Hybridizing Differential Evolution and Novelty Search for Multimodal Optimization Problems

Aritz D. Martinez
Tecnalia Research & Innovation, 48160, Derio, Spain
aritz.martinez@tecnalia.com

Eneko Osaba
Tecnalia Research & Innovation, 48160, Derio, Spain
eneko.osaba@tecnalia.com

Izaskun Oregi
Tecnalia Research & Innovation, 48160, Derio, Spain
izaskun.oregui@tecnalia.com

Iztok Fister Jr.
University of Maribor, Maribor, Slovenia
iztok.fister1@um.si

Iztok Fister
University of Maribor, Maribor, Slovenia
iztok.fister@um.si

Javier Del Ser
University of the Basque Country (UPV/EHU), 48013 Bilbao, Spain
javier.delser@ehu.eus

ABSTRACT

Multimodal optimization has shown to be a complex paradigm underneath real-world problems arising in many practical applications, with particular prevalence in physics-related domains. Among them, a plethora of cases within the computational design of aerospace structures can be modeled as a multimodal optimization problem, such as aerodynamic optimization or airfoils and wings. This work aims at presenting a new research direction towards efficiently tackling this kind of optimization problems, which pursues the discovery of the multiple (at least locally optimal) solutions of a given optimization problem. Specifically, we propose to exploit the concept behind the so-called Novelty Search mechanism and embed it into the self-adaptive Differential Evolution algorithm so as to gain an increased level of controlled diversity during the search process. We assess the performance of the proposed solver over the well-known CEC'2013 suite of multimodal test functions. The obtained outcomes of the designed experimentation supports our claim that Novelty Search is a promising approach for heuristically addressed multimodal problems.

CCS CONCEPTS

• **Theory of computation → Bio-inspired optimization**; *Random search heuristics*; • **Mathematics of computing →** *Evolutionary algorithms*;

KEYWORDS

Multimodal Optimization, Novelty Search, Differential Evolution

ACM Reference Format:
Aritz D. Martinez, Eneko Osaba, Izaskun Oregi, Iztok Fister Jr., Iztok Fister, and Javier Del Ser. 2019. Hybridizing Differential Evolution and Novelty Search for Multimodal Optimization Problems. In *Genetic and Evolutionary Computation Conference Companion (GECCO '19 Companion), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3319619.3326799

GECCO '19 Companion, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3326799

1 INTRODUCTION

For years optimization problems have been a paradigm lying at the core of thousands of industrial processes, such as engineering design or production planning [34]. In this regard, by an optimization problem we refer to the situation where the value of a set of variables must be tailored so as to maximize or minimize a given measure of quality or fitness that quantifies how *good* any solution is with respect to the problem at hand. In many practical cases, the lack of an analytical formulation for the fitness function or its complex mathematical tractability makes it difficult to deterministically compute the optimum solution to a problem (i.e. the fittest combination of values for the aforementioned variables), to the point of requiring non-affordable computational times even for the simplest problem formulations. When this is the case (usually referred to as NP-hard problems), randomized heuristic search techniques can be adopted instead [32]. Heuristics attempt at solving NP-hard problems by resorting to approximative self-learning search strategies, which allow them to explore the search domain of the problem in a more efficient fashion than greedy or enumerative methods. However, this increased computational complexity comes along with a lack of optimality guarantees, namely, there is no certainty that the solution produced by a randomized heuristic solver is the best for a given problem [5].

Many different flavors of optimization problems can be found in the literature, ranging from the existence of multiple conflicting objectives to dynamic problem formulations, each spanning its own flurry of specialized heuristic methods [16]. Among them, multimodal optimization (MMO) deals with problems where the fitness function has multiple global optima that are of practical value for the application at hand. Thereby, the goal is to find as many of these optima as possible by designing a technique capable, not only of finding these optima, but also of retaining it during the search process [12]. The need for exploring different yet equally interesting regions of the solution space leads to different specialized methods that are usually inserted into the search procedure of evolutionary and swarm intelligence heuristics. Among them, niching techniques stand as arguably the most utilized schemes for MMO to date [28].

From a practical perspective, MMO has served as a computational model for real-world problems in many heterogeneous areas,

such as physics [46], medicine [3, 21] or graph theory [9]. However, it is in aerospace and aerodynamics where MMO has been notably present in manifold problems. A excerpt of exemplary works in this field is next provided by beginning with [10, 40], which proposes two algorithms to investigate multimodality in aerodynamic shape optimization problems: a gradient-based multi-start algorithm based on Sobol sampling, and a hybrid optimizer combining a Genetic Algorithm with a gradient-based algorithm. The same problem is also addressed in [30], again under an MMO modeling approach based on the Common Research Model (CRM) wing benchmark, defined by the Aerodynamic Design Optimization Discussion Group. Another interesting research is presented in [37], in which a population-based global MMO framework is proposed to solve a aerodynamic wing optimization problem via three different optimization approaches. Likewise, the authors in [11] elaborate on the effect of uncertainty representation on the results of robust airfoil (cross-sectional shape of a wing) shape optimization. After highlighting the importance of a proper airfoil design to be robust with respect to uncertainties in operating conditions, authors formulate the shape design problem as a MMO instance. All experimentations in this study are conducted by using three different benchmark instances, with varying degrees of complexity and multimodality. Interested readers are referred to recently published works such as [17, 44, 47] and references therein for further bibliographic depth on this subject.

The capital importance of MMO in the above areas, and the vibrant research activity noted lately around innovative multimodal optimization methods, motivate the new research direction presented in this work: the application of Novelty Search for efficiently dealing with MMO problems. Novelty Search (NS) was proposed in 2008 [26] as a means to enhance the exploratory ability of population-based search algorithms. Specifically the work posed a milestone by showing that unprecedentedly good performances in optimization problems could be achieved by directing the search in terms of diversity rather than exclusively in terms of fitness value. Our research hypothesis in this work builds upon recent work [20], in which we have evinced the promising performance of NS for single-objective optimization problems, which we aim to extrapolate to MMO problems. To this end, we have developed an adaptive version of Differential Evolution (DE, [43]), namely, the self-adaptive Differential Evolution (jDE, [8]), and endowed it with a NS-inspired mechanism. Informed insights on the quality of our proposed approach are obtained from an experimentation benchmark over the CEC'2013 MMO test suite [27], which comprises 20 different MMO instances of varying size. The performance of our methods is compared to that of the basic versions of DE, jDE and four ad-hoc MMO solvers from the literature. The results obtained from the performed experiments buttress our hypothesis around the postulated potential of NS for MMO problems.

The remainder of the paper is organized as follows: Section 2 provides background on DE, jDE and NS, whereas Section 3 is devoted to the description of the proposed NS-based scheme. Next, Section 4 outlines and discusses the experimentation and finally, Section 5 ends the paper with conclusions and further work.

2 BACKGROUND

This section delves into baseline concepts required for properly grasping the contributions of our research work. First, we introduce the basic concepts of DE and jDE. We finish this section by sketching the fundamentals of NS.

