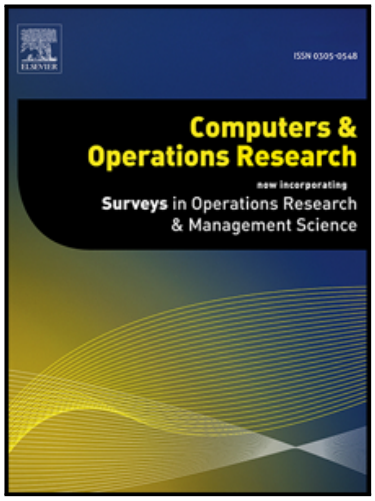

A similarity-based neighbourhood search for enhancing the balance exploration–exploitation of differential evolution

Eduardo Segredo, Eduardo Lalla-Ruiz, Emma Hart, Stefan Voß

PII: S0305-0548(19)30313-2
DOI: https://doi.org/10.1016/j.cor.2019.104871
Reference: CAOR 104871

To appear in: *Computers and Operations Research*

Received date: 30 July 2018
Revised date: 20 December 2019
Accepted date: 23 December 2019

Please cite this article as: Eduardo Segredo, Eduardo Lalla-Ruiz, Emma Hart, Stefan Voß, A similarity-based neighbourhood search for enhancing the balance exploration–exploitation of differential evolution, *Computers and Operations Research* (2019), doi: https://doi.org/10.1016/j.cor.2019.104871

Highlights

- A novel approach that promotes a balance between exploration and exploitation
- Adaptively promotes diversification and intensification based on the search progress
- Hybridisation of this method with both explorative and exploitative variants of DE
- The use of this approach with DE leads to better solutions on large-scale problems

A similarity-based neighbourhood search for enhancing the balance exploration–exploitation of differential evolution

Eduardo Segredo[a,b], Eduardo Lalla-Ruiz[d,*], Emma Hart[a], Stefan Voß[c]

[a] *School of Computing, Edinburgh Napier University, 10 Colinton Road, Edinburgh, EH10 5DT, Scotland, United Kingdom*
[b] *Departamento de Ingeniería Informática y de Sistemas, Universidad de La Laguna, San Cristóbal de La Laguna, Spain*
[c] *Institute of Information Systems, University of Hamburg, Hamburg, Germany*
[d] *Department of Industrial Engineering and Business Information Systems, University of Twente, Enschede, The Netherlands*

Abstract

The success of search-based optimisation algorithms depends on appropriately balancing exploration and exploitation mechanisms during the course of the search. We introduce a mechanism that can be used with *Differential Evolution* (DE) algorithms to adaptively manage the balance between the diversification and intensification phases, depending on current progress. The method—*Similarity-based Neighbourhood Search* (SNS)—uses information derived from measuring Euclidean distances among solutions in the decision space to adaptively influence the choice of neighbours to be used in creating a new solution. SNS is integrated into explorative and exploitative variants of JADE, one of the most frequently used adaptive DE approaches. Furthermore, SHADE, which is another state-of-the-art adaptive DE variant, is also considered to assess the performance of the novel SNS. A thorough experimental evaluation is conducted using a well-known set of large-scale continuous problems, revealing that incorporating SNS allows the performance of both explorative and exploitative variants of DE to be significantly improved for a wide range of the test-cases considered. The method is also shown to outperform variants of DE that are hybridised with a recently proposed global search procedure, designed to speed up the convergence of that algorithm.

Keywords: Differential evolution, global search, diversity management, exploration, exploitation, large-scale continuous optimization

1. Introduction

Numerous problems arising from the real-world can be modelled as optimisation problems. Meta-heuristic approaches often provide an appropriate compromise between computational effort and solution quality, with a broad variety of methods available, depending on the nature of the problem domain. In terms of tackling problems within the field of *continuous optimisation*, one of the most frequently used approaches is *Differential Evolution* (DE), first proposed by Storn & Price (1997), and since then spawning a wealth of variations, described in a recent survey by Das et al. (2016).

As with any meta-heuristic approach, there is a natural tension between increasing convergence speed—to produce results faster—and preventing premature convergence—reducing the quality of results. While exploitation methods favour the former, exploration methods favour the latter. As a result, a significant volume of research within the DE community has been devoted not only to introducing novel operators, e.g. Guo et al. (2017), that can be hybridised with DE, but also to developing schemes that adaptively balance exploration and exploitation mechanisms (Črepinšek et al., 2013; Lozano & García-Martínez, 2010). In this paper, we propose a novel operator to achieve this, namely *Similarity-based Neighbourhood Search* (SNS). It

*Corresponding author
Email addresses: e.segredo@napier.ac.uk (Eduardo Segredo), e.a.twente@utwente.nl (Eduardo Lalla-Ruiz), e.hart@napier.ac.uk (Emma Hart), stefan.voss@uni-hamburg.de (Stefan Voß)

is integrated within one of the most commonly used DE frameworks JADE, as well as with recent ones, such as SHADE. This continues the line of development proposed by Neri & Tirronen (2010) in a classification of approaches to DE, i.e. *"DE integrating an extra component"* in order to complementing DE with improvement methods to enhance its performance.

The method is evaluated on a well-known set of scalable continuous optimisation problems proposed by Li et al. (2013). The requirement to solve large, complex problems with a vast number of decision variables is becoming increasingly important in the era of *big data*, where typical problem instances might have several thousands, even millions, of continuous variables; efforts to develop new methods that can cope with this kind of scale are increasingly apparent in recent literature (LaTorre et al., 2015; Mahdavi et al., 2015). The aforementioned set of problems was provided for the special session and competition on *Large Scale Global Optimisation* organised in the field of the *Congress on Evolutionary Computation* (CEC) 2013 and used in the most recent one CEC'19.

Taking into account previous editions of the competition, in general terms, the approaches showing the best performance make use of multiple algorithms to solve an instance where DE is incorporated among others. In the current work, the main goal is to research and enhance DE state-of-the-art approaches by means of improvement methods that could be later considered in complex and hybrid schemes as those proposed in the literature. Thus, following the research direction (Guo et al., 2017; Črepinšek et al., 2013; Lozano & García-Martínez, 2010) the integration of DE with an extra component promoting an adaptive balance between exploration and exploitation along the search is investigated.

Bearing the above discussion in mind, the main contributions of this paper are therefore as follows:

- A novel *Similarity-based Neighbourhood Search* (SNS) that promotes a suitable balance between exploration and exploitation, depending on the current stage of the search. SNS is based on calculating a similarity value (e.g. Euclidean distance) among individuals and using this to influence the selection of individuals to create new solutions. The method adaptively promotes diversification at early stages of the search, and intensification towards the later stages.

- Hybridisation of the operator with both explorative and exploitative variants of DE based on parameter adaptation mechanisms provided by one of the most widely applied adaptive DE approaches: JADE (Zhang & Sanderson, 2009). Furthermore, SHADE (Tanabe & Fukunaga, 2013) is also taken into account as another adaptive DE variant to evaluate the performance of the novel SNS. This way the contribution of our method to DE is contextualised.

- A broad empirical investigation combined with detailed statistical analysis that demonstrates the utility of hybridising SNS with both explorative and exploitative DE variants in order to attain better solutions on a test-suite consisting of scalable instances.

- Additional experiments that demonstrate that SNS is also able to outperform, in a significant number of cases, a state-of-the-art operator GS that was recently proposed by Guo et al. (2017) to increase the convergence speed of DE to better solutions when dealing with continuous problems.

The remainder of this paper is structured as follows. Section 2 goes over those works related to the contributions of this paper. Afterwards, Section 3 describes the algorithmic proposals applied in the current work, including the particular DE variants, as well as the novel SNS. Then, the experimental evaluation carried out and the discussion of the results obtained are given in Section 4. Finally, Section 5 presents the main conclusions and suggests several directions for further research.

2. Literature review

We consider recent surveys on DE and continuous optimisation (Das et al., 2016; LaTorre et al., 2015; Mahdavi et al., 2015). Moreover, in this literature review, we pay particular attention to research that falls into the category proposed in the classification from Neri & Tirronen (2010), i.e. *"DE integrating an extra component"*. Bearing the above in mind, in the following we concisely review those works within the

said category. Namely, it encompasses those algorithms using DE as an evolutionary framework that are supported by additional algorithmic components. We note that the particular extra components considered in this overview are those concerning the synergy between DE and other search procedures.

Several approaches can be found in the literature concerning the hybridisation of DE with other well-known meta-heuristics, such as *Particle Swarm Optimisation* (PSO), *Simulated Annealing* (SA), *Variable Neighbourhood Search* (VNS), and *Genetic Algorithms* (GAs), among others. According to Das et al. (2008), DE is often hybridised with PSO. In this regard, Xin et al. (2012) presented an overview and taxonomy concerning hybridisations between DE and PSO.

Ghasemi et al. (2016) presented four hybrid approaches based on DE and PSO for solving multi-area economic dispatch problems. They also proposed a hybrid sum-local search optimiser, where the crossover operator of DE considered the best so-called particle of a given local neighbourhood. The authors indicated that their approach presented an appropriate balance between its global search ability and convergence features.

Parouha & Das (2016) recently proposed a hybrid scheme based on DE and the memory concept of PSO for solving continuous optimisation problems. The concept of memory was borrowed from PSO and used during the trial generation strategy of DE in order to enable it to use information belonging to a previous generation in the new one. Results reported that their approach exhibited better performance in comparison to either DE and PSO run as independent optimisers and to the standard hybridisation between DE and PSO.

As previously mentioned, other recent approaches have covered the hybridisation of DE with GA, SA and VNS. Trivedi et al. (2015) proposed a hybrid approach based on DE and GA, which was termed as hGADE, in order to deal with the *Unit Commitment Scheduling* problem. Binary variables were optimised through the GA, while continuous variables were evolved by means of DE operators. The comparison against DE and GA runs independently showed that hGADE led to better results, since the latter was able to significantly outperform the former.

Guo et al. (2014) presented a hybrid algorithm combining DE and SA. It considered two populations with each one ruled by a different DE variant. SA was used to enhance the global search ability of DE during the selection of individuals from both populations, as well as during the updating of the parameter values of DE. The computational study revealed that the usage of SA improved the overall performance of DE.

Kovačević et al. (2014) introduced a hybrid method based on DE and VNS. The ruling idea behind this hybrid approach was the application of the neighbourhood variation with the aim of estimating the parameter values of the DE crossover operator. The hybrid scheme showed to provide a higher performance in comparison to other DE variants proposed in the literature.

Guo et al. (2017) presented a hybridisation between DE and a global neighbourhood search, which was initially proposed by Wang et al. (2013) for integrating with PSO. In Guo et al. (2017), the authors demonstrated that the use of their GS improved the performance of DE when addressing continuous problems with low dimensionalities. Based on the results reported, they claimed that the convergence speed of DE to better solutions was accelerated. Those results motivated us to further analyse the behaviour of that particular GS when tackling continuous problems with a much larger number of dimensions, given that this analysis was missing from (Guo et al., 2017).

With respect to algorithmic schemes proposed to adapt the parameters F and CR (described in the next section) of DE, Zhang & Sanderson (2009) proposed JADE, which is a well-known scheme that provides a parameter control strategy to determine and update the said parameters. Another approach, proposed by Wang et al. (2011) combines three trial vector generation strategies and parameter control settings in the scheme termed *Composite* DE (CODE). Finally, Tanabe & Fukunaga (2013) proposed an improved variant of JADE, termed as *Success History-based Differential Evolution* (SHADE) that uses historical information for setting F and CR. The authors showed that SHADE performs better than JADE and CODE.

