A Novel Similarity-based Mutant Vector Generation Strategy for Differential Evolution

Eduardo Segredo
School of Computing, Edinburgh Napier University
Edinburgh, United Kingdom
e.segredo@napier.ac.uk
Dpto. de Ingeniería Informática y de Sistemas, Universidad de La Laguna
San Cristóbal de La Laguna, Spain
esegredo@ull.edu.es

Eduardo Lalla-Ruiz
Institute of Information Systems, University of Hamburg
Hamburg, Germany
eduardo.lalla-ruiz@uni-hamburg.de

Emma Hart
School of Computing, Edinburgh Napier University
Edinburgh, United Kingdom
e.hart@napier.ac.uk

ABSTRACT

The mutant vector generation strategy is an essential component of Differential Evolution (DE), introduced to promote diversity, resulting in exploration of novel areas of the search space. However, it is also responsible for promoting intensification, to improve those solutions located in promising regions. In this paper we introduce a novel similarity-based mutant vector generation strategy for DE, with the goal of inducing a suitable balance between exploration and exploitation, adapting its behaviour depending on the current state of the search. In order to achieve this balance, the strategy considers similarities among individuals in terms of their Euclidean distance in the decision space. A variant of DE incorporating the novel mutant vector generation strategy is compared to well-known explorative and exploitative adaptive DE variants. An experimental evaluation performed on a well-known suite of large-scale continuous problems shows that the new DE algorithm that makes use of the similarity-based approach provides better performance in comparison to the explorative and exploitative DE variants for a wide range of the problems tested, demonstrating the ability of the new component to properly balance exploration and exploitation.

CCS CONCEPTS

• **Mathematics of computing → Evolutionary algorithms**; **Bio-inspired optimization**; • **Theory of computation → Continuous optimization**; • **Computing methodologies → Continuous space search**;

KEYWORDS

Global optimization, Differential evolution, Similarity, Diversity, Large-scale optimization

GECCO '18, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205628

ACM Reference Format:
Eduardo Segredo, Eduardo Lalla-Ruiz, and Emma Hart. 2018. A Novel Similarity-based Mutant Vector Generation Strategy for Differential Evolution. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205455.3205628

1 INTRODUCTION

Differential Evolution (DE) is a sub-class of Evolutionary Algorithms (EAs), that is arguably one of the most powerful and versatile evolutionary optimisers for continuous parameter spaces in recent times [3]. A large body work considers the importance of maintaining both fitness and population diversity within DE, given that maintaining diversity is understood to be a prerequisite for avoiding rapid convergence to local optima. A recent survey article [3] provides a detailed insight into numerous strategies for promoting exploration at different stages of the evolutionary process with DE. However, as noted by [2], promoting diversity at all stages of an evolutionary process might be counterproductive, resulting in a diverse but poor population.

This motivates a search for solution approaches and strategies that properly manage the population diversity while simultaneously driving high-quality performance through search. Within DE, one way to mitigate diversity loss can be achieved by defining *mutant vector generation strategies* that simultaneously consider the characteristics of the individuals and state of the search when selecting the individuals involved in mutation. In this paper, we propose a novel strategy for generating mutant vectors, where the main goal is to provide a suitable balance between exploration and exploitation depending on the progress of the search procedure. At the same time, it aims to improve the quality of the solutions provided at the end of the executions. Mutant vectors are typically created by combining three randomly selected solutions (r_1, r_2, r_3) from the population: the specific method of combination can be chosen to either promote intensification or diversification, and can be tuned through judicious parameter choice. Our novel method introduces a function that determines what fraction of a ranked population that one of the randomly selected solutions r_3 can be chosen from, rather than randomly selecting from the entire population, depending on how many iterations have taken place.

First, the population is sorted in descending order in terms of the similarity (based on Euclidean distance in the decision space)

of each individual with respect to the fittest individual. A new function dictates what fraction of the sorted population is permitted to be used for selection of r_3 such that at the beginning of a given run, r_3 will be selected from among the best individual's δ least similar neighbours in the current population, therefore promoting exploration. As more and more function evaluations are performed, however, the best 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 best individual's δ most similar neighbours are involved in the selection, and consequently, exploitation is promoted.

We evaluate the new procedure (termed DE-SIM) on a well-known test suite of scalable continuous optimisation problems proposed in [7] consisting of 15 functions. This test suite was proposed for the special session and competition on *Large Scale Global Optimisation* organised in the field of the *Congress on Evolutionary Computation* (CEC) 2013. Both the special session and the competition have been organised in subsequent editions of that conference, including the most recent CEC'18. As a result, it is the latest test suite proposed for large scale continuous optimisation. Considering the different editions of the competition, the best performing approaches usually combine multiple algorithms to solve an instance. In the current work, the main goal is not to outperform those state-of-the-art approaches which rely heavily on algorithm portfolios, since a direct comparison to these hybrid schemes would not be fair.