2.1 Differential Evolution

DE is a classical evolutionary algorithm which optimizes a problem with regard to a given fitness function $f : \mathbb{R}^D \rightarrow \mathbb{R}$, where D denotes the dimensionality or number of decision variables of the problem. First introduced in [43], many DE variants have been ever since proposed in the literature [13, 33]. The canonical version of DE targets numerical optimization problems by representing solutions as a population of real-valued D-length vectors:

$$\mathbf{x}_i^{(t)} = \left(x_{i,1}^{(t)}, x_{i,2}^{(t)}, \ldots, x_{i,D}^{(t)}\right) \text{ for } i = 1, \ldots, N_p, \tag{1}$$

where N_p denotes the size of the population or set of candidates $\mathcal{P}^{(t)} = \{\mathbf{x}_1^{(t)}, \mathbf{x}_2^{(t)}, \ldots, \mathbf{x}_{N_p}^{(t)}\}$ involved in the optimization process at generation t. At every one of such generations mutation, crossover and selection operators are applied to each candidate $\mathbf{x}_i^{(t)}$. First, the mutation operator is applied, with DE/rand/1/bin and DE/best/1/bin being two of the most frequently used variations [14]. The basic mutation operates by generating a *trial* vector, which yields from the scaled difference between two randomly chosen candidates $\mathbf{x}_{r_1}^{(t)}$ and $\mathbf{x}_{r_2}^{(t)}$, plus another random candidate $\mathbf{x}_{r_0}^{(t)}$. Mathematically:

$$\mathbf{u}_i^{(t)} = \mathbf{x}_{r_0}^{(t)} + F_i^{(t)} \cdot (\mathbf{x}_{r_1}^{(t)} - \mathbf{x}_{r_2}^{(t)}), \tag{2}$$

where $\mathbf{F}^{(t)} = \left\{F_i^{(t)}\right\} \in [0.1, 1.0]^{N_P}$ represents the scale factor. Once this mutation has been performed, DE performs a crossover operation, in which DE selects the value of the variables between the parent and trial candidate as:

$$w_{i,j}^{(t)} = \begin{cases} u_{i,j}^{(t)} & \text{if } rand_j \leq CR \text{ or } j = j_{rand}, \\ x_{i,j}^{(t)} & \text{otherwise,} \end{cases} \tag{3}$$

where $CR \in [0, 1]$ is a parameter that controls the fraction of parameters which are drawn from the trial solution; $j_{rand} \in [1, 2, \ldots, D]$ is a randomly selected position from trial vector; and $rand_j$ is the realization of a uniformly distributed random variable with support $\mathbb{R}[0, 1]$. It is important to note that the condition $j = j_{rand}$ ensures that $w_{i,j}^{(t)}$ differs from the original solution in at least one position. Finally, the selection of the remaining candidate for the next generation $t + 1$ is done by selecting the fittest candidate, i.e.:

$$\mathbf{x}_i^{(t+1)} = \begin{cases} \mathbf{w}_i^{(t)} & \text{if } f(\mathbf{w}_i^{(t)}) \geq f(\mathbf{x}_i^{(t)}), \\ \mathbf{x}_i^{(t)} & \text{otherwise,} \end{cases} \tag{4}$$

where $f(\cdot)$ denotes the fitness function of the problem at hand. It should be left clear that other selection criteria could be imposed to the heuristic, including elitism and other variants.

2.2 Self Adaptive Differential Evolution

In the self-adaptive DE (jDE) proposed in [8], the crossover rate CR and scale factor $\mathbf{F}^{(t)}$ are self-adapted during the evolutionary search. As a result, candidates in jDE are encoded as:

$$\mathbf{x}_i^{(t)} = \left(x_{i,1}^{(t)}, x_{i,2}^{(t)}, \ldots, x_{i,D}^{(t)}, CR_i^{(t)}, F_i^{(t)}\right), \text{ for } i = 1, \ldots, N_p, \tag{5}$$

namely, as the concatenation of the optimization variables and the self-adaptive parameters. The equations governing this adaptability for the crossover rate and scale factor are:

$$F_i^{(t+1)} = \begin{cases} F_l + r_1(F_u - F_l) & \text{if } r_2 \leq \tau_1, \\ F_i^{(t)} & \text{otherwise,} \end{cases} \quad (6)$$

$$CR_i^{(t+1)} = \begin{cases} r_3 & \text{if } r_4 \leq \tau_2, \\ CR_i^{(t)} & \text{otherwise,} \end{cases} \quad (7)$$

where $\{r_m\}_{m=1}^4$ are realizations of uniformly distributed random variables with support $\mathbb{R}[0,1]$; τ_1 and τ_2 stand for the learning rate of the adaptation; and F_l and F_u represents the lower and upper bounds of the parameter $\mathbf{F}^{(t)}$, respectively.

This adaptive version of the DE solver has proven to perform remarkably in problems springing from a wide variety of fields, such as energy [22, 39], software engineering [24] or industry [2]. For this reason and for its simplicity, we have embraced jDE as the search heuristic to be combined with a NS-based technique.

2.3 Novelty Search

As has been pointed in Section 1, the rationale behind NS is to increase the diversity of the population by finding novel candidates in the *behavioral* space instead of the *search* space. It is known that unless overcome anyhow, candidates tend to collapse and stagnate in the search space [25], but they do not necessarily do the same in the space of variables. In this alternative space the amount of novel brought by a candidate $\mathbf{x}$ can be measured as:

$$\rho(\mathbf{x}) = \frac{1}{k}\sum_{i=0}^{k} d(\mathbf{x}, \boldsymbol{\mu}_i), \quad (8)$$

where $d(\cdot,\cdot)$ denotes the Euclidean distance, and k is the number of neighbor candidates selected from the ordered subset of neighbors of $\mathbf{x}$, namely, $\mathcal{N}(\mathbf{x}) = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k\} \subseteq \mathcal{P}$, with $\mathcal{P}$ representing the population. The size is problem-dependent, so the value of k must be selected experimentally. Under the NS approach, the above measure of diversity can be adopted as a selection criteria between successive generations of the search heuristic, in such a way that only the individuals inducing more diversity into the remaining population are kept. Intuitively, the measure of distance between candidates $d(\cdot,\cdot)$ can be also selected depending on the characteristics of the problem.

Despite the diversity of applications where NS has showcased its efficiency and outstanding performance [23, 29], in general a NS technique is weakly defined. This leaves the question of how to tailor the search in hands of the algorithm developer, and it is strictly related with the problem which is trying to be solved [19].

3 PROPOSED NS-BASED JDE FOR MMO

With all this, the algorithm proposed in this work focuses in solving multimodal optimization problems by hybridizing jDE with a NS-based mechanism (NSjDE). The hybridization of these two concepts is extremely interesting for this purpose: this is so because one of the main design directives when devising new algorithms for MMO is to generate enough diversity in the population towards finding and maintaining as many local optima of the solution space space as possible. For this reason, NS seems to be a promising method for this kind of optimization problems, since it may promote the generation of new diverse candidates without wasting computational resources. This helps the solver not be stuck in local optima, yet requires further modifications to retain already found niches.

Previous studies have unveiled that jDE tends to form neighborhoods during its search process [1]. When accounted together with its self-adaptive behavior, this tendency makes jDE a very suitable algorithmic choice to work with sub-populations and split the search space in neighborhoods depending on the number of optima. However, the convergence and optimality of solutions provided by the off-the-shelf version of jDE can be limited in MMO problems. To overcome this issue and enhance the multimodal exploration capability of this heuristic, we propose a vicinity-guided crossover by which once an optimum is found, the exploration of other areas is promoted. Meanwhile, the discovered optima keep improving locally by the exploitative capability of the jDE mutation operator.