3. Algorithmic approaches

In this section, we briefly describe both the explorative and exploitative DE variants we have selected as the base algorithms that our novel SNS procedure will be integrated with. The DE versions are depicted

Algorithm 1 Pseudocode of differential evolution

Require: n, F, CR

1: Generate n individuals or target vectors as the initial population through an initialisation strategy. In this case, *Opposition-based Learning* (OBL) is considered
2: **while** (stopping criterion is not satisfied) **do**
3: **for** $(j = 1 : n)$ **do**
4: The individual $\vec{X}_j$ belonging to the current population is referred to as the target vector
5: Obtain a mutant vector $\vec{V}_j$ through the mutant generation strategy
6: Combine $\vec{X}_j$ and $\vec{V}_j$ through the crossover operator to get the trial vector $\vec{U}_j$
7: Select the fittest individual between $\vec{X}_j$ and $\vec{U}_j$ as the survivor for next generation
8: **end for**
9: Apply the novel SNS to the surviving population.
10: **end while**
11: **return** the fittest individual in the population

in Section 3.1, while SNS itself is introduced in Section 3.2. Both DE variants make use of the parameter adaptation mechanisms provided by JADE, which are described in Section 3.1.3.

3.1. Explorative and exploitative differential evolution with parameter adaptation based on JADE

In DE, a vector $\vec{X} = [x_1, \ldots, x_i, \ldots, x_D]$ is used to encode an individual. The i-th decision variable is represented by x_i, and the number of decision variables or dimensions of the problem at hand is given by D. At the same time, when dealing with box-constrained problems, the feasible region is defined by $\Phi = \{X \in \mathbb{R}^D | x_i \in [a_i, b_i],\ i = 1, 2, ..., D\}$, where the lower and upper bounds of variable x_i are given by a_i and b_i, respectively.

Using the most frequently used nomenclature for DE (Storn & Price, 1997), i.e., DE*/x/y/z*, where x is the individual to be mutated, y defines the number of difference vectors used, and z indicates the crossover strategy, we selected the variants DE*/rand/1/bin* and DE*/current-to-pbest/1/bin*: these intrinsically promote exploration and exploitation, respectively (Segura et al., 2015; Zhang & Sanderson, 2009). The term *bin* refers to *binomial crossover*, which is described in the next section.

3.1.1. An explorative differential evolution variant: DE*/rand/1/bin*

The choice of this particular DE variant is due to two main reasons. First, in past research, a configuration of DE/rand/1/bin provided the best performance for a significant number of functions belonging to the test suite we tackle here (Kazimipour et al., 2014). Second, it was shown to be the best performing overall DE version when dealing with a set of scalable continuous problems in previous work (Segura et al., 2015).

Algorithm 1 shows the general operation of DE. First of all, n individuals are generated by means of an initialisation strategy (step 1). In this work, we apply *Opposition-based Learning* (OBL), proposed by Xu et al. (2014), as the initialisation mechanism to enhance the quality of the initial population. In a previous work carried out by the authors (Segredo et al., 2017), it was demonstrated that the combination of DE/rand/1/bin together with OBL is likely to provide better solutions, in comparison to the solutions attained by applying other initialisation schemes, for the set of problems considered herein. Once the initial population is obtained, it is evolved until a given stopping criterion is satisfied (step 2). At each generation, the following steps are carried out for each individual $\vec{X}_{j=1\ldots n}$ belonging to the current population (step 3), denoted as *target vector* in DE terminology (step 4).

First, the *mutant generation strategy* is applied in order to produce a *mutant vector* $\vec{V}_j$ (step 5). This particular DE version applies the mutant generation strategy rand/1. Equation 1 describes that strategy, where r_1, r_2, and r_3 are mutually exclusive integers chosen at random from the range $[1, n]$, and also different to index j. Since all individuals involved in the mutant generation strategy are randomly selected, it promotes exploration rather than exploitation. Nevertheless, by means of the parameter F, which refers to the *mutation scale factor*, the diversification and intensification abilities of the algorithm can be balanced.

Large values of F promote more exploration, while small values turn the approach into a more exploitative scheme.

$$\vec{V}_j = \vec{X}_{r_3} + F \times (\vec{X}_{r_1} - \vec{X}_{r_2}) \tag{1}$$

Once the mutant vector is obtained, it is combined with the target vector through the application of a crossover operator so as to obtain the *trial vector* $\vec{U}_j$ (step 6). The combination of the mutant vector generation strategy and the crossover operator is usually referred to as the *trial vector generation strategy*. For this work, the binomial crossover, which is one of the most widely applied DE crossover methods, was selected. Its operation is shown in Equation 2. The decision variable i belonging to individual $\vec{X}_j$ is represented by $x_{j,i}$. A random number uniformly distributed in the range $[0, 1]$ is given by $rand_{j,i}$, and $i_{rand} \in [1, 2, ..., D]$ is an index selected at random ensuring that at least one decision variable belonging to the mutant vector is inherited by the trial one. Hence, variables are inherited from the mutant vector with probability CR, also denoted as the *crossover rate*. In the remaining cases, variables are inherited from the target vector.

$$u_{j,i} = \begin{cases} v_{j,i} & \text{if } rand_{j,i} \leq CR \text{ or } i = i_{rand} \\ x_{j,i} & \text{otherwise} \end{cases} \tag{2}$$

The trial vector generation strategy might produce individuals outside the feasible region Φ, as it can be observed in Equations 1 and 2. To address this issue, an infeasible value in a given variable is randomly re-initialised in the corresponding feasible range of that variable. Once the trial vector is obtained, it is compared against its corresponding target vector in terms of the objective function value. The fittest individual survives for the next generation (step 7). In our approach, the trial vector survives in case of a tie. Finally, the novel SNS operator, which will be introduced in Section 3.2, is applied to the surviving population at step 9.

3.1.2. An exploitative differential evolution variant: DE*/current-to-pbest/1/bin*

This DE variant is considered due to its ability to promote intensification rather than diversification. Particularly, it is the DE variant considered by the original implementation of JADE (Zhang & Sanderson, 2009). The operation of this DE variant (DE/current-to-pbest/1/bin) is exactly the same as that shown in Algorithm 1. The mutant generation strategy, however, is different.

Here, a mutant vector $\vec{V}_j$ is created starting from a target vector $\vec{X}_j$ as it is described in Equation 3. Indexes r_1 and r_2 are mutually exclusive integers randomly selected from the range $[1, n]$, and also different to index j. Furthermore, the individual $\vec{X}_{r_3}$ is randomly selected from the fittest $p \times 100\%$ individuals. Some of the fittest individuals in the population are taken into account by the mutant generation scheme, and consequently, this DE variant is more exploitative than the approach DE/rand/1/bin, which only uses randomness for selecting the individuals involved in the mutant generation scheme.

$$\vec{V}_j = \vec{X}_j + K \times (\vec{X}_{r_3} - \vec{X}_j) + F \times (\vec{X}_{r_1} - \vec{X}_{r_2}) \tag{3}$$

As can be observed, in addition to the mutation scale factor F, parameter p can be used in order to set the balance between the exploration and exploitation capabilities of the algorithm. By considering large p values, the scheme is more explorative, while it becomes more exploitative with small p values. Finally, parameter K is also introduced, but in order to make the configuration of the approach easier, $K = F$ is usually considered in the related literature (Segura et al., 2015; Zhang & Sanderson, 2009).

3.1.3. Adaptation of the mutation scale factor and crossover rate by means of JADE

As observed in previous sections, values for the mutation scale factor F and the crossover rate CR have to be set to run both aforementioned DE variants. *Controlling* or adapting the parameters of an algorithm while it is run has shown to provide significant benefits with respect to *tuning* or keeping those parameters fixed for the whole execution (Karafotias et al., 2015). Therefore, a significant number of works related to the adaptation of DE parameters have been proposed (Das et al., 2016; Tvrdík et al., 2013).

JADE (Zhang & Sanderson, 2009) includes one of the best performing and most frequently used approaches to adapt the mutation scale factor F and the crossover rate CR. Those control mechanisms produce values for F and CR before executing the trial vector generation strategy (steps 5 and 6 of Algorithm 1), thus generating a new trial vector by using the newly created values. Hence, every individual has associated its own values for parameters F and CR.

In JADE, a particular value for F is randomly obtained by means of a *Cauchy* distribution with location factor μ_F and scale parameter equal to 0.1. If that value is lower than 0, then another one is sampled from the distribution, while if it is greater than 1, then it is truncated to 1. The location factor μ_F is initialised to 0.5, and then, its value is updated at each generation after step 8 of Algorithm 1. In order to do this, the *Lehmer mean* ($mean_L$) of the successful values of F (S_F), the previous value of μ_F, and a parameter c representing the adaptation speed of μ_F are taken into consideration. The set S_F consists of those values of F associated to trial vectors that have been able to replace their corresponding target vectors in the population to survive for the next generation (step 7 of Algorithm 1). Equation 4 illustrates the updating mechanism of μ_F.

$$\mu_F = (1 - c) \cdot \mu_F + c \cdot mean_L(S_F) \tag{4}$$

At this point, we should note that in previous research (Segura et al., 2015), it was demonstrated that the application of Equation 4 decreases the performance of an explorative DE version, such as DE/rand/1/bin, in comparison to keeping μ_F fixed for the whole run. In the same work, however, it was shown that the application of Equation 4 increases the performance of an exploitative DE variant, like DE/current-to-pbest/1/bin. As a result, the updating mechanism of μ_F was disabled for DE/rand/1/bin herein, and values for parameter F were randomly generated by a *Cauchy* distribution by keeping the location factor fixed ($\mu_F = 0.5$) for the whole run. In the case of DE/current-to-pbest/1/bin, the updating mechanism of μ_F was applied.

With respect to the control mechanism of CR, it is similar to the control approach of F. In this case, a value for CR is randomly generated through a *Normal* distribution with mean μ_{CR} and standard deviation equal to 0.1, and then truncated to the range $[0, 1]$. The mean μ_{CR} is initialised to 0.5 and updated by considering the arithmetic mean ($mean_A$) of the successful values of CR (S_{CR}), the previous value of μ_{CR}, and a parameter c that represents the adaptation speed of μ_{CR}. In the current work, the updating mechanism of μ_{CR}, which is shown in Equation 5, is applied to both DE variants with an adaptation speed $c = 0.1$.

$$\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot mean_A(S_{CR}) \tag{5}$$

3.2. Similarity-based neighbourhood search

In order to induce a proper balance between the diversification and intensification abilities of both aforementioned DE variants, and at the same time, with the aim of improving the quality of the solutions provided at the end of the executions, a novel Similarity-based Neighbourhood Search (SNS) is presented.

This method considers the similarity among individuals, thus, first of all, a given similarity metric has to be established, such as the Euclidean distance. Once that metric is selected, the population is sorted in terms of the similarity of its individuals with respect to the fittest one, thus producing a sorted list. A portion of that list is chosen according to a given criterion, which, for instance, can consider the current moment of the search procedure. As a result, individuals involved in the neighbourhood search are selected from that particular portion of the list, which dynamically changes depending on the current moment of the search. In this work, the similarity metric applied is the Euclidean distance and a portion of the sorted list is selected based on the number of function evaluations currently performed. The application of the above strategy seeks to promote diversification at early stages of the optimisation process, while intensification is fostered at the end of the runs. In the following, the specific details of SNS are provided.