Bearing the above discussion in mind, the following questions are addressed in this paper:

- To what extent does DE-SIM outperform an explorative strategy (*rand/1*).
- To what extent does DE-SIM outperform an exploitative strategy (*current-to-pbest/1*).

Experimental results show that the new method statistically outperforms *rand/1* in 73% of the functions tested, and outperforms *current-to-pbest/1* on 40% of test-instances. We conclude with some general comments regarding the manner by which the new method maintains a suitable balance between exploration and exploitation, and suggestions for future-work relating to how the speed by which DE-SIM shifts from exploration to exploitation might be addressed.

2 BACKGROUND

In the related literature, the impact on diversity that mutant vector generation strategies may cause has been explored mainly in the context of low-dimensional continuous optimisation. For example, in [4], a ranking-based mutation operator embedded in DE for unconstrained low-scale continuous optimisation was proposed. In this approach, individuals are ranked according to their fitness, then during mutation, some of them are proportionally selected with regard to the created ranked list. Similarly, recent work proposed in [12] considered the selection of individuals using stochastic ranking and probabilistic-based selection when dealing with small-scale continuous constrained optimisation. In this case, those individuals having higher positions within the ranking are more likely to be selected during the mutation. In [1], alternative ranking-based mutation operators are introduced. Some of the parents in the mutation strategies are proportionally selected according to their fitness ranking in the population. The higher ranking a parent obtains, the more opportunity it will be selected, hence enhancing the exploitation ability of DE. As with the methods described above, the proposed ranking method does not consider diversity *explicitly*, only the objective function values or their constraint violations.

Unlike previously proposed strategies that consider ranking vectors in multiple ways according to fitness, we propose a novel method that ranks a population with respect to similarity of each individual to the fittest one. A linear ranking function then dynamically selects a proportion of individuals from the ranked list from which a vector is chosen at random to participate in the mutation, thus enabling the balance of exploration and exploitation to change over the course of the search, something that has not been addressed previously. In addition, in contrast to previous work in which the vast majority of experimental studies undertaken to evaluate new mutant strategies have only considered low-dimensional continuous problems as test-beds, we tackle a more challenging test-bed composed of large-scale problems.

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
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 vector 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: **end while**
10: **return** the fittest individual in the population

3 METHOD

In Section 3.1 we provide a general description of DE and of the explorative and exploitative DE versions used for comparison throughout the paper. The novel similarity-based mutant vector generation strategy is introduced in Section 3.2. For all DE variants discussed, parameter adaptation mechanisms are provided by JADE: this is described in Section 3.3.

3.1 Differential evolution

We adopt the most frequently used nomenclature for DE [11], 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. The variants DE/*rand/1/bin* and DE/*current-to-pbest/1/bin*, which intrinsically promote exploration and exploitation [10, 15], respectively, are selected as suitable comparison schemes. The term *bin* refers to the *binomial crossover*, which is described later.

Algorithm 1 shows the operation of DE. First of all, n individuals are generated through an initialisation strategy (step 1). In this

work, *Opposition-based Learning* (OBL) [14] is applied as the initialisation mechanism. With respect to the set of large-scale problems addressed, previous work shows that the incorporation of OBL into an explorative DE variant, such as DE/rand/1/bin, is likely to provide better solutions than those achieved by applying other initialisation approaches [8]. 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 vector generation strategy* is applied to produce a *mutant vector* $\vec{V}_j$ (step 5). Equation 1 describes the mutant vector generation strategy *rand/1*, 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 vector generation strategy are randomly selected, this strategy 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 exploration, while small values turn the approach into a more exploitative scheme.

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

With the aim of applying a DE variant that promotes intensification rather than diversification, the mutant vector generation strategy *current-to-pbest/1*, is also considered herein. In this variant, a mutant vector $\vec{V}_j$ is created starting from a target vector $\vec{X}_j$ as it is described in Equation 2. 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 considered by this mutant vector generation strategy, and consequently, it is more exploitative than the approach rand/1, which only considers randomness. As can be observed, in addition to the mutation scale factor F, parameter p can be used to define the balance between the exploration and exploitation capabilities of the algorithm. By considering large p values, the scheme becomes more explorative, while it becomes more exploitative with small p values. Finally, the 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 [10, 15].

$$\vec{V}_j = \vec{X}_j + K \times (\vec{X}_{r_3} - \vec{X}_j) + F \times (\vec{X}_{r_1} - \vec{X}_{r_2}) \tag{2}$$

Once the mutant vector is obtained, it is combined with the target vector through the application of a crossover operator 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 was selected, whose operation is shown in Equation 3. 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 $i_{rand} \in [1, 2, \ldots, D]$ 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.

$$u_{j,i} = \begin{cases} v_{j,i} & if\ (rand_{j,i} \leq CR\ or\ i = i_{rand}) \\ x_{j,i} & otherwise \end{cases} \tag{3}$$

The trial vector generation strategy might produce infeasible individuals. To address this, an infeasible value in a particular decision variable is randomly re-initialised in its corresponding feasible range. 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.