As shown in many studies [8], the canonical version of jDE performs fitness-based selection. By contrast, our proposed method takes into account the exploration area of the parent individual, as well as the location of the mutated offspring to perform the replacement. In order to incorporate the exploration radius, the search space is normalized and the candidate encoding adapted as:

$$\mathbf{x}_i^{(t)} = \left(x_{i,1}^{(t)}, \ldots, x_{i,D}^{(t)}, \sigma_i^{(t)}, CR_i^{(t)}, F_i^{(t)}\right), \text{ for } i = 1, \ldots, N_p, \quad (9)$$

where $x_{i,d}^{(t)} \in \mathbb{R}[0,1]\ \forall d = 1, \ldots, D$, and $\sigma_i^{(t)}$ represents the exploration radius of the i-th candidate, namely, the maximum distance at which candidate $\mathbf{x}_i^{(t)}$ can be replaced. At the same time, in order to maintain the best candidates of the population and preserve those with better fitness values, each candidate is endowed with a radius size according to their fitness value which, as will be later explained, depends on a inverse normalized function given by:

$$\widehat{f}(\mathbf{x}_i^{(t)}) = 1 - \frac{f(\mathbf{x}_i^{(t)}) - f_{\min}^{(t)}}{f_{\max}^{(t)} - f_{\min}^{(t)}}, \quad (10)$$

where $\mathbf{x}_i^{(t)} \in \mathcal{P}$. The two additional values $f_{\min}$ and $f_{\max}$ in the expression above represent the historical fitness values from the beginning of the execution to generation (t). These values are checked every time a candidate is evaluated, and are updated as:

$$f_{\min}^{(t+1)} = \begin{cases} f(\mathbf{x}_i^{(t)}) & \text{if } f(\mathbf{x}_i^{(t)}) \leq f_{\min}, \\ f_{\min}^{(t)} & \text{otherwise,} \end{cases} \quad (11)$$

$$f_{\max}^{(t+1)} = \begin{cases} f(\mathbf{x}_i^{(t)}) & \text{if } f(\mathbf{x}_i^{(t)}) \geq f_{\max}, \\ f_{\max}^{(t)} & \text{otherwise.} \end{cases} \quad (12)$$

Our goal with this mechanism is to make all candidates be as exploratory as possible in early stages of the search process, i.e, all candidates should be configured with maximum exploration radii. To this end, we define $\mathcal{R}^{(t)} = \{\sigma_1^{(t)}, \ldots, \sigma_{N_p}^{(t)}\}$ (i.e. the set of radii of all candidates at generation (t)), and initialize it to $\sigma_i^{(0)} = 1.0\ \forall i = 1, \ldots, N_p$. Once the population has been evaluated in this initial generation, the candidate with best fitness value is inserted into the *maxima* set $\mathcal{M}$ with $\sigma_m^{(0)} = 1.0\ \forall \mathbf{x}_m^{(t)} \in \mathcal{M}$. This newly introduced set $\mathcal{M}$ comprises the group of best candidates found during the search. At the end of the search process, this set

is returned as the solution to the MMO problem, as is designed to retain the whole group of sought optima.

Figure 1 describes the workflow of the proposed NS-based jDE solver, which is divided in three different phases. The first one is *initialization*, where the population $\mathcal{P}^{(t)} = \{\mathbf{x}_1^{(t)}, \ldots, \mathbf{x}_{N_p}^{(t)}\}$ is created and evaluated. Then the initial set of radii $\mathcal{R}^{(0)} = \{\sigma_1^{(0)}, \ldots, \sigma_{N_p}^{(t)}\}$ is initialized to 1.0 $\forall \sigma_i^{(0)} \in \mathcal{R}$. After then, the *evolution* phase is run while a stop criterion (e.g. a maximum number of function evaluations) is not met. This second phase consists of the sequential application of mutation and crossover operators as described in Subsection 2.2. Then, a custom replacement operator – later detailed in Subsection 3.1 – composes a evolved version of the population in generation $(t+1)$, taking in account the exploration radius of every parent candidate and the point in the search space where the trial vector has been generated. At this point subsets $\mathcal{A}$ and $\mathcal{B}$ are fed with the best and feasible novel candidates, respectively. Finally, during the *novelty search* phase diversity is induced in the final population guided by two parameters: the number of neighbors used to calculate the so-called novelty value k and R, the number of candidates in the final population that are going to be replaced by novel solutions. In order to maintain the best candidates, the aforementioned $\mathcal{M} = \{\mathbf{m}_1^{(t)}, ..., \mathbf{m}_M^{(t)}\}$ is introduced, with M representing the number of maxima. This set $\mathcal{M}$ is updated and filled with the best candidates of the population.

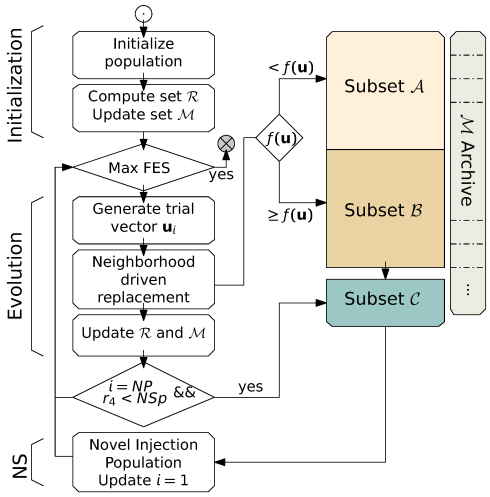

Figure 1: Workflow of the proposed NS-based jDE.

3.1 Custom Mutation Operator

As mentioned above, this study resorts to a custom variation operator, which is selected to be the well-known *DE/rand/1* configuration in the same form as in [18]. Both *DE/nrand/1* and *DE/nrand/2* have demonstrated a good performance for enhancing the niching tendency of the *jDE*. The modification made to these operators hinges on the use of the nearest neighbor $\boldsymbol{\phi}_i^{(t)}$ given by

$$\boldsymbol{\phi}_i^{(t)} = \underset{\substack{j=1,\ldots,Np \\ j \neq i}}{\arg\min}\, d\left(\mathbf{x}_i^{(t)}, \mathbf{x}_j^{(t)}\right), \tag{13}$$

where $d(\cdot,\cdot)$ denotes Euclidean distance as per (8). Our proposal aims to take advantage of the diversity introduced by NS to guide the search to unexplored areas. For this to occur, when a new optimum is found, the value of the niche radius is saved while the optimum is maintained in $\mathcal{M}$. Thus, the search space is partitioned according to the number of optima and their niche radii $\sigma_m^{(t)}\ \forall m : \mathbf{x}_m^{(t)} \in \mathcal{M}$. In this way, the search space would be sectioned in a number of sub-populations or neighborhoods such that $\mathcal{N}_i^{(t)} \subseteq \mathcal{P}$. Additionally, all candidates with a distance less than the radius of a given optimum belong to the same neighborhood. Furthermore, solutions that do not belong to any group are considered to be in the neighborhood $\mathcal{N}_{free}$, composed by candidates that are not linked to any optimum. Figure 2 exemplifies this strategy for a one-dimensional MMO problem instance; as can be seen in this plot, candidates are distributed among the neighborhoods.

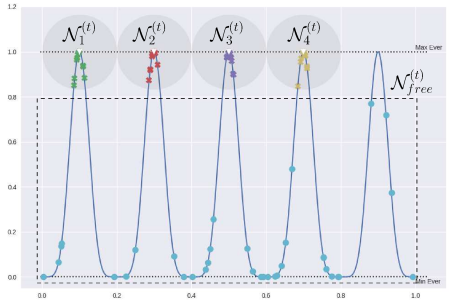

Figure 2: Example of neighborhoods and $\mathcal{N}_{free}$ with the 1D function *Equal Maxima*.

When a candidate belongs to a certain neighborhood, an additional parameter $\rho \in [0, 1]$ determines the probability of the candidate to mutate with other solutions from its same neighborhood. The extreme value $\rho = 0.0$ represents the random selection of the new candidate from the entire population, penalized by the amount of neighboring individuals inside each niche. The main goal of this parameter is to increase the probability of each candidate to be mutated with an element of the same neighborhood.