The operation of SNS is shown in Algorithm 2. First of all, a real number a_1 is uniformly selected at random from the range $[0, 1]$, together with defining a_2 such that the condition $a_1 + a_2 = 1$ holds (step 1). Afterwards, an individual $\vec{X}_k$ is uniformly selected at random from the current population (step 2). Then,

Algorithm 2 Pseudocode of the similarity-based neighbourhood search

Require: n, δ, Ω, ω

1: Set a_1 to a random real number uniformly selected from the range $[0, 1]$, together with defining a_2 such that the condition $a_1 + a_2 = 1$ is satisfied
2: Uniformly select an individual $\vec{X}_k$ from the current population at random
3: Sort the current population in descending order in terms of the similarity of each individual, i.e., the Euclidean distance in the decision space, with respect to the fittest individual in the population $\vec{X}_{best}$
4: Create a sub-population including those individuals indexed within the $[l(\omega), u(\omega)]$ positions in the sorted population and select another individual $\vec{X}_{r_1}$ at random from that limited population such that $r_1 \in [l(\omega), u(\omega)]$. Index r_1 must be different to index k
5: Generate a new individual $\vec{V}$ by means of Equation 6
6: Replace the best individual's least similar neighbour by the newly created individual $\vec{V}$

the current population is sorted in descending order in terms of the similarity of each individual with respect to the fittest individual in the population, i.e. $\vec{X}_{best}$ (step 3). The above means that the fittest individual's least similar individuals will be found at the beginning of the list, while the fittest individual's most similar individuals will be found at the end of the list. The particular similarity metric to be applied has to be established by the algorithm designer. Here, we use the Euclidean distance in the decision space. In step 4, a sub-population composed of those individuals indexed in the $[l(\omega), u(\omega)]$ positions of the sorted population is used for selecting at random another individual $\vec{X}_{r_1}$. The computation of $l(\omega)$ and $u(\omega)$ will be described in detail later.

After that, Equation 6 is applied to produce a new individual $\vec{V}$ (step 5). It can be observed that Equation 6 allows individual $\vec{X}_k$ to be attracted by $\vec{X}_{best}$ and $\vec{X}_{r_1}$, depending on the values that a_1 and a_2 take. The idea behind Equation 6 is that SNS promotes exploration or exploitation depending on the particular individual chosen as $\vec{X}_{r_1}$. If $\vec{X}_{r_1}$ is different to $\vec{X}_{best}$, then SNS will promote exploration. Otherwise, if $\vec{X}_{r_1}$ is similar to $\vec{X}_{best}$ then SNS will promote exploitation.

$$\vec{V} = \vec{X}_k + a_1 \times (\vec{X}_{Best} - \vec{X}_k) + a_2 \times (\vec{X}_{r_1} - \vec{X}_k) \tag{6}$$

Finally, the newly generated individual $\vec{V}$ replaces the best individual's least similar neighbour in the population (step 6). A number of different replacement strategies were tested in preliminary experimentation: replacement of the fittest individual's least similar neighbour; replacement of the fittest individual's most similar neighbour; replacement of individual $\vec{X}_k$ only in the case the newly generated individual $\vec{V}$ is fitter than the former. The first replacement strategy provided the best overall results in these preliminary experiments and therefore is used in the remainder of the paper.

The method by which individual $\vec{X}_{r_1}$ is selected from the sorted population (step 4) is described below. Index r_1, which must be different to index k, is uniformly chosen at random from the range $[l(\omega), u(\omega)]$. Functions $l(\omega)$ and $u(\omega)$ set a lower and an upper bound, respectively, for the range from which $\vec{X}_{r_1}$ is selected, and depend on the current stage of the search, given by the number of function evaluations ω performed until that particular moment. The linear ascending function shown in Equation 7 is applied to calculate $u(\omega)$, where n is the population size, the total number of function evaluations of a run is given by Ω, and parameter $\delta < n$ refers to the minimum number of individuals involved in the selection. Once a particular value is given by $u(\omega)$, the lower bound $l(\omega)$ is calculated as Equation 8 shows.

As a result, at the beginning of a particular run, when only a few function evaluations have been performed, the lower and upper bounds will be close to 0 and δ, respectively. As the execution progresses, both bounds will linearly increase. Finally, at the end of the run, the lower and upper bounds will be close to $n - \delta$ and n, respectively.

$$u(\omega) = \frac{n - \delta}{\Omega} \cdot \omega + \delta \tag{7}$$

Table 1: Overview of experiments. Considering a particular experiment, bullet points in the last column indicate the best-performing overall approaches from among those specified in the corresponding second column.

Experiment	Methods	Goal	Overall best		
			SNS	GS	DE
First	DE-RAND-SNS DE-RAND-GS DE-RAND	Analysing the performance of the proposed SNS when it is embedded into the explorative DE-RAND	•		
Second	DE-CURR-SNS DE-CURR-GS DE-CURR	Analysing the performance of the proposed SNS when it is embedded into the exploitative DE-CURR	•		
Third	DE-SHA-SNS DE-SHA-GS DE-SHA	Analysing the performance of the proposed SNS when it is embedded into SHADE	•	•	

$$l(\omega) = u(\omega) - \delta \tag{8}$$

Recall that the population from which $\vec{X}_{r_1}$ is selected is sorted in descending order in terms of the similarity of each individual with respect to the fittest individual in the population. As a result, at the beginning of a given run, $\vec{X}_{r_1}$ will be selected from among the δ least similar neighbours to the fittest individual in the current population. Exploration is thus promoted at early stages of the search procedure (see Equation 6). Nevertheless, as more and more function evaluations are performed, the fittest individual's least similar neighbours are progressively discarded, and therefore, the balance is moved from exploration towards exploitation. At the end of the execution, only the fittest individual's δ most similar neighbours are involved in the selection, and consequently, exploitation is promoted.

Finally, it is worth noting that for a fixed population size, parameter δ allows the balance between the exploration and exploitation abilities of SNS to be dynamically adjusted. With small values of δ, its intensification ability is increased at late stages of the optimisation process, while it is decreased considering large values.

4. Experimental evaluation

This section is devoted to describing the computational experiments performed to assess the performance of SNS. As previously discussed, SNS is combined with the two DE variants, DE/rand/1/bin and DE/current-to-pbest/1/bin described in Section 3.1; these are referred to as DE-RAND-SNS and DE-CURR-SNS, respectively. It is important to note that both DE versions are adaptive, as the control mechanisms provided by JADE are applied to adapt the values of the mutation scale factor F and the crossover rate CR (as described in Section 3.1.3). We also compare performance to the same DE variants with and without the global neighbourhood search operator (GS) proposed by Guo et al. (2017): the variants including GS are termed as DE-RAND-GS and DE-CURR-GS in the rest of the paper, while those without as DE-RAND and DE-CURR. Finally, DE-SHA-SNS and DE-SHA-GS refer to hybridisations of SHADE embedding SNS and GS, respectively, while DE-SHA refers to the original implementation of SHADE given by Tanabe & Fukunaga (2013). At this point, we would like to remind that the remaining components of all the different algorithms compared in each experiment were the same. For instance, the initialisation strategy OBL was applied by all the approaches included in the comparisons. An overview of the experiments carried out along this section, including a description of their goals, the particular approaches involved, and the schemes showing the best overall resulting performance, is given in Table 1.

Table 2: Benchmark functions

Name	Bounds	Optimum
f_1: Shifted Elliptic Function	$[-100, 100]^D$	0
f_2: Shifted Rastrigin's Function	$[-5, 5]^D$	0
f_3: Shifted Ackley's Function	$[-32, 32]^D$	0
f_4: 7-nonseparable, 1-separable Shifted and Rotated Elliptic Function	$[-100, 100]^D$	0
f_5: 7-nonseparable, 1-separable Shifted and Rotated Rastrigin's Function	$[-5, 5]^D$	0
f_6: 7-nonseparable, 1-separable Shifted and Rotated Ackley's Function	$[-32, 32]^D$	0
f_7: 7-nonseparable, 1-separable Shifted Schwefel's Function	$[-100, 100]^D$	0
f_8: 20-nonseparable Shifted and Rotated Elliptic Function	$[-100, 100]^D$	0
f_9: 20-nonseparable Shifted and Rotated Rastrigin's Function	$[-5, 5]^D$	0
f_{10}: 20-nonseparable Shifted and Rotated Ackley's Function	$[-32, 32]^D$	0
f_{11}: 20-nonseparable Shifted Schwefel's Function	$[-100, 100]^D$	0
f_{12}: Shifted Rosenbrock's Function	$[-100, 100]^D$	0
f_{13}: Shifted Schwefel's Function with Conforming Overlapping Subcomponents	$[-100, 100]^D$	0
f_{14}: Shifted Schwefel's Function with Conflicting Overlapping Subcomponents	$[-100, 100]^D$	0
f_{15}: Shifted Schwefel's Function	$[-100, 100]^D$	0

Experimental method. All the above algorithmic approaches were implemented by means of the *Metaheuristic-based Extensible Tool for Cooperative Optimisation* (METCO) proposed by León et al. (2009). Experiments were executed on one Debian GNU/Linux computer with four AMD® Opteron™ processors (model number 6348 HE) at 2.8 GHz and 64 GB RAM. Since all the approaches considered are stochastic, each run was repeated 100 times. The following statistical testing procedure, which was previously used in a former work by the authors (Segura et al., 2016), was applied to conduct comparisons between approaches. First, a *Shapiro-Wilk test* was performed to check whether the values of the results followed a normal (Gaussian) distribution. If so, the *Levene test* checked for the homogeneity of the variances. If the samples had equal variance, an ANOVA *test* was done. Otherwise, a *Welch test* was performed. For non-Gaussian distributions, the non-parametric *Kruskal-Wallis* test was used. For all tests, a significance level $\alpha = 0.05$ was considered.

Problem set. We test the proposed algorithms using the continuous optimisation benchmark suite presented by Li et al. (2013). It consists of 15 different scalable minimisation functions (f_1–f_{15}) as follows: fully-separable functions (f_1–f_3), partially additively separable functions (f_4–f_{11}), overlapping functions (f_{12}–f_{14}), and a non-separable function (f_{15}). As proposed by Li et al. (2013), we fix the number of decision variables D to 1000 for all functions, with the exception of f_{13} and f_{14}, where 905 decision variables were considered due to overlapping subcomponents. Large-scale optimisation problems are thus considered herein. Table 2 shows a summary of the functions tested in the current work, including information about the bounds of the decision variables and the value of the global optimum for each of them. As it can be observed, all the test cases are based on transformations and/or combinations of well-known base functions, such as the *Sphere* function and the *Rastrigin's* function, among others. For instance, Equation 9 shows the formal definition of the *Rastrigin's* function, where $\vec{x}$ is a vector with D decision variables or dimensions. The goal is to find the values of the D decision variables belonging to vector $\vec{x}$ such that $f_{rastrigin}(\vec{x})$ is minimised.

$$f_{rastrigin}(\vec{x}) = \sum_{i=1}^{D} [x_i^2 - 10cos(2\pi x_i) + 10] \tag{9}$$

Table 3: Parameterisation of DE-RAND-SNS, DE-RAND-GS and DE-RAND

Parameter	Value	Parameter	Value
Stopping criterion	$3 \cdot 10^6$ evals.	Mutation scale factor (F)	Adapted by Cauchy(0.5, 0.1)
Population size (n)	50	Crossover rate (CR)	Adapted by JADE

4.1. Analysing the performance of the similarity-based neighbourhood search with an explorative adaptive DE version: DE/rand/1/bin

Experiments in this section address two questions: (1) Does SNS enable an appropriate balance between the diversification and intensification abilities of an *explorative* DE algorithm?; (2) Does the hybrid approach DE-RAND-SNS provide better solutions in comparison to DE-RAND-GS and/or DE-RAND? DE-RAND-SNS, DE-RAND-GS, and DE-RAND were applied with the parameterisation shown in Table 3. A stopping criterion equal to $3 \cdot 10^6$ function evaluations was set for all the approaches by following the suggestions given by Li et al. (2013).

