerned, the chosen input parameters (for the Earth-Mars, Earth-Venus, JUICE and solar sail problems, respectively) are $K = 50, 25, 75, 56$, $\overline{NG} = 2, 1, 57, 8$, $q = 0.75, 0.75, 0.85, 1.0$, $\overline{T} = 100, 100, 200, 35$. The general strategy is to choose the archive size (K) as large as the number of individuals in the population, only if the problem is very difficult (e.g., the solar sail problem), and to select it around 25%, 50% or 75% of the population size otherwise (depending on the difficulty of the problem the archive size is typically increased). It is important to notice that the size of the archive only influences the internal evolutionary mechanism of the algorithm and not the number of solutions in which the hypervolume metric is computed for comparing the performance across different algorithms. Concerning the q parameter, a fixed value of 0.75 was chosen as baseline for the easier problems, and increased (to 0.85 and 1.0) for the JUICE and solar sailing problems, as they result to be more cumbersome. Concerning the $\overline{NG}$ parameter, a very low value (e.g., 1 or 2) was chosen for the easiest problems (e.g., Earth-Mars and Earth-Venus transfers), but its value was increased

to either 1/8 or 1/5 of the number of generations for the others. The reason is that for more difficult problems, we would like to allow the algorithm to search more around the local found optimal solutions (by reducing the standard deviations) before making the search spreader in the search space. On the other hand, for easier problems, the search space around the local optimal solutions is explored more rapidly and we therefore want to allow the algorithm explore other regions of the search space more often. Finally, the threshold value ($\overline{T}$) was set to either 20% or 40% or around 90% of the total generations number, depending on the difficulty of the problem (the higher, the more difficult). Furthermore, all the algorithms have been run (with the same initial populations) over t trials, where $t = 100$ for the first three problems and 10 for the solar sail one. The reason why this trial number was reduced for the latter case, is that the solar sail problem is way more computationally expensive, due to the long mission duration (i.e., each function evaluation takes 50 times the time it takes to run the other problems, on average). In Fig. 2, 3, 4 and 5, we show the average (across the trials) hypervolume values together with the standard deviations (highlighted as vertical bars up and below the mean values) as a function of the function evaluations. As can be seen, NSGA-II always outperforms the other algorithms for the easier Earth-Mars and Earth-Venus problems, whereas MHACO shows the best performances on both JUICE and the solar sailing problem. To corroborate this, the final average hypervolume values (at the latest generation) are also shown in Table I. Furthermore, as shown in the figures, the hypervolume curves have a tendency to grow and the standard deviations to diminish, as the function evaluations are increased. This is reasonable, as we would expect the population of the algorithms to improve the convergence (i.e., to find better Pareto-optimal fronts), as the number of evaluations are increased (i.e., exploitation is favoured w.r.t. exploration, as the num). Moreover, we also observe a different convergence behavior for the algorithms, especially in the solar sailing problem. Indeed, as can be seen from Fig. 5, both MHACO and NSGA-II have a wobbling behavior across the first function evaluations. The reason is due to the convergence behavior of MOEA/D, which monotonically improves the hypervolume values in the first evaluations, with strong and fast improvements. Therefore, since all the hypervolume values of all the algorithms are normalized and computed for the same reference point, if an algorithm strongly improves its population with respect to the others, this causes a decreasing behavior for the hypervolume of the latter algorithms. However, for the solar sailing problem, MOEA/D does not seem to be able to maintain such a behavior for higher function evaluations. Indeed, as the evaluations increase, MHACO manages to outperform its competitors.

V. Conclusions

In this paper, we have introduced a novel multi-objective ant colony optimizer for space trajectory optimization problems. As hinted from recent research, when applied to such problems, this type of algorithm seems to be competitive with

TABLE I
AVERAGE HYPERVOLUME VALUES AT THE LAST GENERATION FOR MOEA/D, NSGA-II AND MHACO ON THE EARTH-MARS (EM), EARTH-VENUS (EV), JUICE AND SOLAR SAIL (SS) TEST PROBLEMS.