3.2 Similarity-based mutant vector generation strategy

This section introduce a novel similarity-based mutant vector generation strategy. The main goal of the strategy is to provide a suitable balance between exploration and exploitation, depending on the current stage of the search procedure, in order to improve the quality of the solutions found. As with the other mutant vector generation strategies described in the previous section, the similarity-based approach is applied at step 5 of Algorithm 1. In order to generate a new individual, the similarity-based mutant vector generation strategy makes use of Equation 2. Depending on the particular selection of the individual $\vec{X}_{r_3}$, the strategy can promote exploration or exploitation. If $\vec{X}_{r_3}$ is similar to the fittest individuals in the population, it will promote exploitation. In this case, its behaviour will be similar to the strategy current-to-pbest/1. On the other hand, if $\vec{X}_{r_3}$ is different from the fittest individuals in the population, it will move the balance towards exploration.

Before selecting individual $\vec{X}_{r_3}$, the population is sorted in descending order in terms of the similarity, i.e., the Euclidean distance in the decision space, of each individual with respect to the fittest individual in the population. Afterwards, an index r_3 (which must be different to indexes r_1, r_2 and j) is randomly selected 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 index r_3 is selected, and depend on the current stage of the search, defined by the number of function evaluations ω performed so far. We apply a linear ascending function shown in Equation 4 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 of $\vec{X}_{r_3}$. Once a particular value is given by $u(\omega)$, the lower bound $l(\omega)$ is calculated as $l(\omega) = u(\omega) - \delta$. Hence, 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.

$$u(\omega) = \frac{n - \delta}{\Omega} \cdot \omega + \delta \tag{4}$$

As a result, at the beginning of a given run, $\vec{X}_{r_3}$ will be selected from among the fittest individual's δ least similar neighbours in the current population. Exploration is thus promoted at early stages of

the search procedure. As more and more function evaluations are performed however, 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 the similarity-based mutant vector generation strategy to be 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.

3.3 Parameter adaptation through JADE

As can be observed in previous sections, values for the mutation scale factor F and the crossover rate CR have to be set to run DE with the mutant vector generation strategies described. *Controlling* or adapting the parameters of an algorithm while it is executed has shown to provide significant benefits with respect to *tuning* or keeping those parameters fixed for the whole execution [5]. Therefore, a significant amount of research relating to the adaptation of DE parameters can be found [3, 13].

JADE [15] includes one of the best performing and most frequently used approaches to adapt parameters F and 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 through 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 achieve 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 considered. 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 5 illustrates the updating mechanism of μ_F.

$$\mu_F = (1 - c) \cdot \mu_F + c \cdot mean_L(S_F) \tag{5}$$

With respect to the control mechanism of CR, it is similar to the control approach of F (Equation 6). 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}.

$$\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot mean_A(S_{CR}) \tag{6}$$

4 EXPERIMENTAL EVALUATION

This section is devoted to describing the computational experiments carried out to assess the performance of the novel similarity-based mutant vector generation strategy proposed in Section 3.2. From now on, the DE version incorporating the novel mutant vector strategy will be termed as DE-SIM. At the same time, the explorative and exploitative DE variants selected for comparison purposes (Section 3.1), which incorporate the mutant vector generation strategies rand/1 and current-to-pbest/1, will be termed as DE-RAND and DE-CURRENT, respectively. At this point, it is worth noting that the three aforementioned DE versions are *adaptive*, since the control mechanisms provided by JADE are used to adapt the values of the mutation scale factor F and the crossover rate CR, as we previously described in Section 3.3.

Experimental method. All the approaches just described were implemented through the *Meta-heuristic-based Extensible Tool for Cooperative Optimisation* (METCO) [6]. 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 the approaches considered are stochastic, each run was repeated 30 times. In order to compare the different DE versions, the following statistical testing procedure was considered [9]. 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 taken into account.

Problem set. A test suite of scalable continuous optimisation problems proposed in [7] has been chosen. Specifically, the problem set consists of 15 functions: 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}). Following the suggestions given in [7], a total number of $D = 1000$ decision variables was fixed for all problems, with the exception of functions f_{13} and f_{14}. Because of overlapping subcomponents, $D = 905$ decision variables were set for these two test cases.