In order to fix the population size n, we carried out a preliminary study where we executed 20 runs of DE-RAND by considering 15, 50, 150 and 300 individuals to solve functions f_1–f_{15}. The best overall performance in our preliminary study was attained by applying $n = 50$ individuals. As a result, all experiments with DE-RAND-SNS, DE-RAND-GS and DE-RAND were conducted using that population size. Finally, the minimum number of individuals involved in the selection process of SNS was set to five individuals ($\delta = 5$), which represents 10% of the whole population. This value was selected as in a preliminary study it provided the best overall results in terms of the quality of the solutions attained at the end of the executions. Particularly, we executed 20 independent runs of DE-RAND-SNS with problems f_1–f_{15} by considering values 5, 10, 15, 20, 25 and 50 for parameter δ. Since parameter δ is fixed to a relatively small value, exploitation is increased by SNS at late stages of the search process, as we previously mentioned in Section 3.2.

Figure 1 shows, for each of the three approaches DE-RAND-SNS, DE-RAND-GS and DE-RAND, the evolution of the mean of the error with respect to the objective function value considering 100 independent runs. Note that for some test cases (f_1, f_3 and f_{12}), axes were modified in order to properly visualise differences among approaches. Furthermore, in the particular case of f_{12}, axes were adjusted to show differences between DE-RAND-GS and DE-RAND, thus discarding the results of DE-RAND-SNS, since the latter attained a worse performance in comparison to the first two approaches. DE-RAND-SNS was able to provide the lowest mean of the error during the whole search process on 9 out of 15 functions. To understand the role that diversity might play in contributing to these results, we examine three example functions (f_3, f_6 and f_{10}) in more detail. Those three functions were selected as they provide a representative set, i.e., similar conclusions than those given below can be extracted for the remaining test cases. Figure 2 describes the evolution of the mean distance to the closest neighbour (DCN) attained by DE-RAND-SNS, DE-RAND-GS and DE-RAND. Note that although DE-RAND-GS and DE-RAND preserve a higher diversity in the population during the execution in comparison to DE-RAND-SNS, they have a higher mean error than DE-RAND-SNS. In other words, the tendency of the DE variant used to promote *exploration* is not suppressed by DE-RAND-GS or DE-RAND. On the other hand, the adaptive mechanism induced by DE-RAND-SNS appears to counter-balance the explorative tendency of the base-variant to provide better results. In general, DE-RAND-SNS tends to increase diversity at the beginning of a run; as executions advance, diversity is then decreased.

In some instances, e.g. f_{11}, DE-RAND-SNS does not converge as fast as DE-RAND-GS and/or DE-RAND during the early stages of the search process, but achieves the lowest mean of the error by the end of the execution. This is explained by the fact that the novel SNS operator shifts the balance from exploration towards exploitation as the run progresses, which ultimately delivers better results than the variants that consistently promote exploration. Finally, although DE-RAND-SNS exhibited the fastest convergence to better solutions in the majority of test cases in comparison to DE-RAND-GS and DE-RAND, there are six functions for which DE-RAND-GS and DE-RAND showed a better performance with respect to DE-RAND-SNS. It is likely that exploration should be promoted during the whole run in order to better deal with those test cases. In fact, four out of those six test cases (i.e., f_2, f_5, f_9 and f_{12}) are multimodal problems, where approaches that mainly promote exploration may attain better results. Consequently, an approach like DE-RAND-SNS, which moves the balance towards intensification as the run progresses, might be counterproductive when solving those particular functions when compared to schemes that mainly promote exploration during the entire run, such as DE-RAND-GS and DE-RAND.

Table 4 shows the mean, the median and the standard deviation (SD) of the error attained by DE-RAND-SNS, DE-RAND-GS and DE-RAND on each problem instance at the end of each execution. The best results obtained are shown in **boldface**. DE-RAND-SNS provides the lowest mean and median of the error at the

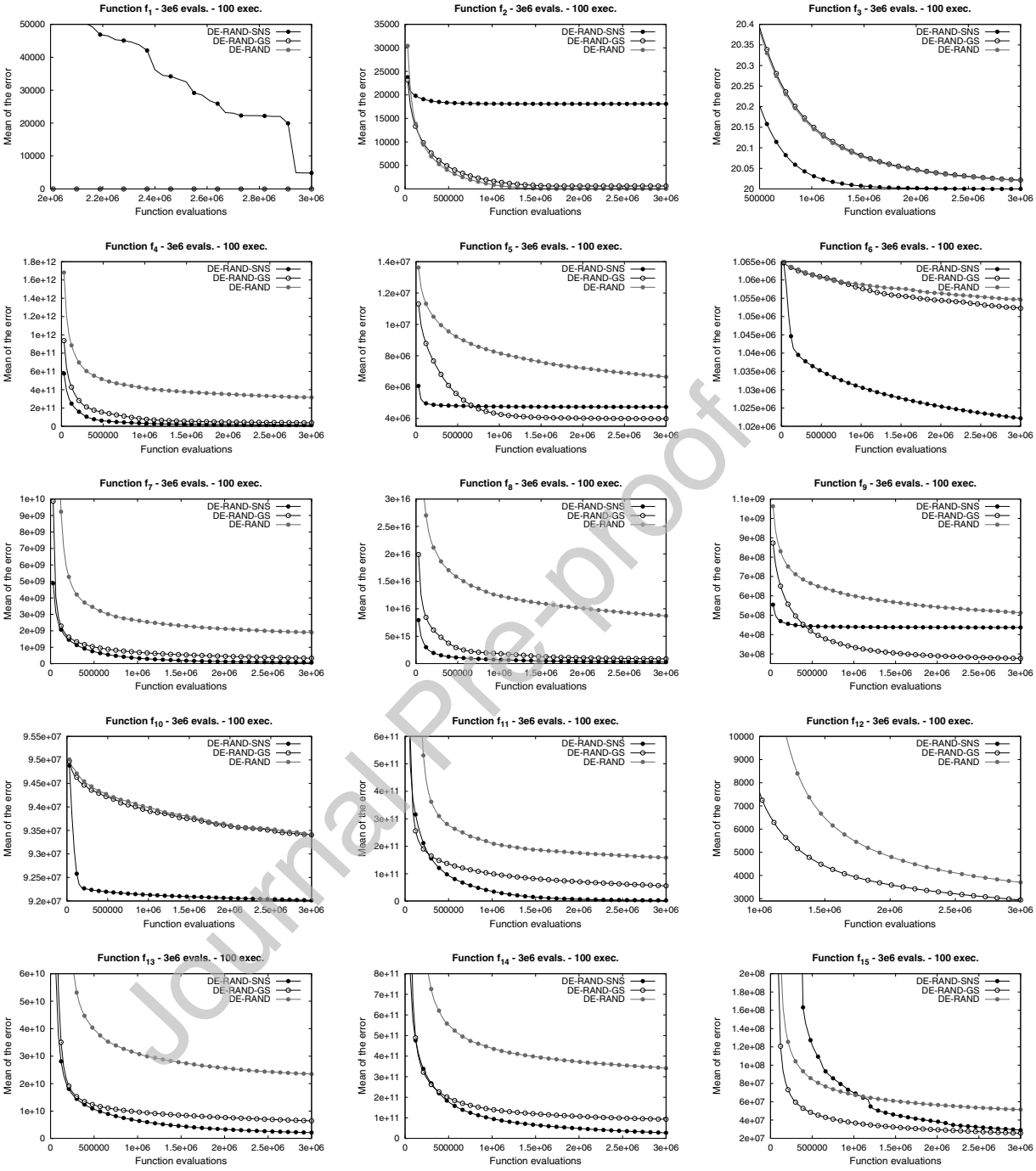

Figure 1: Evolution of the mean of the error for schemes DE-RAND-SNS, DE-RAND-GS, and DE-RAND considering 100 executions

end of the executions in 9 out of 15 functions (f_3, f_4, f_6–f_8, f_{10}, f_{11}, f_{13} and f_{14}). In f_{15}, DE-RAND-SNS gives the best median, while DE-RAND-GS provides the best mean. DE-RAND-GS obtains the best mean and median on four instances (f_1, f_5, f_9 and f_{12}), and DE-RAND on one problem (f_2).

A pairwise statistical comparison among the different optimisation schemes is presented in Table 5,

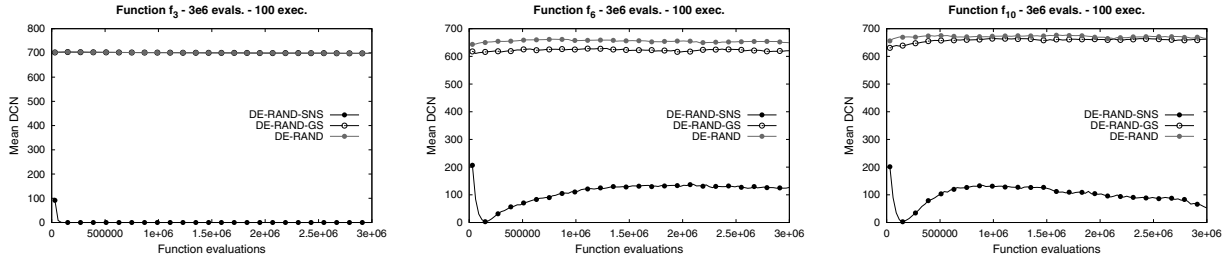

Figure 2: Evolution of the mean distance to the closest neighbour (DCN) for schemes DE-RAND-SNS, DE-RAND-GS, and DE-RAND considering 100 executions

following the statistical procedure described at the beginning of Section 4. In particular, p-values and results of the statistical comparison between the first and second approaches of each pair are depicted. In cases where statistically significant differences appeared, p-values are shown in **boldface**. Moreover, the table also shows whether the first approach statistically outperformed the second one (↑), if the first scheme was statistically outperformed by the second one (↓), and if statistically significant differences did not arise between both approaches (↔). A configuration A statistically outperforms another configuration B if there exists statistically significant differences between them, i.e., if the p-value is lower than $\alpha = 0.05$, and if at the same time, A provides a lower mean and median of the error than B. For those cases where approach A attained the lowest mean of the error, while configuration B achieved the lowest median of the error, and vice-versa, the *Vargha-Delaney A measure* was considered in order to check for effect size, and therefore, to determine the best performing scheme. We provide the following observations based on the data explained above.

- DE-RAND-SNS statistically outperformed DE-RAND-GS in **10 out of 15 test cases** (f_3, f_4, f_6–f_8, f_{10}, f_{11} and f_{13}–f_{15}), while DE-RAND-GS outperformed DE-RAND-SNS in the remaining **five problems**.
- DE-RAND-SNS was statistically better than DE-RAND on **12 out of 15 problems** (f_3–f_{11} and f_{13}–f_{15}), while DE-RAND statistically outperformed DE-RAND-SNS in the remaining **three test cases**.
- DE-RAND-GS statistically outperformed DE-RAND in **11 out of 15 functions** (f_1, f_4, f_5, f_7–f_9, and f_{11}–f_{15}), while DE-RAND was statistically better than DE-RAND-GS in **two test cases** (f_2 and f_3). Significant differences were not observed for the remaining problems.