Therefore, the custom mutation taking into account this new selection operator is given by:

$$\mathbf{u}_i^{(t)} = \boldsymbol{\phi}_i^{(t)} + F_i^{(t)} \cdot (\mathbf{x}_{a,r_0}^{(t)} - \mathbf{x}_{b,r_1}^{(t)}), \tag{14}$$

where $r_{0,1} \in [1, ..., N_p]$ represents two randomly selected candidates from $\mathcal{P}^{(t)}$ following a custom distribution guided by the above described ρ parameter and the number of elements by neighborhood. Additionally, $\mathbf{x}_{a,r_0}^{(t)}$ and $\mathbf{x}_{b,r_1}^{(t)}$ $(\mathbf{x}_{a,r_0}^{(t)} \neq \mathbf{x}_{b,r_1}^{(t)})$ represent neighborhood driven selected candidates from $\mathcal{N}_a$ and $\mathcal{N}_b$ neighborhoods, respectively. On the other hand, optima are treated as special candidates which save, along with the radius and the candidate, the last three neighbors with which they have been mutated.

3.2 Neighborhood Driven Replacement

The replacement of candidates in MMO is not the same as the one used in its canonical version for mono-modal optimization. In problems with a single optimum, the replacement can be done by only using the fitness metric. When dealing with multiple optima, however, it is crucial to also account for the diversity within the population to capture and retail such optima without replacing them continuously.

This being said, in this work we propose to assign a specific value to the candidates aiming to evaluate their quality by their normalized fitness as per Expression 10. Thus, when a candidate has a high fitness value, it only accepts changes in a low distance variation. On the other hand, if a candidate has very low fitness value, it accepts more drastic changes, following the procedure described in Section 3.1.

Many approaches have been used in the literature for properly determining which candidates are to be replaced, with the main goal of keeping the niches or sub-populations as less overcrowded as possible. Renowned examples along the history of MMO are crowding methods [15, 31], which have been extensively used for this kind of problems. Another approach with notable presence in the literature consists of using a formula to assign probabilities of being a survivor, taking fitness values as its reference. Among all available methods, it is interesting to let *DE* evolve and form neighborhoods; then, once Novelty Search is applied, we perform a *clearance* of overcrowded niches. In this way the radius or exploration distance is computed as follows:

$$\sigma_i^{(t+1)} = \begin{cases} \widehat{f}(\mathbf{x}_i^{(t)}) & \text{if } \mathbf{x}_i^{(t)} \in \mathcal{N}_{free}, \\ 1.0 & \text{if } \mathbf{x}_i^{(t)} \in \mathcal{N}_m,\ m \in \{1, \ldots, M\}, \end{cases} \tag{15}$$

Let us elaborate further on the particularities of the proposed replacement criterion. On one hand, Equation 4 in Section 2.1) determines how the replacement is done in standard jDE. On the other hand, in NSjDE the replacement is done by considering the Euclidean distance between the new candidate and its parent. If the new candidate falls within a distance smaller than the radius associated to its parent, the replacement is done based on fitness value. Otherwise, if the distance is higher than the radius of the parent, the parent will accept a change, and is swapped with another candidate in the area to which the new candidate belongs. Each time a replacement is performed, the candidate is inserted in $\mathcal{A}$ or $\mathcal{B}$ depending on its relative fitness with respect to its parent (better or worse, respectively). Algorithm 1 describes a pseudo-code illustrating this behavior.

Algorithm 1 Custom Replacement (assuming maximization).

$\mathbf{x}_i^{(t)}$: Parent solution
$d(\mathbf{x}_i^{(t)}, \mathbf{x}_j^{(t)})$: Distance function between candidate i and j
$\sigma_i^{(t)}$: Parent radius
$\mathbf{u}_i^{(t)}$: Candidate solution (trial)
procedure Replacement operator
  if $d(\mathbf{x}_i^{(t)}, \mathbf{u}_i^{(t)}) \leq \sigma_i^{(t)}$ **then**
    if $f(\mathbf{u}_i^{(t)}) \geq f(\mathbf{p}^{(t)})$ **then**
      $\mathbf{x}_i^{(t+1)} \leftarrow \mathbf{u}_i^{(t)}$
      Add $\mathbf{u}_i^{(t)}$ to subset $\mathcal{A}$ (*Good trial*)
    else
      Add $\mathbf{u}_i^{(t)}$ to subset $\mathcal{B}$ (*Feasible novel trial*)
  else
    Compute nearest neighbor of $\mathbf{u}_i^{(t)}$ as $\boldsymbol{\phi}_i^{(t)}$
    if $f(\mathbf{u}_i^{(t)}) \geq f(\boldsymbol{\phi}_i^{(t)})$ **then**
      $\boldsymbol{\phi}_i^{(t+1)} \leftarrow \mathbf{u}_i^{(t)}$
  Update $\sigma_i^{(t+1)}$ $\forall i$ as per (15)

3.3 Novelty Search for MMO

As has been mentioned in previous sections, the benefits of NS were first exposed over mono-modal, single-objective optimization problems. This seminal work and other contributions appearing ever since induce diversity by using the candidates produced in the last generation so as to introduce novel candidates in the population. However, in MMO we do not know a priori how many optima characterize the problem has, nor where they are are located over the search space. Therefore, our main search rationale is not related only with the fitness, but also with the efficient traversing a more unexplored area as possible with the series of candidates yielded by the search heuristic. As a result of this design principle, the search can be guided through the solution space in a more efficient way, maintaining good candidates within the population and attracting other ones.

For this purpose we partially embrace our initial findings in [20] to yield the overall search algorithm outlined graphically in Figure 1. We maintain three different subsets $\mathcal{A}, \mathcal{B}, \mathcal{C} \subseteq \mathcal{P}$, each one with a fixed size. On the one hand, $\mathcal{A}$ and $\mathcal{B}$ retain the best and worst candidates produced in the last generation of the jDE heuristic, respectively. On the other hand, C represent the candidates which are more likely to be novel (as told by a measure of diversity) and, therefore, to be inserted in the population again. Such a measure of diversity can be furnished by using Expression (8), for which an inner distance $d(\cdot, \cdot)$ between candidates must be defined. For numerical optimization not linked to any specific application, the default choice is an Euclidean distance between candidates

$$d(\mathbf{x}_i^{(t)}, \mathbf{x}_j^{(t)}) = \sqrt{\sum_{d=1}^{D} \left(x_{i,d}^{(t)} - x_{j,d}^{(t)} \right)^2}. \tag{16}$$

Back to Figure 1, when a trial candidate $\mathbf{u}_i$ achieves better fitness than the candidate that it is going to replace, it is inserted both into the population and in $\mathcal{A}$, whereas the replaced candidate is fed to $\mathcal{B}$. Otherwise, if the candidate can not be introduced in the population, it is inserted in $\mathcal{B}$. Once the t-th generation comes to end, subsets $\mathcal{A}$ and $\mathcal{B}$ should not be empty. In that case, if r_{ns}, (being this a random number selected from a normal distribution) is lower than the NS parameter $NS_P \in [0.0, 1.0]$, the NS algorithm is executed as shown in Algorithm 2.

4 EXPERIMENTS AND RESULTS

This section is devoted to the presentation of the experimentation carried out to assess the performance of the proposed NSjDE. The main goal of these experiments is to showcase that NS is a promising mechanism to solve MMO problems with random search heuristics. To this end, we propose a jDE adaptation for MMO that in no way we intend to compete with the best algorithm in the current state of the art, but rather to gauge the improvement when NS is added to jDE. Thereby, the experimentation can be divided into three different steps:

- *First experiment*, where we compare the Peak Rate (PR) and accuracy between the proposed NSjDE (configured with [$R = 15$ and $|\mathcal{P}| = 100$) and a basic version of this method with the same

Algorithm 2 Multimodal Novelty Search.