4.1 Comparison of the similarity-based strategy and an explorative strategy

In this first experiment, the main goal was to analyse the performance of the novel similarity-based mutant vector generation strategy with respect to an explorative mutant vector generation method. Thus, the approaches DE-SIM and DE-RAND were compared. The scheme DE-RAND only considers random selection of individuals involved in the mutant vector generation strategy, and therefore, it manages diversity by moving the balance significantly towards exploration. However, if diversity management is not carried out in a intelligent manner, it can be counterproductive. Our hypothesis is that DE-RAND does not manage diversity intelligently, i.e., it mainly promotes exploration during the search, and as a result, an approach inducing a suitable balance between exploration and exploitation, such as DE-SIM, may provide a better performance in terms of the solutions attained. Both schemes were executed by

Table 1: Mean, median, and standard deviation (SD) of the error achieved by DE-SIM and DE-RAND at the end of 30 executions for problems f_1–f_{15}. The last two columns shows if DE-SIM statistically outperformed DE-RAND (↑), if the former was statistically outperformed by the latter (↓) and if both approaches did not show statistically significant differences (↔)

Alg.	DE-SIM			DE-RAND			Statistical comparison	
Func.	Mean	Median	SD	Mean	Median	SD	p-Value	Winner
f_1	**1.898e-13**	**3.225e-18**	1.027e-12	7.215e-02	7.508e-02	1.315e-02	**2.872e-11**	↑
f_2	7.142e+02	6.009e+02	5.226e+02	**1.630e-03**	**1.621e-03**	2.580e-04	**2.872e-11**	↓
f_3	2.002e+01	2.002e+01	4.934e-04	**2.002e+01**	**2.002e+01**	5.830e-04	**6.222e-11**	↓
f_4	**5.357e+10**	**4.780e+10**	2.790e+10	3.533e+11	3.520e+11	1.027e+11	**2.872e-11**	↑
f_5	**7.240e+06**	**7.128e+06**	6.240e+05	7.467e+06	7.592e+06	8.934e+05	1.691e-01	↔
f_6	**1.054e+06**	**1.054e+06**	1.785e+03	1.054e+06	1.057e+06	1.187e+04	**4.789e-05**	↑
f_7	**6.445e+08**	**5.998e+08**	2.547e+08	2.221e+09	2.160e+09	4.139e+08	**3.863e-25**	↑
f_8	**8.633e+14**	**5.896e+14**	9.704e+14	9.460e+15	9.876e+15	2.292e+15	**3.175e-11**	↑
f_9	**5.519e+08**	**5.417e+08**	4.922e+07	5.854e+08	6.079e+08	7.592e+07	**8.875e-03**	↑
f_{10}	**9.334e+07**	**9.346e+07**	2.966e+05	9.337e+07	9.359e+07	7.482e+05	7.604e-02	↔
f_{11}	**1.500e+11**	**1.195e+11**	9.157e+10	1.835e+11	1.791e+11	4.967e+10	**1.406e-03**	↑
f_{12}	**2.714e+03**	**2.726e+03**	1.998e+02	5.019e+03	5.010e+03	1.272e+02	**5.759e-51**	↑
f_{13}	**8.972e+09**	**8.703e+09**	1.926e+09	2.407e+10	2.469e+10	3.577e+09	**4.734e-11**	↑
f_{14}	**1.423e+11**	**1.286e+11**	4.549e+10	3.976e+11	3.896e+11	5.177e+10	**2.872e-11**	↑
f_{15}	**4.472e+07**	**4.162e+07**	1.333e+07	8.294e+07	8.250e+07	7.769e+06	**5.841e-10**	↑

considering $n = 50$ individuals, and by following the suggestions given in [7], the stopping criterion was set to a maximum number of $3 \cdot 10^6$ function evaluations. Furthermore, as we previously stated in Section 3.3, values for parameters F and CR were adapted through the control mechanisms provided by JADE with an adaptation speed $c = 0.1$. In the case of the DE-SIM approach, appropriate values for parameter δ were obtained from a preliminary analysis. In general, $\delta = 5$ provided the best performance for all test cases, with the exception of functions f_2, f_7 and f_8, where $\delta = 10$ attained the best results, and f_{14}, where the value attaining the best performance was $\delta = 15$.

Table 1 shows the mean, the median and the standard deviation (SD) of the error attained by DE-SIM and DE-RAND at the end of the runs for each of the problems tested. Moreover, the results of the statistical comparison carried out with both schemes, by following the statistical procedure described at the beginning of Section 4, is also included. Particularly, the last two columns show, for each function, the p-value and if DE-SIM statistically outperformed DE-RAND (↑), if the former was statistically outperformed by the latter (↓), and if both schemes did not present statistically significant differences (↔). Method A statistically outperforms method B if they present statistically significant differences (p-value < 0.05), and at the same time, A provides a lower mean and median of the error in comparison to B.

The clear superiority of DE-SIM with respect to DE-RAND in terms of the solutions attained at the end of the runs can be observed from this table. DE-SIM was able to achieve the lowest mean and median of the error for all test cases, with the exception of functions f_2 and f_3, for which DE-RAND provided the best performance. In fact, DE-SIM statistically outperformed DE-RAND in 11 out of 15 problems, which represents 73.3% of the functions tested, while DE-RAND was statistically better than DE-SIM only for problems f_2 and f_3. In the case of functions f_5 and f_{10}, both approaches did not present statistically significant differences. The experimental evidence thus demonstrates the benefits of the proposed similarity-based mutant vector generation strategy in comparison to an explorative strategy, such as rand/1.