We conclude that on the scalable optimisation problems tested, it is worth hybridising the explorative adaptive DE variant with an additional operator, whether it is the novel SNS or the global neighbourhood search operator GS. This is evidenced by the fact that DE-RAND-SNS and DE-RAND-GS statistically outperformed DE-RAND in 12 and 11 problems, respectively.

However, DE-RAND-SNS performed statistically better than DE-RAND-GS for a significant number of problems (10 out of 15), demonstrating its clear superiority over the recently proposed global search operator. This is most likely attributed to the tendency of the global neighbourhood to lead to premature convergence, for example as in test cases f_4 and f_{11}, while DE-RAND-SNS is able to maintain a better balance between exploration and exploitation during different phases of the algorithm.

4.2. Analysing the performance of the similarity-based neighbourhood search with an exploitative *adaptive* DE *version:* DE*/current-to-pbest/1/bin*

Next, we repeat the above study with the goal of analysing whether SNS is able to induce a suitable balance between the diversification and intensification abilities of a DE variant which mainly promotes *exploitation* on the same suite of problem instances. We compare the performances of the three algorithms DE-CURR-SNS, DE-CURR-GS and DE-CURR, all of which utilise an exploitative version of DE, using the parameterisation shown in Table 6. The population size n was fixed to 300 individuals, following a preliminary analysis that

Table 4: Mean, median, and standard deviation (SD) of the error achieved by DE-RAND-SNS, DE-RAND-GS, and DE-RAND at the end of 100 executions for problems f_1–f_{15}

Alg.	DE-RAND-SNS			DE-RAND-GS		
Func.	Mean	Median	SD	Mean	Median	SD
f_1	4.846e+03	6.333e+00	3.005e+04	**1.480e-12**	**1.473e-12**	2.870e-13
f_2	1.809e+04	1.807e+04	9.847e+02	6.265e+02	6.019e+02	2.069e+02
f_3	**2.000e+01**	**2.000e+01**	1.467e-04	2.002e+01	2.002e+01	8.052e-04
f_4	**1.260e+10**	**1.205e+10**	4.173e+09	3.808e+10	3.552e+10	1.036e+10
f_5	4.734e+06	4.755e+06	8.471e+05	**3.984e+06**	**3.986e+06**	6.934e+05
f_6	**1.022e+06**	**1.022e+06**	1.073e+04	1.052e+06	1.056e+06	1.267e+04
f_7	**7.740e+07**	**6.792e+07**	3.235e+07	3.383e+08	3.166e+08	1.231e+08
f_8	**2.942e+14**	**3.152e+14**	1.035e+14	8.792e+14	9.236e+14	3.571e+14
f_9	4.366e+08	4.362e+08	5.288e+07	**2.770e+08**	**2.759e+08**	4.425e+07
f_{10}	**9.201e+07**	**9.206e+07**	7.459e+05	9.341e+07	9.354e+07	6.541e+05
f_{11}	**2.968e+09**	**1.388e+09**	5.362e+09	5.596e+10	5.430e+10	1.718e+10
f_{12}	8.466e+05	6.546e+03	6.791e+06	**2.944e+03**	**2.932e+03**	2.963e+02
f_{13}	**2.173e+09**	**2.039e+09**	6.081e+08	6.455e+09	6.367e+09	9.688e+08
f_{14}	**2.768e+10**	**2.637e+10**	1.198e+10	9.336e+10	9.153e+10	1.247e+10
f_{15}	2.892e+07	**2.083e+07**	3.930e+07	**2.563e+07**	2.516e+07	3.415e+06

Alg.	DE-RAND		
Func.	Mean	Median	SD
f_1	4.292e-12	4.296e-12	5.516e-13
f_2	**1.224e+00**	**9.950e-01**	1.272e+00
f_3	2.002e+01	2.002e+01	6.832e-04
f_4	3.156e+11	3.323e+11	1.140e+11
f_5	6.652e+06	6.678e+06	5.782e+05
f_6	1.055e+06	1.056e+06	9.660e+03
f_7	1.902e+09	1.937e+09	3.679e+08
f_8	8.701e+15	8.253e+15	2.803e+15
f_9	5.128e+08	5.165e+08	4.130e+07
f_{10}	9.341e+07	9.351e+07	5.985e+05
f_{11}	1.587e+11	1.510e+11	4.208e+10
f_{12}	3.702e+03	3.708e+03	1.248e+02
f_{13}	2.348e+10	2.395e+10	3.217e+09
f_{14}	3.413e+11	3.427e+11	5.314e+10
f_{15}	5.131e+07	5.158e+07	3.447e+06

Table 5: Pairwise statistical comparison among DE-RAND-SNS, DE-RAND-GS, and DE-RAND considering their results achieved at the end of 100 executions for problems f_1–f_{15}

	DE-RAND-SNS vs. DE-RAND-GS		DE-RAND-SNS vs. DE-RAND		DE-RAND-GS vs. DE-RAND	
Func.	p-value	Stat.	p-value	Stat.	p-value	Stat.
f_1	**2.524e-34**	↓	**2.524e-34**	↓	**6.847e-89**	↑
f_2	**2.524e-34**	↓	**2.524e-34**	↓	**2.524e-34**	↓
f_3	**2.524e-34**	↑	**2.524e-34**	↑	**1.083e-13**	↓
f_4	**2.100e-33**	↑	**2.524e-34**	↑	**1.561e-33**	↑
f_5	**9.349e-11**	↓	**1.469e-43**	↑	**1.706e-74**	↑
f_6	**2.079e-26**	↑	**5.329e-30**	↑	2.083e-01	↔
f_7	**6.018e-34**	↑	**2.524e-34**	↑	**2.524e-34**	↑
f_8	**1.496e-29**	↑	**1.526e-51**	↑	**2.204e-32**	↑
f_9	**1.899e-57**	↓	**4.757e-23**	↑	**9.301e-95**	↑
f_{10}	**9.187e-25**	↑	**2.370e-26**	↑	6.199e-01	↔
f_{11}	**5.030e-34**	↑	**2.524e-34**	↑	**4.524e-47**	↑
f_{12}	**2.524e-34**	↓	**2.524e-34**	↓	**1.545e-30**	↑
f_{13}	**2.601e-34**	↑	**2.524e-34**	↑	**2.294e-81**	↑
f_{14}	**3.209e-34**	↑	**2.524e-34**	↑	**4.931e-73**	↑
f_{15}	**3.497e-07**	↑	**2.496e-30**	↑	**2.524e-34**	↑

Table 6: Parameterisation of DE-CURR-SNS, DE-CURR-GS and DE-CURR

Parameter	Value	Parameter	Value
Stopping criterion	$3 \cdot 10^6$ evals.	Mutation scale factor (F)	Adapted by JADE
Population size (n)	300	Crossover rate (CR)	Adapted by JADE

indicated that this value provided the best overall performance for problems f_1–f_{15}. Given that this DE variant promotes exploitation, it makes sense that larger population sizes provide some means of exploration to balance this. The minimum number of individuals involved in the selection process of SNS was set to five individuals ($\delta = 5$), as in the first experiment.

Figure 3 shows the evolution of the mean of the error with respect to the objective function value

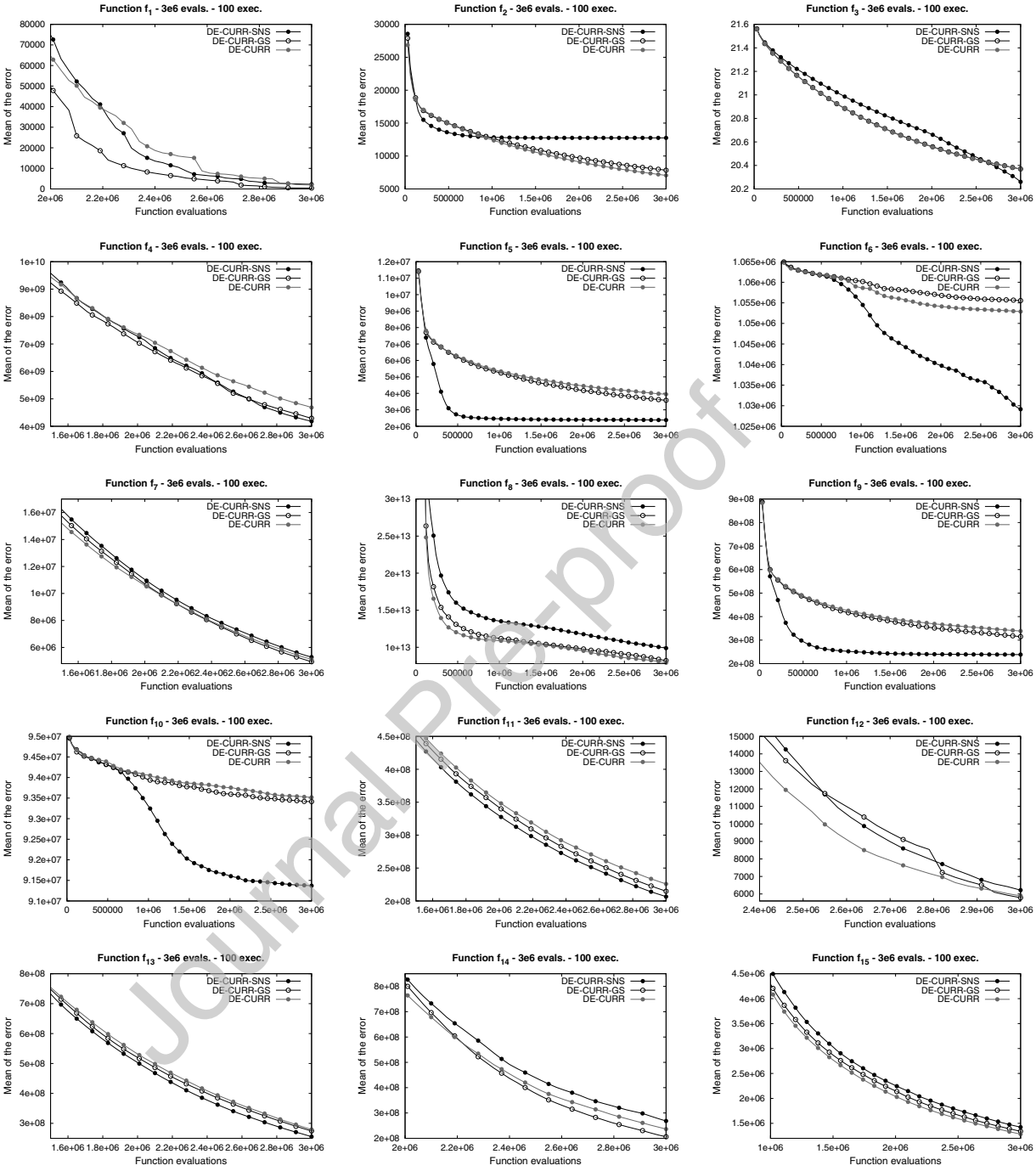

Figure 3: Evolution of the mean of the error for schemes DE-CURR-SNS, DE-CURR-GS, and DE-CURR considering 100 executions

over 100 independent runs for schemes DE-CURR-SNS, DE-CURR-GS and DE-CURR. As in the case of the previous experiment, axes were modified for some test cases, (f_1, f_4 and f_7), to facilitate visualisation of the differences among approaches. In 8 out of 15 cases, the best result is obtained by DE-CURR-SNS: in six test cases (f_5, f_6, f_9, f_{10}, f_{11} and f_{13}), DE-CURR-SNS exhibits the lowest mean of the error for almost

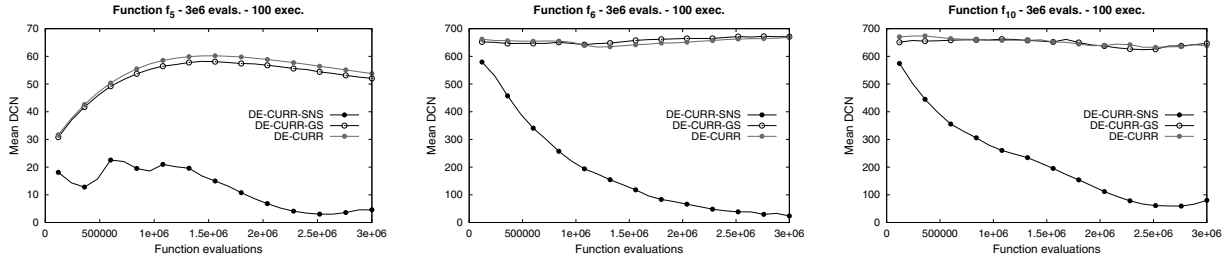

Figure 4: Evolution of the mean distance to the closest neighbour (DCN) for schemes DE-CURR-SNS, DE-CURR-GS, and DE-CURR considering 100 executions

the entire run, while for functions f_3 and f_4, DE-CURR-SNS overtakes the other algorithms during the latter stages of the search process. Functions f_5, f_6 and f_{10} clearly illustrate that SNS behaves differently to the other approaches in terms of speed of convergence.