$\mathcal{A}$: Set of optima
$\mathcal{B}$: Set of possible novel
$\mathcal{C}$: Set of real novel candidates
$\mathcal{M}$: Set of actual optima
$\mathcal{P}^{(t)}$: Population at generation t
procedure NOVELTY SEARCH
    dBA ← Pairwise distances from $\mathcal{B}$ to $\mathcal{A}$
    Update distances of candidates in $\mathcal{C}$
    Update candidates in $\mathcal{C}$ using new candidates as per dBA
    dm ← Compute distances from $\mathcal{C}$ to $\mathcal{M}$
    Clearing: Select candidates in population to be replaced
    Select the R most novel candidates from $\mathcal{C}$
    Replace in $\mathcal{P}^{(t)}$ selected candidates and yield $\mathcal{P}^{(t+1)}$

population size, but without any diversity injection into the population (i.e. $R = 0$). The objective of this first experiment is to quantify whether the inclusion of NS improves the performance of the jDE solver.

- *Second experiment*, where we compare NSjDE with three parameter sets: $[R = 15, |\mathcal{P}| = 100]$, $[R = 40, |\mathcal{P}| = 250]$ and $[R = 0, |\mathcal{P}| = 250]$. Parameter tuning is arguably one of the most important aspects to take into account when performing heuristic benchmarks. This second set of experiments aims at showing the sensitivity of the developed MMO solver by inspecting the performance differences under different values of its search parameters, with emphasis on those impacting more severely on the computational efficiency (namely, replacement and population size).
- *Third experiment*, which comprises a comparison of NSjDE with $[R = 15, |\mathcal{P}| = 100$ and $[R = 40, |\mathcal{P}| = 250]$ to four MMO algorithms that took part in the competitions held during the 2013 and 2015 editions of IEEE Congress on Evolutionary Computation (CEC). We recall that it is not our goal to discern whether the developed NS-jDE outperforms the state of the art, as this would require adapting and tailoring the NS mechanism to more powerful global search heuristics than the one adopted in this study. Instead, we want to show in this third experiment that our approach is competitive when compared to other methods utilized in competitions in the past.

Experiments are run over the CEC'2013 test suite for MMO [27], wich comprises 20 maximization test functions with dimensions from $D = 1$ to $D = 20$. The benchmark is described in Table 1.

Figure 3 gives an idea of the complexity of the considered benchmark by showing the shape of some of the problems that can be found in the competition. The main reason for choosing the CEC'2013 test suite is its heterogeneity, which contemplates very diverse multi-modal problem instances to tackle. As a summary, functions f_1 to f_5 should be easy to solve, with just one and two dimensions, and a low amount of optima. By contrast, functions like *Vincent* or *Shubert* contain a much higher number of optima, which makes them difficult to detect, capture and retain within the population. Finally, *composition* functions pose a challenge due to the presence of local optima where the algorithm can get stuck.

Table 1: CEC'2013 test suite for MMO. D: dimension of the problem. $\mathbf{f_i^* = f(x^*)}$: fitness value of optimal solution $\mathbf{x^*}$. No. Max: number of optima to be detected.

i	Functions	Description	D	$f_i^* = f(\mathbf{x}^*)$	No. Max.
1	f_1	Five-Uneven-Peak Trap	1	200.0	2
2	f_2	Equal Maxima	1	1.0	5
3	f_3	Uneven Decreasing Maxima	1	1.0	1
4	f_4	Himmelblau	2	200.0	4
5	f_5	Six-Hump Camel Back	2	1.03163	2
6	f_6	Shubert	2	186.731	18
7	f_7	Vincent	2	1.0	36
8	f_8	Shubert	3	2709.0935	81
9	f_9	Vincent	3	1.0	216
10	f_{10}	Modified Rastrigin - All Optima	2	-2.0	12
11	f_{11}	Composition Function 1	2	0.0	6
12	f_{12}	Composition Function 2	2	0.0	8
13	f_{13}	Composition Function 3	2	0.0	6
14	f_{14}	Composition Function 3	3	0.0	6
15	f_{15}	Composition Function 4	3	0.0	8
16	f_{16}	Composition Function 3	5	0.0	6
17	f_{17}	Composition Function 4	5	0.0	8
18	f_{18})	Composition Function 3	10	0.0	6
19	f_{19})	Composition Function 4	10	0.0	8
20	f_{20})	Composition Function 4	20	0.0	8

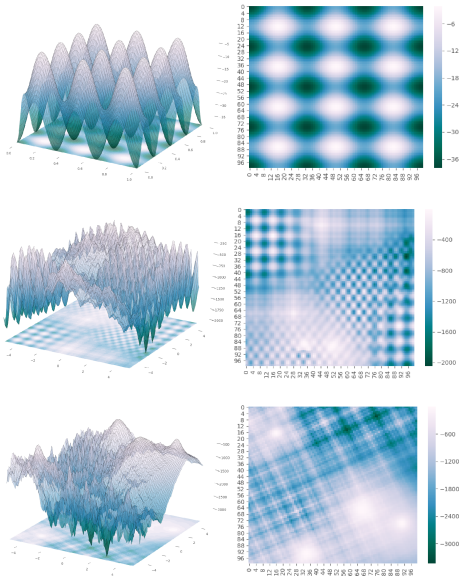

Figure 3: Instances f_{10}, f_{12} and f_{13}, and their heatmaps.

The ending criterion of all NSjDE methods is fixed to a maximum amount of fitness evaluations. This number is defined based on the characteristics of each problem, and it can be analyzed in Table 2.

Finally, the evaluation score that we will hereafter use in our results is the Peak Rate (*PR*), which is calculated as the number of global optimal found over the different runs, expressed as a percentage. Therefore, the algorithm with higher *PR* value is declared to dominate the benchmark.

Table 2: Maximum number of evaluations per instance.

Range	Maximum function evaluations
f_1 to f_5 (1D or 2D)	$5.0 \cdot 10^4$
f_6 to f_{11} (2D)	$2.0 \cdot 10^5$
f_6 to f_{12} (3D or higher)	$4.0 \cdot 10^5$

4.1 Results and Discussion

We now discuss the results obtained from the experimentation described previously. To begin with, Table 3 (next page) lists the results obtained for the first experiment. As has been mentioned, the main goal of this experiment is to assess how jDE works *with* and *without* novelty injection. To this end, each problem has been run 10 independent times for both solvers. Statistics shown in the Table represent the mean *PR* value averaged over all the executions. Furthermore, five further values are shown per solver and problem instance, each corresponding to a different value of ε. This parameter represents the precision in terms of fitness value required for accepting a produced solution as a hit of an optimum: the lower ε is, the more strict the declaration of a produced solution as a global optima will be. For the sake of clarity, best results are shown in bold. The table is also complemented with a last row with the average scores reached by every algorithm over different precision levels.

Both methods have been run with the same configuration: a population size of $N_p = |\mathcal{P}| = 100$ (except for instances f_8 and f_9, whose higher number of optima require an accordingly larger population of size $N_p = 250$). One of them has no novelty injection ($R = 0$) whereas the other, as indicated previously, is configured with $R = 15$ and $k = 5$. This configuration has been chosen by following the guidelines of [20], meaning that k nearest candidates have been used to compute the novelty distance, and R novel candidates are inserted in each NS iteration. Outcomes of this first phase show an improvement in the performance of NSjDE algorithm. This alternative obtains best results in all the instances, with emphasis on f_6-f_9, f_{11}-f_{13} and f_{16}. Most of these instances are the ones with lowest dimensionality ($D = 2$ and $D = 3$), containing a high number of maxima. The better performance of the NS-based solver proves that MMO solution spaces are explored more efficiently if NS is incorporated to the algorithmic workflow of the search solver. This leads ultimately a higher amount of detected optima. It should be emphasized that the algorithm keeps steadily showing off better performance statistics up to precision $\varepsilon = 10^{-5}$.

Results of the second experiment are shown in Table 4 (next page). This experiment focuses on how the algorithm performs when configured with three parameter value sets. The first corresponds to the above described NSjDE ($R = 15$ and $|\mathcal{P}| = 100$]). The other alternative configurations can be regarded as two computationally heavy variants, with $|\mathcal{P}| = 250$ for all datasets, and $R = 15$ and $R = 0$, respectively. In this table, two different *PR* values are depicted per instance, denoting the attained peak rates for $\varepsilon = 10^{-1}$ and $\varepsilon = 10^{-4}$. The reason of depicting these two values is to evaluate the exploration and exploitation capacity of each solver.