It would be interesting, however, to analyse how DE-SIM manages diversity with respect to DE-RAND. Figure 1 shows the evolution of the mean distance to the closest neighbour (DCN) for DE-SIM and DE-RAND considering functions f_1–f_{15}. In general, for almost all test cases, it can be observed that the mean DCN values provided by DE-RAND are higher than those attained by DE-SIM during the whole execution. This suggests that DE-RAND preserves higher diversity in the population in comparison to DE-SIM. Nevertheless, as we stated in our hypothesis, managing diversity in an unsuitable manner may be counterproductive. In fact, although DE-RAND preserved higher diversity in the population with respect to DE-SIM, the latter performed significantly better than the former in terms of the solutions provided at the end of the runs for the majority of the functions tested. As we previously stated, DE-RAND randomly selects individuals to be involved in the mutant vector generation strategy, and as a result, it manages diversity by mainly promoting exploration. On the contrary, DE-SIM starts a run by promoting exploration, and as the run progresses, the balance is moved towards exploitation, thus providing a smarter method to manage diversity.

The general behaviour changes, however, when considering functions f_2, f_3 and f_6. In the case of function f_3, DE-RAND achieved better solutions at the end of the executions by preserving higher diversity during the whole run with respect to DE-SIM. The above may be explained by the fact that f_3 could require the application of approaches that mainly promote exploration rather than schemes that in addition consider exploitation. For problems such as f_3, promoting exploitation may cause the optimiser to stagnate in local optima. Taking into account problems f_2 and f_6, DE-SIM provided a higher diversity in comparison to DE-RAND for almost the whole run. For those test cases, DE-SIM may have failed by not moving the balance towards exploitation in a proper way. Despite the above fact, it can be observed how problem f_2 requires the application of an explorative optimiser, while schemes that also consider exploitation are more suitable for function f_6.

4.2 Comparison of the similarity-based strategy and an exploitative strategy

The main objective of this second experiment is to study the performance of the similarity-based mutant vector generation strategy with respect to an exploitative mutant vector generation approach. In this case, we compare schemes DE-SIM and DE-CURRENT. The mutant vector generation strategy current-to-pbest/1 of DE-CURRENT always considers some of the fittest individuals in the population