To gain further insight into the role that diversity plays in improving results, we show the evolution of the mean distance to the closest neighbour (DCN) for each of the schemes DE-CURR-SNS, DE-CURR-GS and DE-CURR in Figure 4, for functions f_5, f_6 and f_{10}. As with the previous experiment, DE-CURR-GS and DE-CURR maintain high diversity during the entire run with respect to DE-CURR-SNS, but are outperformed by DE-CURR-SNS. Although all approaches utilise the same underlying exploitative version of DE, the incorporation of SNS enables smarter diversity management, decreasing diversity as the execution advances. The fact that our proposed SNS is able to promote diversification and intensification at early and late stages of the optimisation process, respectively, is therefore shown once more. Despite DE-CURR-SNS showed the best performance for a significant number of the functions tested in comparison to DE-CURR-GS and DE-CURR, in the case of other problems, such as f_8, f_{14} and f_{15}, DE-CURR-GS and DE-CURR demonstrated to perform better than DE-CURR-SNS during the entire execution. Those problems are unimodal, and therefore, schemes that mainly promote exploitation during the whole run show to be more suitable. DE-CURR-SNS not only promotes exploitation at the end of the runs but exploration at the beginning of the executions, which may be counterproductive when addressing unimodal problems. A possibility to mitigate the above, which may be a line of work worth being carried out, could be to speed up the way the balance from exploration towards exploitation is performed.

Table 7 shows the mean, the median, and the standard deviation (SD) of the error attained by DE-CURR-SNS, DE-CURR-GS and DE-CURR over the repeated experiments. DE-CURR-SNS provided the lowest mean and median of the error at the end of the executions in 7 out of 15 functions (f_3, f_5, f_6, f_9–f_{11} and f_{13}), while DE-CURR-GS and DE-CURR attained the lowest mean and median of the error at the end of the runs for problems f_1 and f_7, and f_2 and f_8, respectively.

In order to statistically support the above results, Table 8 shows the pairwise statistical comparison among the different approaches taken into account for this particular experiment. We make the following observations:

- DE-CURR-SNS statistically outperformed DE-CURR-GS in **five out of 15 test cases** (f_3, f_5, f_6, f_9 and f_{10}), while DE-CURR-GS outperformed DE-CURR-SNS in **four test cases** (f_1, f_2, f_{12} and f_{15}). For the remaining problems, DE-CURR-SNS and DE-CURR-GS did not present statistically significant differences.

- DE-CURR-SNS was statistically better than DE-CURR on **7 out of 15 problems** (f_3–f_6 and f_9–f_{11}), while DE-CURR statistically outperformed DE-CURR-SNS in **five test cases** (f_1, f_2, f_8, f_{12} and f_{15}). For the remaining functions, both schemes did not present statistically significant differences.

- DE-CURR-GS statistically outperformed DE-CURR in **four out of 15 functions** (f_1, f_4, f_5 and f_9), while DE-CURR was statistically better than DE-CURR-GS in **two test cases** (f_2 and f_{15}). Significant differences between both approaches did not arise for the remaining problems.

Table 7: Mean, median, and standard deviation (SD) of the error achieved by DE-CURR-SNS, DE-CURR-GS, and DE-CURR at the end of 100 executions for problems f_1–f_{15}

Alg.	DE-CURR-SNS			DE-CURR-GS		
Func.	Mean	Median	SD	Mean	Median	SD
f_1	2.272e+03	3.092e+02	9.656e+03	**4.178e+02**	**1.156e+02**	1.486e+03
f_2	1.276e+04	1.297e+04	1.090e+03	7.849e+03	6.897e+03	2.334e+03
f_3	**2.026e+01**	**2.026e+01**	2.182e-02	2.037e+01	2.037e+01	6.342e-03
f_4	**4.184e+09**	4.082e+09	1.176e+09	4.292e+09	**3.959e+09**	1.262e+09
f_5	**2.378e+06**	**2.377e+06**	3.508e+05	3.580e+06	3.588e+06	3.463e+05
f_6	**1.029e+06**	**1.030e+06**	9.572e+03	1.056e+06	1.058e+06	8.883e+03
f_7	5.282e+06	4.852e+06	2.110e+06	**4.941e+06**	**4.524e+06**	1.975e+06
f_8	9.885e+12	8.693e+12	6.152e+12	8.211e+12	8.092e+12	4.642e+12
f_9	**2.385e+08**	**2.377e+08**	2.380e+07	3.148e+08	3.110e+08	2.360e+07
f_{10}	**9.137e+07**	**9.123e+07**	5.046e+05	9.341e+07	9.378e+07	1.044e+06
f_{11}	**2.066e+08**	**1.986e+08**	5.312e+07	2.148e+08	2.094e+08	4.666e+07
f_{12}	6.218e+03	5.959e+03	1.063e+03	**5.787e+03**	5.680e+03	6.657e+02
f_{13}	**2.553e+08**	**2.408e+08**	9.506e+07	2.746e+08	2.643e+08	9.762e+07
f_{14}	2.684e+08	**1.294e+08**	4.005e+08	**2.060e+08**	1.381e+08	2.295e+08
f_{15}	1.423e+06	1.385e+06	2.780e+05	1.340e+06	1.310e+06	1.778e+05

Alg.	DE-CURR		
Func.	Mean	Median	SD
f_1	1.941e+03	1.942e+02	1.506e+04
f_2	**7.060e+03**	**6.063e+03**	2.033e+03
f_3	2.037e+01	2.037e+01	6.781e-03
f_4	4.685e+09	4.640e+09	1.336e+09
f_5	3.946e+06	3.944e+06	3.036e+05
f_6	1.053e+06	1.058e+06	1.330e+04
f_7	5.112e+06	4.672e+06	2.219e+06
f_8	**7.924e+12**	**7.005e+12**	4.920e+12
f_9	3.387e+08	3.390e+08	1.899e+07
f_{10}	9.352e+07	9.378e+07	9.070e+05
f_{11}	2.258e+08	2.239e+08	5.024e+07
f_{12}	5.894e+03	**5.579e+03**	2.181e+03
f_{13}	2.793e+08	2.649e+08	9.648e+07
f_{14}	2.364e+08	1.349e+08	2.630e+08
f_{15}	**1.284e+06**	**1.255e+06**	1.513e+05

Table 8: Pairwise statistical comparison among DE-CURR-SNS, DE-CURR-GS, and DE-CURR considering their results achieved at the end of 100 executions for problems f_1–f_{15}

	DE-CURR-SNS vs. DE-CURR-GS		DE-CURR-SNS vs. DE-CURR		DE-CURR-GS vs. DE-CURR	
Func.	p-value	Stat.	p-value	Stat.	p-value	Stat.
f_1	**3.125e-08**	↓	**1.853e-03**	↓	**9.735e-03**	↑
f_2	**2.278e-28**	↓	**1.367e-31**	↓	**1.691e-03**	↓
f_3	**2.701e-78**	↑	**9.589e-79**	↑	4.113e-01	↔
f_4	8.584e-01	↔	**8.024e-03**	↑	**2.001e-02**	↑
f_5	**1.571e-61**	↑	**3.588e-84**	↑	**1.324e-13**	↑
f_6	**1.735e-28**	↑	**9.463e-23**	↑	3.456e-01	↔
f_7	2.200e-01	↔	3.544e-01	↔	7.638e-01	↔
f_8	8.278e-02	↔	**1.937e-02**	↓	5.462e-01	↔
f_9	**3.385e-57**	↑	**3.554e-82**	↑	**2.720e-13**	↑
f_{10}	**8.112e-21**	↑	**9.664e-25**	↑	7.843e-01	↔
f_{11}	1.021e-01	↔	**3.906e-03**	↑	1.090e-01	↔
f_{12}	**5.259e-05**	↓	**7.598e-07**	↓	1.579e-01	↔
f_{13}	1.501e-01	↔	9.760e-02	↔	8.145e-01	↔
f_{14}	8.892e-01	↔	8.546e-01	↔	9.942e-01	↔
f_{15}	**4.319e-03**	↓	**1.312e-06**	↓	**1.950e-02**	↓

Thus we conclude that hybridising SNS with both exploitative and explorative DE variants is beneficial for this test-suite of scalable optimisation problems. The approach outperforms the basic DE variant and also the recently introduced global-search operator GS on a wide selection of instances. However, SNS provides more noticeable benefit when combined with the explorative DE than the exploitative DE with respect to GS.

4.3. Analysing the performance of the similarity-based neighbourhood search with SHADE

In this third experiment, we analyse if the novel SNS is able to provide any advantage in terms of performance when it is embedded into SHADE, which is another adaptive DE variant with a different operation than that applied by JADE. For doing that, we compare the three approaches DE-SHA-SNS, DE-SHA-GS and DE-SHA, which are applied with the parameterisation shown in Table 9. As in the case of the second

Table 9: Parameterisation of DE-SHA-SNS, DE-SHA-GS and DE-SHA

Parameter	Value	Parameter	Value
Stopping criterion	$3 \cdot 10^6$ evals.	Mutation scale factor (F)	Adapted by SHADE
Population size (n)	300	Crossover rate (CR)	Adapted by SHADE

experiment, the population size n was fixed to 300 individuals, carrying out a preliminary study that indicated that this value provided the best overall performance for problems f_1–f_{15}. The minimum number of individuals involved in the selection process of SNS was set to five individuals ($\delta = 5$), as in previous experiments.

Figure 5 shows the evolution of the mean of the error with respect to the objective function value over 100 independent runs for schemes DE-SHA-SNS,DE-SHA-GS and DE-SHA. As in the case of previous experiments, axes were modified for several test cases, with the aim of facilitating visualisation of the differences among approaches. In 6 out of 15 test cases (f_6, f_7, f_9, f_{10}, f_{13} and f_{14}), the best mean of the error was achieved by DE-SHA-SNS, either during almost the whole execution or at its end. In the case of DE-SHA-GS, the best mean of the error was provided in 5 out of 15 functions (f_1, f_4, f_5, f_{11} and f_{12}). Bearing the above in mind, we can conclude that in 11 out 15 problems, which represents 73.3% of all test cases, the hybridisation between SHADE and an additional mechanism to improve the search—either SNS or GS—provided benefits in terms of performance. Only for test cases f_2, f_3, f_8 and f_{15}, the approach DE-SHA, which is the original implementation of SHADE, was able to attain the best results.