When analyzing the results in Table 4, a turning point in the performance of the algorithm can be noted. On the one hand, more optima have been found by NSjDE-2 ($R = 40$ and $|\mathcal{P}| = 250$) in many highly-dimensional problems (like f_{14}, f_{17} or f_{18}), whereas a worse performance can be observed in functions such as f_8 or f_{11}. This fact lends the opportunity to think about how the different characteristics of the benchmarks can affect the configuration of the amount of candidates participating in the NS procedure. Furthermore, we can see in this same table that the difference between the values in the columns $\varepsilon = 10^{-1}$ and $\varepsilon = 10^{-4}$ is, in general, larger than that in Table 3. This issue elucidates again the effect of the novelty injection in the exploitation of the search space.

Continuing with the above analysis, we can also confirm that the alternative without NS ($R = 0$ and $|\mathcal{P}| = 250$) is dominated by the first two alternatives, thereby stressing on the importance of this mechanism when seeking multiple optima. Furthermore, the light version of NSjDE (NSjDE with $R = 15$ and $|\mathcal{P}| = 100$) renders the same overall performance at a precision of $\varepsilon = 10^{-4}$. This is a great advantage for this alternative, concluding that similar performance levels can be obtained by using much less computational resources.

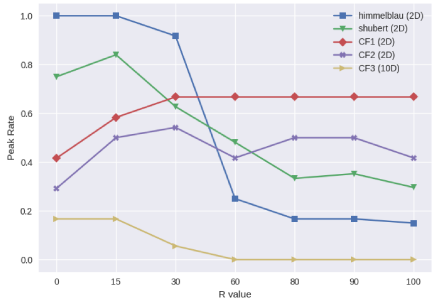

Figure 4: PR versus R (number of injected novel candidates).

An interesting discussion can be held around the exploitative capacity of the proposed method, which could be affected by the *clearing* stage performed before replacing the candidates in the NS injection. This issue can be further inspected in Figure 4, where the PR values obtained using different R values are depicted for five different problems. These results have been obtained by NSjDE configured with $R = 15$, $|\mathcal{P}| = 100$ and using $\varepsilon = 10^{-5}$ as the reference precision level. In this figure it is shown that an increase in R yields a lower exploitation capacity of the method, as exposed by its worse PR values. This is the reason why in the second experimentation NSjDE-2 solvers are endowed with a higher population size in order to maintain a proper balance between exploration and exploitation.

Finally, results of the third experiment are summarized in Table 5. As has been argued, our aim with these third tests is to compare our proposed NSjDE with two parameter sets ([$R = 15, |\mathcal{P}| = 100$] and [$R = 40, |\mathcal{P}| = 250$]) with other algorithms participating in the CEC competitions, towards demonstrating that NSjDE can rival in this competition. Four well-known methods have been selected for this comparison, because of their efficiency and similarities with the proposed NSjDE: a Multi-Sub-Swarm Particle Swarm Optimization algorithm (MSSPSO, [48]), a Multinational Evolutionary Algorithms (MEA, [45]), DECG [41] and the restart (μW , α) Covariance Matrix Adaptation Evolution Strategy with increasing population (IPOP-CMA-ES, [6]). As in the previous step, PR values for $\varepsilon = 10^{-1}$ and $\varepsilon = 10^{-4}$ precision levels are shown per method and problem

Table 3: Comparison (Peak Rate, *PR*) between NSjDE and NSjDE without novelty injection ($R = 0$).

	NSjDE: R=15, $\|\mathcal{P}\|$=100					NSjDE: R=0, $\|\mathcal{P}\|$=100				
Function	$\varepsilon = 10^{-1}$	$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$	$\varepsilon = 10^{-1}$	$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$
f_1	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
f_2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
f_3	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
f_4	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
f_5	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
f_6	**0.875**	**0.875**	**0.875**	**0.875**	**0.875**	0.792	0.792	0.764	0.764	0.750
f_7	**0.972**	**0.972**	**0.972**	**0.972**	**0.972**	0.958	0.958	0.931	0.924	0.924
f_8	**0.754**	**0.752**	**0.742**	**0.733**	**0.730**	0.358	0.358	0.349	0.343	0.339
f_9	**0.588**	**0.588**	**0.559**	**0.469**	**0.431**	0.321	0.321	0.252	0.209	0.185
f_{10}	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
f_{11}	**0.697**	**0.697**	**0.697**	**0.697**	**0.697**	0.417	0.417	0.417	0.417	0.417
f_{12}	**0.643**	**0.643**	**0.625**	**0.589**	**0.589**	0.281	0.281	0.281	0.281	0.281
f_{13}	**0.667**	**0.667**	**0.667**	**0.667**	**0.667**	0.381	0.381	0.381	0.381	0.381
f_{14}	**0.167**	**0.167**	**0.167**	**0.167**	**0.167**	**0.167**	**0.167**	**0.167**	**0.167**	**0.167**
f_{15}	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**
f_{16}	**0.375**	**0.333**	**0.333**	**0.333**	**0.333**	0.167	0.167	0.167	0.167	0.167
f_{17}	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**
f_{18}	**0.167**	**0.167**	**0.167**	**0.167**	**0.167**	**0.167**	**0.167**	**0.167**	**0.167**	**0.167**
f_{19}	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**
f_{20}	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**
Average	**0.620**	**0.618**	**0.615**	**0.608**	**0.606**	0.525	0.525	0.519	0.516	0.514
Tot. Ave.			**0.613**					0.520		

Table 4: Comparison (Peak Rate, *PR*) between *light* and *heavy* variants of the proposed NSjDE, and the NSjDE without novelty injection ($R = 0$).

	NSjDE: R=15, $\|\mathcal{P}\|$=100		NSjDE: R=40, $\|\mathcal{P}\|$=250		NSjDE: R=0, $\|\mathcal{P}\|$=250	
Function	$\varepsilon = 10^{-1}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-1}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-1}$	$\varepsilon = 10^{-4}$
f_1	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
f_2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
f_3	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
f_4	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
f_5	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
f_6	0.875	0.875	**1.000**	**1.000**	**1.000**	**1.000**
f_7	0.972	0.972	**1.000**	**1.000**	**1.000**	**1.000**
f_8	0.754	0.733	0.617	0.571	**0.864**	**0.839**
f_9	0.588	0.469	**0.679**	**0.569**	0.477	0.477
f_{10}	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
f_{11}	**0.697**	**0.697**	0.694	0.583	0.417	0.417
f_{12}	**0.643**	**0.589**	0.625	0.542	0.375	0.313
f_{13}	**0.667**	**0.667**	**0.667**	**0.667**	**0.667**	**0.667**
f_{14}	0.167	0.167	**0.417**	**0.417**	0.167	0.167
f_{15}	0.125	0.125	0.125	0.125	0.125	0.125
f_{16}	**0.375**	0.333	0.352	**0.259**	0.333	0.333
f_{17}	0.125	**0.125**	**0.300**	0.025	0.125	**0.125**
f_{18}	0.167	**0.167**	**0.214**	0.143	0.167	0.167
f_{19}	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**
f_{20}	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**	**0.125**
Average	0.620	**0.608**	**0.647**	**0.608**	0.598	0.594

instance, aiming to analyze the exploratory and exploitative capacity of every solver. The last row of the table also indicates the average PR scores for each algorithm in the benchmark. The main conclusion drawn from this third experimentation is that NSjDE outperforms their counterparts in terms of exploration and exploitation, reaching a much better average performance and confirming that it is a promising method for solving MMO problems, being able to rival other solvers from the literature.