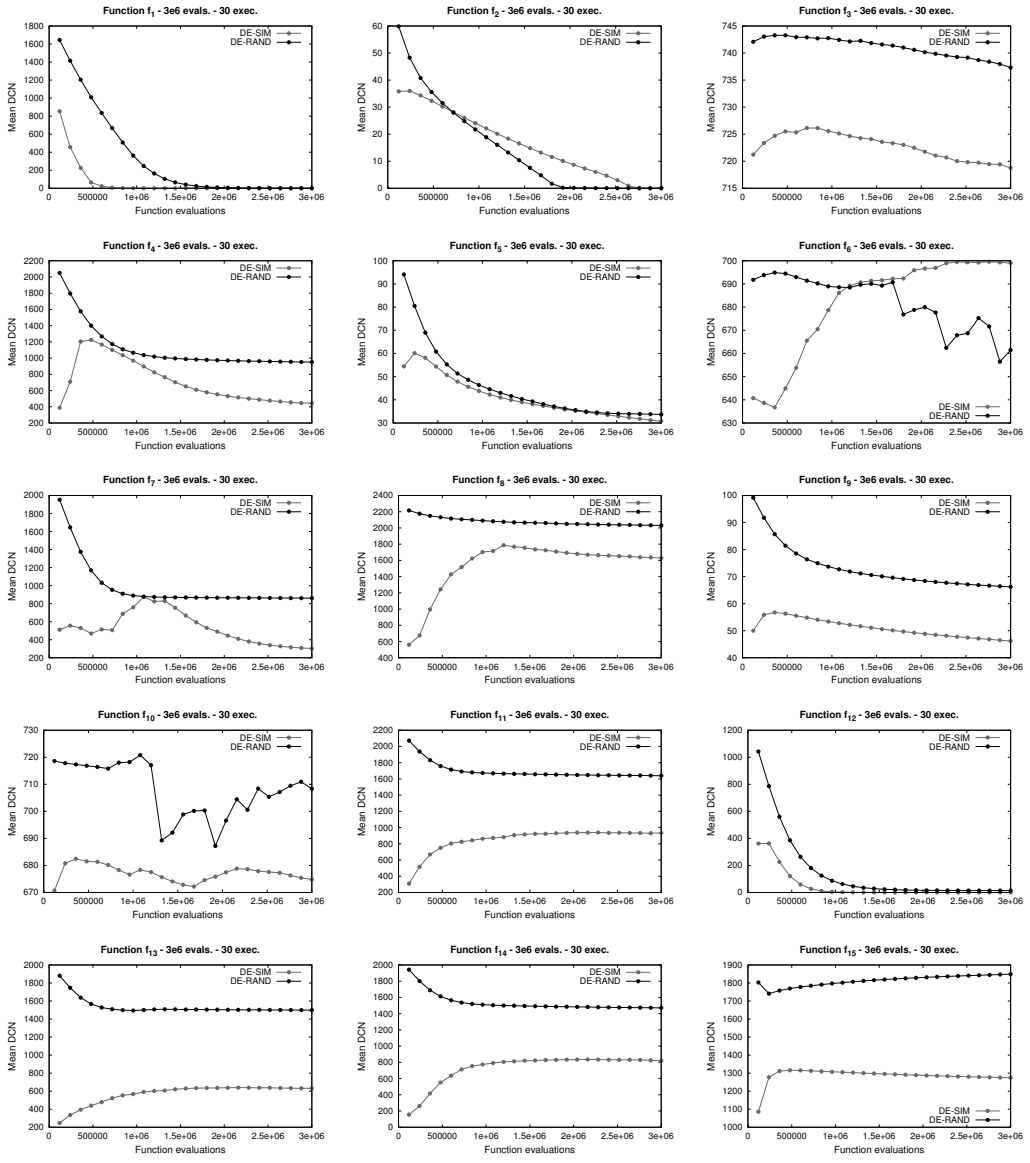

Figure 1: Evolution of the mean distance to the closest neighbour (DCN) for schemes DE-SIM and DE-RAND

to generate novel individuals, and consequently, it is a more exploitative approach in comparison to DE-RAND, which uses random selection. Our hypothesis for this second experiment is similar to that suggested in the first one: DE-CURRENT does not manage diversity in a suitable manner because it mainly promotes exploitation, so DE-SIM may perform better.

The approach DE-CURRENT was run with $n = 300$ individuals, following a preliminary study that revealed that small population sizes decreased performance significantly. This is likely due to a shift to intensification caused by the small population size. In addition, parameter p of the strategy current-to-pbest/1 was set to 0.05. All the remaining parameters were fixed to those values considered in the first experiment.

From Table 2, it can be observed that DE-SIM attained the best mean and median of the error, providing statistically significant differences, at the end of the executions in 6 test cases, which represents 40% of the problems tested. DE-CURRENT provides the best results with statistically significant differences in the remaining 9 problems. We therefore suggest that this leads to the general conclusion that this particular test suite favours the application of optimisers that mainly promote exploitation, such as DE-CURRENT.

Although DE-SIM was shown to manage diversity in a smarter way than DE-RAND by inducing a suitable balance between exploration and exploitation, DE-SIM might be failing with respect to DE-CURRENT in terms of the speed with which it changes from promoting exploration towards exploitation. This could be addressed by

Table 2: Mean, median, and standard deviation (SD) of the error achieved by DE-SIM and DE-CURRENT at the end of 30 executions for problems f_1–f_{15}. The last two columns shows if DE-SIM statistically outperformed DE-CURRENT (↑), if the former was statistically outperformed by the latter (↓) and if both approaches did not show statistically significant differences (↔)

Alg.	DE-SIM			DE-CURRENT			Statistical comparison	
Func.	Mean	Median	SD	Mean	Median	SD	p-Value	Winner
f_1	**1.898e-13**	**3.225e-18**	1.027e-12	1.825e+03	1.588e+02	5.900e+03	**2.872e-11**	↑
f_2	**7.142e+02**	**6.009e+02**	5.226e+02	8.044e+03	6.297e+03	2.765e+03	**2.872e-11**	↑
f_3	**2.002e+01**	**2.002e+01**	4.934e-04	2.037e+01	2.037e+01	7.491e-03	**2.025e-50**	↑
f_4	5.357e+10	4.780e+10	2.790e+10	**4.538e+09**	**4.599e+09**	1.347e+09	**2.872e-11**	↓
f_5	7.240e+06	7.128e+06	6.240e+05	**3.893e+06**	**3.955e+06**	3.801e+05	**3.224e-29**	↓
f_6	**1.054e+06**	**1.054e+06**	1.785e+03	1.054e+06	1.058e+06	1.077e+04	**2.077e-06**	↑
f_7	6.445e+08	5.998e+08	2.547e+08	**4.990e+06**	**5.261e+06**	1.492e+06	**3.071e-14**	↓
f_8	8.633e+14	5.896e+14	9.704e+14	**8.440e+12**	**7.668e+12**	4.409e+12	**2.872e-11**	↓
f_9	5.519e+08	5.417e+08	4.922e+07	**3.333e+08**	**3.349e+08**	1.968e+07	**1.210e-23**	↓
f_{10}	**9.334e+07**	**9.346e+07**	2.966e+05	9.360e+07	9.381e+07	8.223e+05	**5.657e-06**	↑
f_{11}	1.500e+11	1.195e+11	9.157e+10	**2.048e+08**	**2.053e+08**	4.963e+07	**2.872e-11**	↓
f_{12}	**2.714e+03**	**2.726e+03**	1.998e+02	5.748e+03	5.617e+03	7.484e+02	**2.872e-11**	↑
f_{13}	8.972e+09	8.703e+09	1.926e+09	**2.772e+08**	**2.512e+08**	1.196e+08	**2.872e-11**	↓
f_{14}	1.423e+11	1.286e+11	4.549e+10	**1.602e+08**	**1.272e+08**	9.562e+07	**2.872e-11**	↓
f_{15}	4.472e+07	4.162e+07	1.333e+07	**1.264e+06**	**1.268e+06**	9.860e+04	**2.872e-11**	↓