	EM	*EV*	*JUICE*	*SS*
NSGA-II	1.209978	1.209992	1.2054	1.4482
MOEA/D	1.194581	1.198877	1.2051	1.4530
MHACO	1.209960	1.209973	1.2062	1.4562

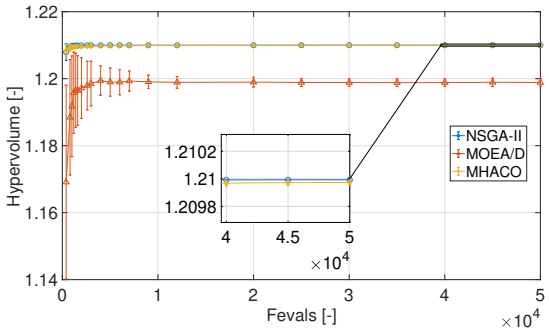

Fig. 2. Average hypervolume evolution of MOEA/D, NSGA-II and MHACO on the Earth-Venus bi-objective transfer problem.

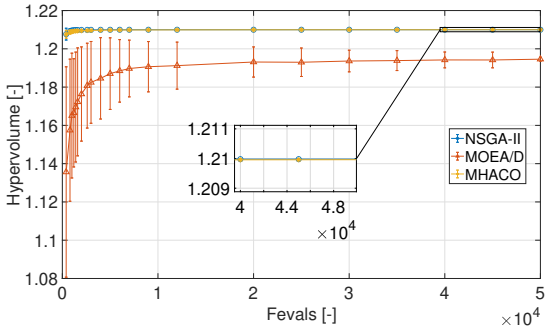

Fig. 3. Average hypervolume evolution of MOEA/D, NSGA-II and MHACO on the Earth-Mars bi-objective transfer problem.

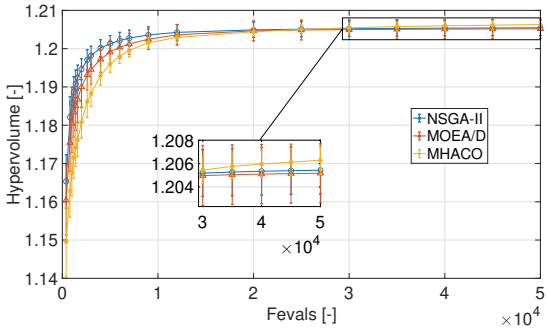

Fig. 4. Hypervolume evolution of MOEA/D, NSGA-II and MHACO on the JUICE bi-objective problem.

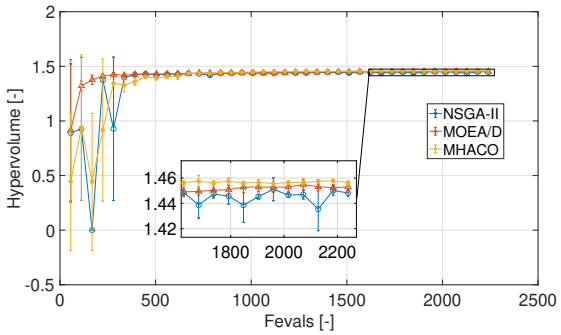

Fig. 5. Hypervolume evolution of MOEA/D, NSGA-II and MHACO on the four objectives solar sail problem.

the most used multi-objective algorithms (i.e., NSGA-II and MOEA/D). In particular, the developed algorithm has shown a better convergence behavior both on the difficult solar sail problem and on the bi-objective formulation of the ESA's JUICE mission. Furthermore, this algorithm was developed and made available in ESA's *pagmo2* optimization toolbox, which therefore allows future researchers to test and enhance the algorithms capabilities, in an open-source fashion.

References

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

[2] T. Vinkó, D. Izzo, and C. Bombardelli, "Benchmarking different global optimisation techniques for preliminary space trajectory design," in *58th international astronautical congress*. International Astronautical Federation Hyderabad, India, 2007, pp. 24–28.