Table 10 shows the mean, the median, and the standard deviation (SD) of the error attained by DE-SHA-SNS, DE-SHA-GS and DE-SHA at the end of the executions, while Table 11 shows the pairwise statistical comparison among the different approaches involved in this third experiment. Considering Table 10, the results shown in Figure 5 can be confirmed. DE-SHA-SNS provided the best mean and median of the error at the end of the runs in six test cases, while DE-SHA-GS and DE-SHA yielded the best mean and median of the error in five and four test cases, respectively. Thus, hybridising the particular SHADE, either with SNS or GS, is effective.

On the other hand, Table 11 shows that, in terms of the distribution of results over multiple runs and comparison of mean and medians, there is no statistically significant difference in 9 out 15 functions between the two hybrid methods, with DE-SHA-GS producing a better mean and median in four of the remaining six problems (in contrast to the results shown in Table 10, where it can be observed that DE-SHA-SNS provided a better mean and median in a larger number of instances in comparison to DE-SHA-GS). Finally, DE-SHA-SNS was statistically better or did not show any statistical difference with respect to DE-SHA in 10 out of 15 problems, while DE-SHA-GS was statistically superior or did not present statistically significant differences in comparison to DE-SHA in 13 out of 15 test cases. The above demonstrates again that hybridising is a better approach to be taken into consideration.

In order to perform a more in-depth analysis, and as we previously mentioned, DE-SHA-SNS was able to achieve the best mean and median of the error in six test cases (f_6, f_7, f_9, f_{10}, f_{13} and f_{14}) as Table 10 shows. Considering those particular six functions, in the case of the comparison against DE-SHA, DE-SHA-SNS was statistically superior in functions f_6, f_9, f_{10} and f_{14}, while for problems f_7 and f_{13}, differences were not significant. DE-SHA-GS was able to provide the best mean and median of the error in five problems (f_1, f_4, f_5, f_{11} and f_{12}) as Table 10 shows. Taking into account those five problems, with respect to DE-SHA, DE-SHA-GS was statistically better in problems f_4 and f_5, while for the remaining problems differences were not significant. The above means that from a total number of 11 test cases, where either DE-SHA-SNS or DE-SHA-GS provided the best results at the end of the runs, in six of them, the performance of any of both approaches was statistically better in comparison to DE-SHA.

Finally, as it was previously mentioned, DE-SHA-GS statistically outperformed DE-SHA-SNS in four problems (f_1, f_2, f_3 and f_{12}), but the above is only relevant for test cases f_1 and f_{12}, where DE-SHA-GS provided the best mean and median of the error. For functions f_2 and f_3, DE-SHA attained the best mean and median of the error. Actually, DE-SHA statistically outperformed DE-SHA-SNS and DE-SHA-GS in the case of function f_2, and DE-SHA-SNS in the case of problem f_3.

As a clear conclusion, we can state that hybridising, not only the novel SNS, but also GS, with a different

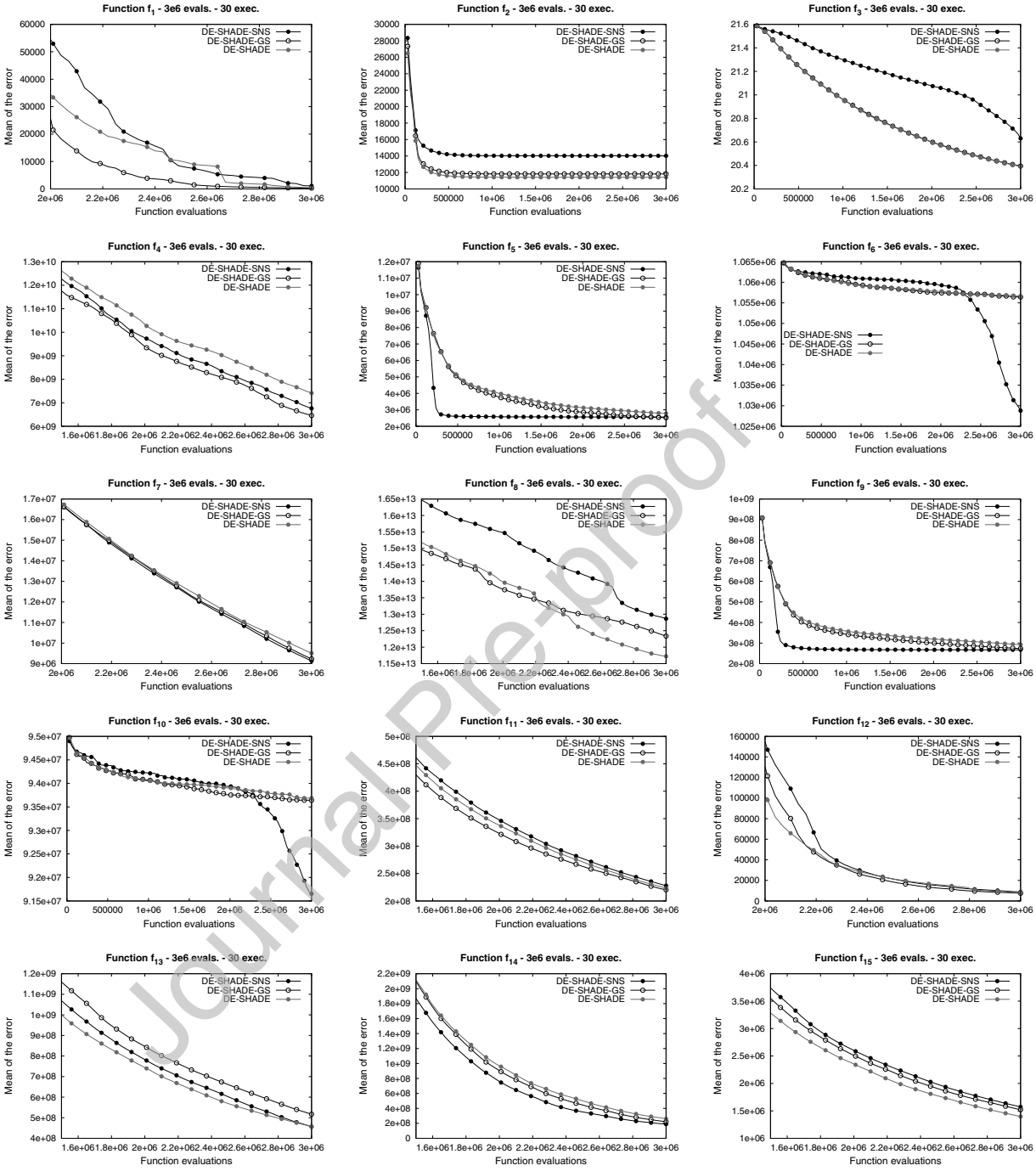

Figure 5: Evolution of the mean of the error for schemes DE-SHA-SNS, DE-SHA-GS, and DE-SHA considering 100 executions

adaptive variant of DE, such as SHADE, provides benefits in terms of the results yielded for this particular test-suite of large-scale problems.

Table 10: Mean, median, and standard deviation (SD) of the error achieved by DE-SHA-SNS, DE-SHA-GS, and DE-SHA at the end of 100 executions for problems f_1–f_{15}

Alg.	DE-SHA-SNS			DE-SHA-GS		
Func.	Mean	Median	SD	Mean	Median	SD
f_1	1.098e+03	4.089e+02	1.590e+03	**2.960e+02**	**1.978e+02**	5.832e+02
f_2	1.403e+04	1.416e+04	6.293e+02	1.182e+04	1.186e+04	4.259e+02
f_3	2.063e+01	2.063e+01	4.067e-02	2.039e+01	2.040e+01	7.375e-03
f_4	6.761e+09	6.453e+09	1.838e+09	**6.460e+09**	**6.257e+09**	1.181e+09
f_5	2.565e+06	2.620e+06	3.951e+05	**2.520e+06**	**2.556e+06**	2.996e+05
f_6	**1.029e+06**	**1.027e+06**	1.141e+04	1.056e+06	1.057e+06	2.036e+03
f_7	**9.130e+06**	**8.182e+06**	4.156e+06	9.231e+06	8.929e+06	3.267e+06
f_8	1.286e+13	1.128e+13	5.902e+12	1.234e+13	1.131e+13	5.245e+12
f_9	**2.671e+08**	**2.643e+08**	2.874e+07	2.733e+08	2.758e+08	1.798e+07
f_{10}	**9.165e+07**	**9.153e+07**	6.820e+05	9.364e+07	9.370e+07	2.641e+05
f_{11}	2.279e+08	2.250e+08	5.528e+07	**2.203e+08**	**2.127e+08**	4.241e+07
f_{12}	8.764e+03	8.250e+03	1.608e+03	**7.481e+03**	**7.324e+03**	8.482e+02
f_{13}	**4.565e+08**	**4.168e+08**	1.653e+08	5.169e+08	5.010e+08	1.868e+08
f_{14}	**1.891e+08**	**1.456e+08**	1.357e+08	2.215e+08	1.862e+08	1.697e+08
f_{15}	1.573e+06	1.564e+06	2.062e+05	1.522e+06	1.491e+06	2.046e+05

Alg.	DE-SHA		
Func.	Mean	Median	SD
f_1	4.467e+02	2.659e+02	6.837e+02
f_2	**1.142e+04**	**1.135e+04**	4.233e+02
f_3	**2.039e+01**	**2.039e+01**	6.686e-03
f_4	7.414e+09	7.140e+09	1.817e+09
f_5	2.780e+06	2.812e+06	2.506e+05
f_6	1.057e+06	1.057e+06	1.475e+03
f_7	9.506e+06	9.153e+06	3.722e+06
f_8	**1.172e+13**	**1.090e+13**	5.945e+12
f_9	2.924e+08	2.930e+08	1.761e+07
f_{10}	9.369e+07	9.375e+07	2.478e+05
f_{11}	2.235e+08	2.240e+08	3.753e+07
f_{12}	8.128e+03	7.667e+03	1.486e+03
f_{13}	4.566e+08	4.422e+08	1.538e+08
f_{14}	2.595e+08	2.165e+08	1.542e+08
f_{15}	**1.396e+06**	**1.345e+06**	1.471e+05

Table 11: Pairwise statistical comparison among DE-SHA-SNS, DE-SHA-GS, and DE-SHA considering their results achieved at the end of 100 executions for problems f_1–f_{15}