replacing the linear ascending function used by the novel similarity-based mutant vector generation strategy. Nevertheless, that study is out of the scope of this paper, and will be addressed as a future line of work. Despite the above, it is encouraging that DE-SIM was able to provide the best results in 40% of the functions tested. This is particularly noteworthy given that DE-CURRENT is one of the most successful and frequently used DE variants within the literature. As in the case of the first experiment, it is instructive to study how DE-SIM manages diversity in comparison to DE-CURRENT. Figure 2 shows, for each of the problems addressed, the evolution of the mean DCN considering schemes DE-SIM and DE-CURRENT. In general terms, it can be observed that the DE-CURRENT approach preserves a less diversity during the whole run compared to DE-SIM for those cases where the former statistically outperformed the latter in terms of the quality of the solutions achieved. The above confirms our hypothesis regarding the requirements of this particular test suite, in that schemes that mainly promote intensification (such as DE-CURRENT), are able to attain better performance.

For those test functions where DE-SIM provided the best results at the end of the runs, it can be observed that it is able to induce a proper balance between diversification and intensification at each stage of the search. Clear examples that illustrate the above are test cases f_1, f_2 and f_{12}, where DE-SIM started the execution with a diverse population, and as the run progressed, the population gradually converged, becoming less diverse. Exceptions to the aforementioned behaviour are apparent for functions f_3, f_6 and f_{10}. We draw a similar conclusion to that given in the first experiment regarding problem f_3: DE-SIM attained better solutions at the end of the runs by preserving higher diversity for the whole execution in comparison to DE-CURRENT. We conclude that promoting exploitation significantly when addressing function f_3 could be counterproductive, since the optimiser seems to converge to local optima prematurely. Similar conclusions apply to f_6 and f_{10}.

5 CONCLUSIONS AND FUTURE WORK

In this paper we have proposed a novel, similarity-based mutant vector generation strategy for DE. Firstly, individuals are ranked according to their similarity in terms of the Euclidean distance in the decision space with regard to the fittest member in the population. Secondly, the selection of individuals for mutation is guided by a newly introduced linear function that adaptively determines what fraction of the ranked population is considered during selection.

We have evaluated the contribution of our approach in the context of explorative and exploitative DE schemes. In this regard, the results demonstrated that the new strategy outperforms the schemes *rand/1* and *current-to-pbest/1* in approximately 73% and 40% of the functions tested, respectively. This comparison validates our hypothesis that the strategy can maintain a suitable balance between exploration and exploitation, resulting in significantly improved performance.

Based on the features of some of the problems analysed in which exploitative schemes provide better performance than explorative ones, the linear ascending function considered by the novel similarity-based mutant vector generation strategy may in future be replaced by other functions that facilitate a faster change from exploration to exploitation. This is a natural future line of research. Furthermore, undertaking a study with longer run-times with DE-SIM and DE-CURRENT could be another promising avenue of research, as sometimes, diversity-based approaches require longer runs to provide noticeable benefits. It would also be interesting to analyse, in the long-term, if DE-SIM provides better solutions at the end of the runs in comparison to DE-CURRENT, because of potential stagnation in local optima of the latter.

ACKNOWLEDGMENTS

This work has been supported by the Spanish Ministry of Economy, Industry and Competitiveness inside the program "I+D+i Orientada a los Retos de la Sociedad" with contract number TIN2016-78410-R.

REFERENCES

[1] Yiqiao Cai and Jiahai Wang. 2013. Differential evolution with neighborhood and direction information for numerical optimization. *IEEE Transactions on Cybernetics* 43, 6 (2013), 2202–2215.
[2] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. 2013. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)* 45, 3 (2013), 35.
[3] 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.
[4] Wenyin Gong and Zhihua Cai. 2013. Differential evolution with ranking-based mutation operators. *IEEE Transactions on Cybernetics* 43, 6 (2013), 2066–2081.
[5] G. Karafotias, M. Hoogendoorn, and A. E. Eiben. 2015. Parameter Control in Evolutionary Algorithms: Trends and Challenges. *IEEE Transactions on Evolutionary Computation* 19, 2 (April 2015), 167–187.