5 CONCLUSIONS AND FUTURE WORK

This work has elaborated on a new way to induce diversity for population-based solvers addressing MMO problems. In doing so, a variation of the self-adaptive DE algorithm is proposed, whose guided mutation strategy and a tailored neighborhood-based replacement strategy makes the population efficiently traverse unexplored search space regions and retain captured optima along its way. The insertion of novel (diverse) candidates performed by the Novelty Search mechanism embodies an enhancement of the baseline jDE algorithm, but we have experimentally seen that the parameter configuration of the solver should be tuned differently as per the problem under consideration. A comparison with other well-known CEC'13 and CEC'15 algorithms has been carried out, verifying that NSjDE could enter these competitions. Nevertheless, there is still room for further improvement towards making the algorithm more competitive with respect to the state of the art.

It is indeed in this direction where future efforts are foreseen to be invested. To begin with, the most immediate research line to be undertaken is to lessen the computational cost of the algorithm, for which some adjustments will be imprinted to the NS mechanism in regards to repeated distance calculations. On the order hand, alternative Novelty Search induction strategies will be also investigated. Finally, the findings resulting from this research will be utilized for solving real problems arising in Aerospacial Sciences, such as the design of optimal aerodynamic shapes.

ACKNOWLEDGEMENTS

The authors would like to thank the Basque Government for its funding support through the EMAITEK program.

REFERENCES

[1] Sarah Hazwani Adnan, Shir Li Wang, Haidi Ibrahim, and Theam Foo Ng. 2018. An Overview on the Application of Self-Adaptive Differential Evolution. In *Proceedings of the 10th International Conference on Computer Modeling and Simulation*. ACM, 82–86.

Table 5: Comparison between NSjDE variants and MSSPSO, MEA, DECG and IPOP-CMA-ES.

	NSjDE: R=15, $\|\mathcal{P}\|$=100		NSjDE: R=40, $\|\mathcal{P}\|$=250		MSSPSO		MEA		DECG		IPOP-CMA-ES	
Function	$\varepsilon = 10^{-1}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-1}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-1}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-1}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-1}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-1}$	$\varepsilon = 10^{-4}$
f_1	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.810	0.050	**1.000**	**1.000**	0.78	0.78
f_2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.996	**1.000**	**1.000**	0.772	0.752
f_3	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	1.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
f_4	**1.000**	**1.000**	**1.000**	**1.000**	0.685	0.005	**1.000**	0.005	**1.000**	**1.000**	0.730	0.725
f_5	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.050	**1.000**	0.640	**1.000**	**1.000**	**1.000**	**1.000**
f_6	0.875	0.875	**1.000**	**1.000**	0.004	0.000	0.188	0.000	0.997	**0.997**	0.100	0.090
f_7	0.972	0.972	**1.000**	**1.000**	0.989	0.030	0.490	0.383	0.991	0.656	0.112	0.111
f_8	**0.754**	**0.733**	0.617	0.571	0.000	0.000	0.000	0.000	0.309	0.308	0.021	0.020
f_9	0.588	0.469	**0.679**	**0.569**	0.571	0.000	0.195	0.112	0.334	0.244	0.027	0.027
f_{10}	**1.000**	**1.000**	**1.000**	**1.000**	0.883	0.007	0.992	0.965	**1.000**	**1.000**	0.377	0.313
f_{11}	**0.697**	**0.697**	0.694	0.583	0.170	0.000	0.930	0.477	0.687	0.667	0.493	0.470
f_{12}	0.643	0.589	0.625	0.542	0.010	0.000	0.355	0.138	**0.718**	**0.718**	0.345	0.300
f_{13}	0.667	**0.667**	0.667	**0.667**	0.043	0.000	**0.790**	0.337	0.657	0.657	0.253	0.253
f_{14}	0.167	0.167	0.417	0.417	0.667	0.000	**0.913**	0.350	0.497	**0.490**	0.337	0.327
f_{15}	0.125	0.125	0.125	0.125	0.125	0.000	**0.840**	0.135	0.333	**0.330**	0.178	0.178
f_{16}	0.375	**0.333**	0.352	0.259	0.000	0.000	**1.000**	0.000	0.077	0.073	0.267	0.267
f_{17}	0.125	0.125	0.300	0.025	0.000	0.000	**0.507**	0.000	0.025	0.025	0.160	**0.160**
f_{18}	0.167	**0.167**	**0.214**	0.143	0.000	0.000	0.000	0.000	0.000	0.000	0.097	0.097
f_{19}	0.125	0.125	0.125	0.125	0.000	0.000	0.000	0.000	0.000	0.000	**0.193**	**0.193**
f_{20}	**0.125**	**0.125**	**0.125**	**0.125**	0.000	0.000	0.000	0.000	0.000	0.000	0.128	0.128
Average	0.620	**0.608**	**0.647**	**0.608**	0.407	0.155	0.601	0.279	0.581	0.558	0.369	0.360

[2] Fawaz S Al-Anzi and Ali Allahverdi. 2007. A self-adaptive differential evolution heuristic for two-stage assembly scheduling problem to minimize maximum lateness with setup times. *European Journal of Operational Research* 182, 1 (2007), 80–94.

[3] ADG Anderson, CE McNaught, J MacFie, I Tring, P Barker, and CJ Mitchell. 2003. Randomized clinical trial of multimodal optimization and standard perioperative surgical care. *British journal of surgery* 90, 12 (2003), 1497–1504.

[4] Alfredo Arias-Montano, Carlos A Coello Coello, and Efrén Mezura-Montes. 2012. Multiobjective evolutionary algorithms in aeronautical and aerospace engineering. *IEEE Transactions on Evolutionary Computation* 16, 5 (2012), 662–694.

[5] Anne Auger and Benjamin Doerr. 2011. *Theory of randomized search heuristics: Foundations and recent developments*. Vol. 1. World Scientific.

[6] A. Auger and N. Hansen. 2005. A restart CMA evolution strategy with increasing population size. In *IEEE Congress on Evolutionary Computation*, Vol. 2. 1769–1776 Vol. 2. https://doi.org/10.1109/CEC.2005.1554902

[7] Dimitris Bertsimas and John N Tsitsiklis. 1997. *Introduction to linear optimization*. Vol. 6. Athena Scientific Belmont, MA.

[8] Janez Brest, Sao Greiner, Borko Boskovic, Marjan Mernik, and Viljem Zumer. 2006. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE transactions on evolutionary computation* 10, 6 (2006), 646–657.

[9] Ran Cheng, Miqing Li, Ke Li, and Xin Yao. 2018. Evolutionary Multiobjective Optimization-Based Multimodal Optimization: Fitness Landscape Approximation and Peak Detection. *IEEE Transactions on Evolutionary Computation* 22, 5 (2018), 692–706.

[10] Oleg Chernukhin and David W Zingg. 2013. Multimodality and global optimization in aerodynamic design. *AIAA journal* 51, 6 (2013), 1342–1354.

[11] Laurence W Cook and Jerome P Jarrett. 2017. Robust airfoil optimization and the importance of appropriately representing uncertainty. *AIAA Journal* (2017), 3925–3939.

[12] Swagatam Das, Sayan Maity, Bo-Yang Qu, and Ponnuthurai Nagaratnam Suganthan. 2011. Real-parameter evolutionary multimodal optimizationâĂŤA survey of the state-of-the-art. *Swarm and Evolutionary Computation* 1, 2 (2011), 71–88.

[13] Swagatam Das, Sankha Subhra Mullick, and Ponnuthurai N Suganthan. 2016. Recent advances in differential evolution–an updated survey. *Swarm and Evolutionary Computation* 27 (2016), 1–30.

[14] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. 2011. Differential evolution: a survey of the state-of-the-art. *IEEE transactions on evolutionary computation* 15, 1 (2011), 4–31.