	DE-SHA-SNS vs. DE-SHA-GS		DE-SHA-SNS vs. DE-SHA		DE-SHA-GS vs. DE-SHA	
Func.	p-value	Stat.	p-value	Stat.	p-value	Stat.
f_1	**1.140e-04**	$\downarrow$	**1.662e-02**	$\downarrow$	1.646e-01	$\leftrightarrow$
f_2	**2.241e-21**	$\downarrow$	**1.494e-24**	$\downarrow$	**5.147e-04**	$\downarrow$
f_3	**5.936e-25**	$\downarrow$	**6.237e-25**	$\downarrow$	1.949e-01	$\leftrightarrow$
f_4	6.898e-01	$\leftrightarrow$	1.715e-01	$\leftrightarrow$	**2.109e-02**	$\uparrow$
f_5	6.256e-01	$\leftrightarrow$	**1.497e-02**	$\uparrow$	**5.761e-04**	$\uparrow$
f_6	**2.868e-11**	$\uparrow$	**4.533e-14**	$\uparrow$	8.883e-01	$\leftrightarrow$
f_7	5.059e-01	$\leftrightarrow$	2.939e-01	$\leftrightarrow$	7.338e-01	$\leftrightarrow$
f_8	8.708e-01	$\leftrightarrow$	5.250e-01	$\leftrightarrow$	6.739e-01	$\leftrightarrow$
f_9	3.233e-01	$\leftrightarrow$	**1.594e-04**	$\uparrow$	**1.126e-04**	$\uparrow$
f_{10}	**5.228e-11**	$\uparrow$	**4.286e-11**	$\uparrow$	4.923e-01	$\leftrightarrow$
f_{11}	5.444e-01	$\leftrightarrow$	7.222e-01	$\leftrightarrow$	3.912e-01	$\leftrightarrow$
f_{12}	**2.921e-04**	$\downarrow$	**4.595e-02**	$\downarrow$	1.008e-01	$\leftrightarrow$
f_{13}	1.171e-01	$\leftrightarrow$	8.476e-01	$\leftrightarrow$	1.558e-01	$\leftrightarrow$
f_{14}	2.036e-01	$\leftrightarrow$	**1.300e-02**	$\uparrow$	1.316e-01	$\leftrightarrow$
f_{15}	1.137e-01	$\leftrightarrow$	**4.101e-04**	$\downarrow$	**1.905e-03**	$\downarrow$

5. Conclusions and future lines of work

We have introduced a novel SNS procedure which is able to promote a suitable balance between exploration and exploitation depending on the current stage of the search with the aim of improving the performance on a test-suite of scalable optimisation problems. The method incorporates a selection procedure that takes into account similarities among individuals, as well as information about the current stage of the search process. This information is used to move the balance from exploration towards exploitation as the search progresses. We note, however, that other options, such as moving the balance from exploitation towards exploration as the run advances, may be possible because of the generality of SNS.

A wide experimental assessment of SNS was performed by hybridising it with explorative and exploitative

versions that make use of the parameter adaptation mechanisms provided by one of the most frequently applied adaptive DE schemes: JADE. Both explorative and exploitative DE versions combined with SNS were termed as DE-RAND-SNS and DE-CURR-SNS, respectively. In order to measure the contribution of SNS, both DE-RAND-SNS and DE-CURR-SNS were compared to DE-RAND-GS and DE-CURR-GS, which are hybrid schemes combining the same aforementioned explorative and exploitative DE versions with a state-of-the-art global neighbourhood search (GS) selected for comparison purposes. At the same time, both DE variants were executed isolatedly with the aim of studying if it is worth hybridising them with an additional search procedure. They were referred to as DE-RAND and DE-CURR, respectively. Finally, SNS and GS were also combined with SHADE in order to extend our analyses. Three different approaches were considered in this particular comparison: DE-SHA-SNS, DE-SHA-GS and DE-SHA, with the latest scheme being the original implementation of SHADE. All the above methods were applied to a well-known suite of scalable continuous problems.

From the results, we can conclude that it is worth hybridising an explorative DE version, such as DE/rand/1/bin, with the novel SNS, as well as with the GS selected for comparison purposes. The performance of that particular explorative DE variant was significantly improved by combining it with SNS and GS taking into account a wide range of problems (12 and 11 test cases, respectively). At the same time, the clear superiority of SNS with respect to GS was also shown. DE-RAND-SNS was able to statistically outperform DE-RAND-GS in 10 out 15 problems.

Similar conclusions can be extracted when SNS is combined with an exploitative DE variant, like DE/current-to-pbest/1/bin. Since SNS is able to induce a proper balance between exploration and exploitation, it could be hybridised with both explorative and exploitative DE versions in order to increase their performance when tackling continuous problems with a significant number of dimensions. However, the combination of the GS selected for comparison purposes with an exploitative DE variant did not provide any advantage for a wide range of test cases.

It is worth mentioning that for those functions where SNS clearly showed a significantly better behaviour than the rest of approaches in terms of convergence and solution quality, the diversity of the population played a major role. In that regard, we reported and observed that, while DE and GS attained a very diverse population along generations, SNS promoted it in a much smarter way. This points out the importance that a proper management of the diversity may cause over the performance of DE.

Finally, the third experiment revealed that the hybridisation of SHADE with either SNS (DE-SHA-SNS) or GS (DE-SHA-GS) provided better performance in 11 out of 15 test cases—presenting statistically significant differences in 6 out of those 11 problems—in comparison to the original implementation of SHADE (DE-SHA), which did not make use of any additional procedure to improve the search process.

Due to the generality of the novel SNS, an interesting line of future research would be to assess its performance with other definitions of the functions $l(\omega)$ and $u(\omega)$. Furthermore, not only knowledge about the current stage of the search, but also additional information of the search procedure may be used by the individual selection mechanism of SNS. Finally, the incorporation and contribution analysis of our approach in solution frameworks ensembling different DE variants such as EDEV (Wu et al., 2018) would be another line worth being explored.

Credit Author Statement

Eduardo Segredo: Conceptualization, software, data curation, methodology, validation, formal analysis, investigation, resources, writing - original draft, writing - review & editing, visualization, supervision

Eduardo Lalla-Ruiz: Conceptualization, software, data curation, methodology, validation, formal analysis, investigation, resources, writing - original draft, writing - review & editing, visualization, supervision

Emma Hart: Conceptualization, validation, formal analysis, investigation, writing - review & editing

Stefan Voss: Conceptualization, validation, formal analysis, investigation, writing - review & editing

Acknowledgements

We are grateful to the anonymous referees for their constructive and valuable comments that helped to improve this manuscript.

Funding. This work was supported by the Spanish Ministry of Economy, Industry and Competitiveness as part of the "I+D+i Orientada a los Retos de la Sociedad" programme [contract number TIN2016-78410-R].

References

Das, S., Abraham, A., & Konar, A. (2008). Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives. In Y. Liu, A. Sun, H. T. Loh, W. F. Lu, & E.-P. Lim (Eds.), *Advances of Computational Intelligence in Industrial Systems* (pp. 1–38). Berlin, Heidelberg: Springer.

Das, S., Mullick, S. S., & Suganthan, P. (2016). Recent advances in differential evolution – An updated survey. *Swarm and Evolutionary Computation*, *27*, 1 – 30.

Ghasemi, M., Aghaei, J., Akbari, E., Ghavidel, S., & Li, L. (2016). A differential evolution particle swarm optimizer for various types of multi-area economic dispatch problems. *Energy*, *107*, 182–195.

Guo, H., Li, Y., Li, J., Sun, H., Wang, D., & Chen, X. (2014). Differential evolution improved with self-adaptive control parameters based on simulated annealing. *Swarm and Evolutionary Computation*, *19*, 52 – 67.

Guo, Z., Liu, G., Li, D., & Wang, S. (2017). Self-adaptive differential evolution with global neighborhood search. *Soft Computing*, *21*, 3759–3768.

Karafotias, G., Hoogendoorn, M., & Eiben, A. E. (2015). Parameter control in evolutionary algorithms: trends and challenges. *IEEE Transactions on Evolutionary Computation*, *19*, 167–187.

Kazimipour, B., Li, X., & Qin, A. (2014). Effects of population initialization on differential evolution for large scale optimization. In *2014 IEEE Congress on Evolutionary Computation (CEC)* (pp. 2404–2411).

Kovačević, D., Mladenović, N., Petrović, B., & Milošević, P. (2014). DE-VNS: Self-adaptive differential evolution with crossover neighborhood search for continuous global optimization. *Computers & Operations Research*, *52*, 157–169.

LaTorre, A., Muelas, S., & Peña, J. M. (2015). A comprehensive comparison of large scale global optimizers. *Information Sciences*, *316*, 517 – 549.

León, C., Miranda, G., & Segura, C. (2009). METCO: a parallel plugin-based framework for multi-objective optimization. *International Journal on Artificial Intelligence Tools*, *18*, 569–588.

Li, X., Tang, K., Omidvar, M., Yang, Z., & Qin, K. (2013). *Benchmark Functions for the CEC'2013 Special Session and Competition on Large Scale Global Optimization*. Technical Report Evolutionary Computation and Machine Learning Group, RMIT University Australia.

Lozano, M., & García-Martínez, C. (2010). Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: overview and progress report. *Computers & Operations Research*, *37*, 481 – 497. doi:10.1016/j.cor.2009.02.010.

Mahdavi, S., Shiri, M. E., & Rahnamayan, S. (2015). Metaheuristics in large-scale global continues optimization: a survey. *Information Sciences*, *295*, 407 – 428.

Neri, F., & Tirronen, V. (2010). Recent advances in differential evolution: a survey and experimental analysis. *Artificial Intelligence Review*, *33*, 61–106.

Parouha, R. P., & Das, K. N. (2016). A robust memory based hybrid differential evolution for continuous optimization problem. *Knowledge-Based Systems*, *103*, 118–131.

Segredo, E., Paechter, B., Segura, C., & González-Vila, C. I. (2017). On the comparison of initialisation strategies in differential evolution for large scale optimisation. *Optimization Letters*, .

Segura, C., Coello Coello, C. A., Segredo, E., & Aguirre, A. H. (2016). A novel diversity-based replacement strategy for evolutionary algorithms. *IEEE Transactions on Cybernetics*, *46*, 3233–3246.

Segura, C., Coello Coello, C. A., Segredo, E., & León, C. (2015). On the adaptation of the mutation scale factor in differential evolution. *Optimization Letters*, *9*, 189–198.

Storn, R., & Price, K. (1997). Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*, 341–359.

Tanabe, R., & Fukunaga, A. (2013). Success-history based parameter adaptation for differential evolution. In *2013 IEEE Congress on Evolutionary Computation* (pp. 71–78).

Trivedi, A., Srinivasan, D., Biswas, S., & Reindl, T. (2015). Hybridizing genetic algorithm with differential evolution for solving the unit commitment scheduling problem. *Swarm and Evolutionary Computation*, *23*, 50–64.

Tvrdík, J., Poláková, R., Veselský, J., & Bujok, P. (2013). Adaptive variants of differential evolution: towards control-parameter-free optimizers. In I. Zelinka, V. Snášel, & A. Abraham (Eds.), *Handbook of Optimization: From Classical to Modern Approach* (pp. 423–449). Berlin, Heidelberg: Springer.

Črepinšek, M., Liu, S.-H., & Mernik, M. (2013). Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.*, *45*, 35:1–35:33. URL: http://doi.acm.org/10.1145/2480741.2480752. doi:10.1145/2480741.2480752.

Wang, H., Sun, H., Li, C., Rahnamayan, S., & Pan, J.-S. (2013). Diversity enhanced particle swarm optimization with neighborhood search. *Information Sciences*, *223*, 119 – 135.

Wang, Y., Cai, Z., & Zhang, Q. (2011). Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation*, *15*, 55–66.

Wu, G., Shen, X., Li, H., Chen, H., Lin, A., & Suganthan, P. N. (2018). Ensemble of differential evolution variants. *Information Sciences*, *423*, 172–186.

Xin, B., Chen, J., Zhang, J., Fang, H., & Peng, Z.-H. (2012). Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: a review and taxonomy. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*, 744–767.

Xu, Q., Wang, L., Wang, N., Hei, X., & Zhao, L. (2014). A review of opposition-based learning from 2005 to 2012. *Engineering Applications of Artificial Intelligence*, *29*, 1 – 12.

Zhang, J., & Sanderson, A. C. (2009). JADE: adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, *13*, 945–958.