[3] M. Schlüter, M. Gerdts, and J. Rückmann, "A numerical study of midaco on 100 minlp benchmarks," *Optimization*, vol. 61, no. 7, pp. 873–900, 2012.

[4] M. Schlüter, M. Wahib, and M. Munetomo, "Numerical optimization of esa's messenger space mission benchmark," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2017, pp. 725–737.

[5] M. Schlüter, "Midaco software performance on interplanetary trajectory benchmarks," *Advances in Space Research*, vol. 54, no. 4, pp. 744–754, 2014.

[6] M. Schlüter, C. H. Yam, T. Watanabe, and A. Oyama, "Parallelization impact on many-objective optimization for space trajectory design," *International Journal of Machine Learning and Computing*, vol. 6, no. 1, p. 9, 2016.

[7] F. Biscani and D. Izzo, "esa/pagmo2: pagmo 2.13.0," 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3603747

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[9] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on evolutionary computation*, vol. 11, no. 6, pp. 712–731, 2007.

[10] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Da Fonseca Grunert, "Performance assessment of multiobjective optimizers: An analysis and review," *TIK-Report*, vol. 139, 2002.

[11] E. Zitzler, *Evolutionary algorithms for multiobjective optimization: Methods and applications*. Citeseer, 1999, vol. 63.

[12] K. Nowak, M. Märtens, and D. Izzo, "Empirical performance of the approximation of the least hypervolume contributor," in *International Conference on Parallel Problem Solving From Nature*. Springer, 2014, pp. 662–671.

[13] A. P. Guerreiro, C. M. Fonseca, and M. T. Emmerich, "A fast dimension-sweep algorithm for the hypervolume indicator in four dimensions." in *CCCG*, 2012, pp. 77–82.

[14] N. Beume, C. M. Fonseca, M. López-Ibáñez, L. Paquete, and J. Vahrenhold, "On the complexity of computing the hypervolume indicator," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1075–1082, 2009.

[15] L. While, L. Bradstreet, and L. Barone, "A fast way of calculating exact hypervolumes," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, pp. 86–95, 2011.

[16] K. Bringmann and T. Friedrich, "Parameterized average-case complexity of the hypervolume indicator," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 2013, pp. 575–582.

[17] C. Priester, K. Narukawa, and T. Rodemann, "A comparison of different algorithms for the calculation of dominated hypervolumes," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 2013, pp. 655–662.

[18] K. Bringmann and T. Friedrich, "Approximating the least hypervolume contributor: Np-hard in general, but fast in practice," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2009, pp. 6–20.

[19] M. Dorigo, "Optimization, learning and natural algorithms," *PhD Thesis, Politecnico di Milano*, 1992.

[20] S. Martin, "Nonlinear mixed integer based optimization technique for space applications," Ph.D. dissertation, University of Birmingham, 2012.

[21] M. Schlüter, "Non-linear mixed-integer-based optimization technique for space applications." ESA Conference - International NPI Day, 2010.

[22] R. M. Bernardo Jr and P. C. Naval Jr, "Implementation of an ant colony optimization algorithm with constraint handling for continuous and mixed variable domains," in *Proceedings of the 10th Philippine Computing Science Congress, PCSC*, vol. 10, 2010, pp. 95–101.

[23] D. Izzo, "esa/pykep: The gym and more," 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3560642

[24] E. Mooij, R. Noomen, and S. Candy, "Evolutionary optimization for a solar sailing solar polar mission," in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, no. 6180, 2006.

[25] C. Igel, N. Hansen, and S. Roth, "Covariance matrix adaptation for multi-objective optimization," *Evolutionary computation*, vol. 15, no. 1, pp. 1–28, 2007.

[26] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii," *IEEE transactions on evolutionary computation*, vol. 13, no. 2, pp. 284–302, 2008.