[15] Kenneth Alan De Jong. 1975. Analysis of the behavior of a class of genetic adaptive systems. (1975).

[16] Javier Del Ser, Eneko Osaba, Daniel Molina, Xin-She Yang, Sancho Salcedo-Sanz, David Camacho, Swagatam Das, Ponnuthurai Nagaratnam Suganthan, Carlos A Coello-Coello, and Francisco Herrera. 2019. Bio-inspired Computation: What's Next? *Swarm and Evolutionary Computation, to appear* (2019).

[17] Xiaosong Du, Anand Amrit, Andrew S Thelen, Leifur T Leifsson, Yu Zhang, Zhong-Hua Han, and Slawomir Koziel. 2017. Aerodynamic Design of a Rectangular Wing in Subsonic Inviscid Flow by Direct and Surrogate-based Optimization. In *35th AIAA Applied Aerodynamics Conference*. 4366.

[18] M. G. Epitropakis, D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis. 2011. Enhancing Differential Evolution Utilizing Proximity-Based Mutation Operators. *IEEE Transactions on Evolutionary Computation* 15, 1 (Feb 2011), 99–119. https://doi.org/10.1109/TEVC.2010.2083670

[19] Iztok Fister, Andres Iglesias, Akemi Galvez, Javier Del Ser, and Eneko Osaba. 2018. Using Novelty Search in Differential Evolution. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*. Springer, 534–542.

[20] Iztok Fister, Andres Iglesias, Akemi Galvez, Javier Del Ser, Eneko Osaba, Iztok Fister, MatjaÅ¿ Perc, and Mitja Slavinec. 2019. Novelty search for global optimization. *Appl. Math. Comput.* 347 (2019), 865 – 881. https://doi.org/10.1016/j.amc.2018.11.052

[21] M Gatt, ADG Anderson, BS Reddy, P Hayward-Sampson, IC Tring, and J MacFie. 2005. Randomized clinical trial of multimodal optimization of surgical care in patients undergoing major colonic resection. *British journal of surgery* 92, 11 (2005), 1354–1362.

[22] Sujan Ghimire, Ravinesh C Deo, Nathan J Downs, and Nawin Raj. 2018. Self-adaptive differential evolutionary extreme learning machines for long-term solar radiation prediction with remotely-sensed MODIS satellite and Reanalysis atmospheric products in solar-rich cities. *Remote Sensing of Environment* 212 (2018), 176–198.

[23] Jorge Gomes, Pedro Mariano, and Anders Lyhne Christensen. 2015. Devising effective novelty search algorithms: A comprehensive empirical study. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 943–950.

[24] Jitendra Kumar and Ashutosh Kumar Singh. 2018. Workload prediction in cloud using artificial neural network and adaptive differential evolution. *Future Generation Computer Systems* 81 (2018), 41–52.

[25] Jouni Lampinen, Ivan Zelinka, et al. 2000. On stagnation of the differential evolution algorithm. In *Proceedings of MENDEL*. 76–83.

[26] Joel Lehman and Kenneth O Stanley. 2008. Exploiting open-endedness to solve problems through the search for novelty.. In *ALIFE*. 329–336.

[27] Xiaodong Li, Andries Engelbrecht, and Michael G Epitropakis. [n. d.]. Benchmark functions for CECâĂŹ2013 special session and competition on niching methods for multimodal function optimization. ([n. d.]).

[28] Xiaodong Li, Michael G Epitropakis, Kalyanmoy Deb, and Andries Engelbrecht. 2017. Seeking multiple solutions: an updated survey on niching methods and their applications. *IEEE Transactions on Evolutionary Computation* 21, 4 (2017), 518–538.

[29] Antonios Liapis, Georgios N Yannakakis, and Julian Togelius. 2015. Constrained novelty search: A study on game content generation. *Evolutionary computation* 23, 1 (2015), 101–129.

[30] Zhoujie Lyu, Gaetan KW Kenway, and Joaquim RRA Martins. 2014. Aerodynamic shape optimization investigations of the common research model wing benchmark. *AIAA Journal* 53, 4 (2014), 968–985.

[31] Samir W Mahfoud. 1995. Niching methods for genetic algorithms. *Urbana* 51, 95001 (1995), 62–94.

[32] Zbigniew Michalewicz and David B Fogel. 2013. *How to solve it: modern heuristics*. Springer Science & Business Media.

[33] Ferrante Neri and Ville Tirronen. 2010. Recent advances in differential evolution: a survey and experimental analysis. *Artificial Intelligence Review* 33, 1-2 (2010), 61–106.

[34] Godfrey C Onwubolu and BV Babu. 2013. *New optimization techniques in engineering*. Vol. 141. Springer.

[35] Christos H Papadimitriou and Kenneth Steiglitz. 1998. *Combinatorial optimization: algorithms and complexity*. Courier Corporation.
[36] A. Petrowski. 1996. A clearing procedure as a niching method for genetic algorithms. In *Proceedings of IEEE International Conference on Evolutionary Computation*. 798–803. https://doi.org/10.1109/ICEC.1996.542703
[37] DJ Poole, CB Allen, and TCS Rendall. 2018. Global Optimization of Wing Aerodynamic Optimization Case Exhibiting Multimodality. *Journal of Aircraft* (2018), 1–16.
[38] Mike Preuss. 2010. Niching the CMA-ES via Nearest-better Clustering. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '10)*. ACM, New York, NY, USA, 1711–1718. https://doi.org/10.1145/1830761.1830793
[39] Harish Pulluri, Ram Naresh, and Veena Sharma. 2017. An enhanced self-adaptive differential evolution based solution methodology for multiobjective optimal power flow. *Applied Soft Computing* 54 (2017), 229–245.
[40] Raul-Cristian Roman, Radu-Emil Precup, and Radu-Codrut David. 2018. Second order intelligent proportional-integral fuzzy control of twin rotor aerodynamic systems. *Procedia computer science* 139 (2018), 372–380.
[41] Jani Rönkkönen et al. 2009. *Continuous Multimodal Global Optimization with Differential Evolution-Based Methods*. Lappeenranta University of Technology.
[42] B. Sareni and L. Krahenbuhl. 1998. Fitness sharing and niching methods revisited. *IEEE Transactions on Evolutionary Computation* 2, 3 (Sep. 1998), 97–106. https://doi.org/10.1109/4235.735432
[43] Rainer Storn and Kenneth Price. 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* 11, 4 (1997), 341–359.
[44] Alexandra-Iulia Szedlak-Stinean, Claudia-Adina Bojan-Dragos, Radu-Emil Precup, and Mircea-Bogdan Radac. 2018. Gain-Scheduling Control Solutions for a Strip Winding System with Variable Moment of Inertia. *IFAC-PapersOnLine* 51, 4 (2018), 370–375.
[45] R. K. Ursem. 1999. Multinational evolutionary algorithms. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, Vol. 3. 1633–1640 Vol. 3. https://doi.org/10.1109/CEC.1999.785470
[46] Dong-Kyun Woo, Jong-Ho Choi, Mohammad Ali, and Hyun-Kyo Jung. 2011. A novel multimodal optimization algorithm applied to electromagnetic optimization. *IEEE Transactions on Magnetics* 47, 6 (2011), 1667–1673.
[47] Yin Yu, Zhoujie Lyu, Zelu Xu, and Joaquim RRA Martins. 2018. On the influence of optimization algorithm and initial design on wing aerodynamic shape optimization. *Aerospace Science and Technology* 75 (2018), 183–199.
[48] Jun Zhang, De-Shuang Huang, and Kun-Hong Liu. 2007. Multi-sub-swarm particle swarm optimization algorithm for multimodal function optimization. In *2007 IEEE Congress on Evolutionary Computation*. 3215–3220. https://doi.org/10.1109/CEC.2007.4424883