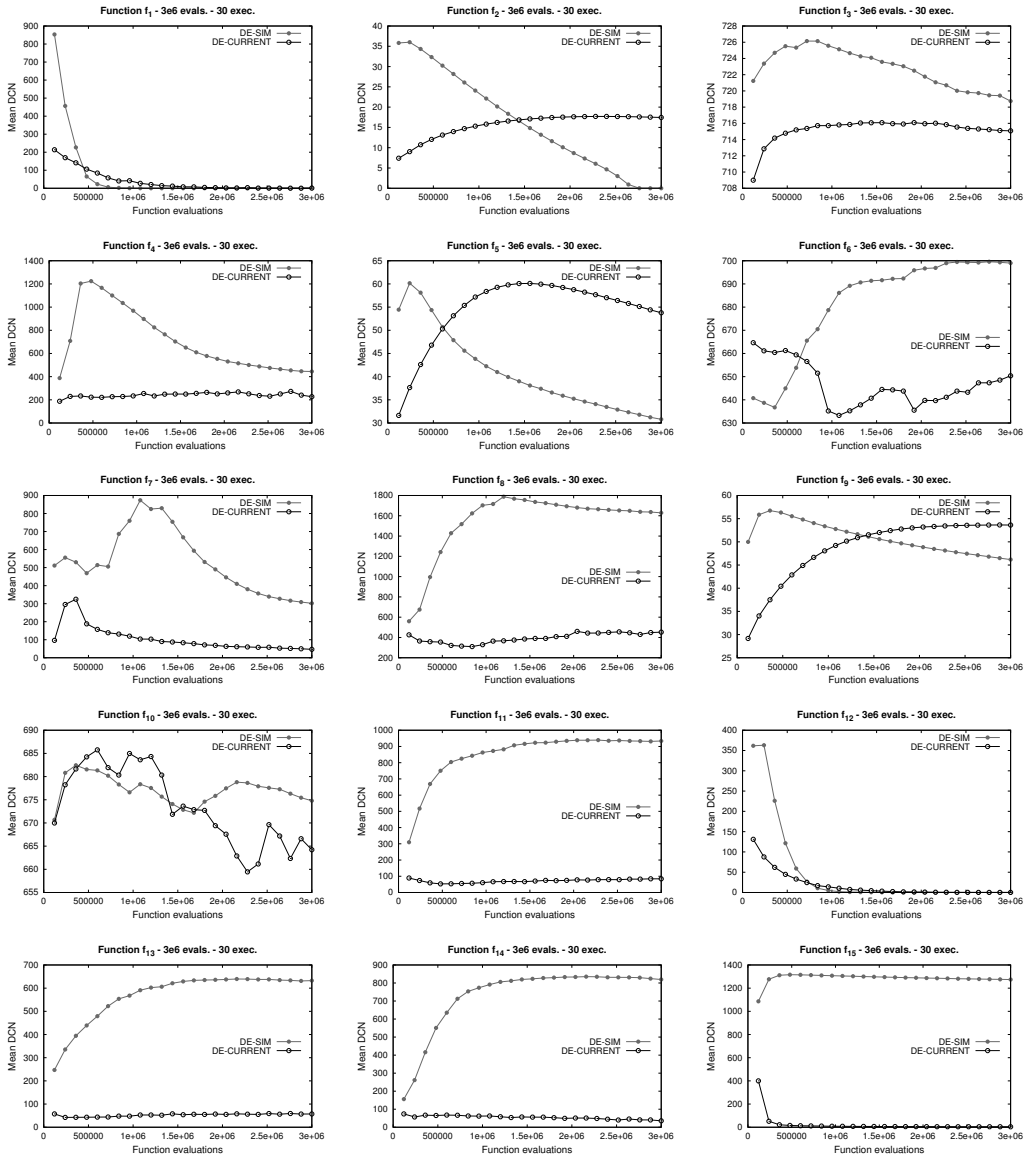

Figure 2: Evolution of the mean distance to the closest neighbour (DCN) for schemes DE-SIM and DE-CURRENT

[6] Coromoto León, Gara Miranda, and Carlos Segura. 2009. METCO: a parallel plugin-based framework for multi-objective optimization. *International Journal on Artificial Intelligence Tools* 18, 04 (2009), 569–588.

[7] X Li, K Tang, M Omidvar, Z Yang, and K Qin. 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.

[8] Eduardo Segredo, Ben Paechter, Carlos Segura, and Carlos I. González-Vila. 2018. On the comparison of initialisation strategies in differential evolution for large scale optimisation. *Optimization Letters* 12, 1 (01 Jan 2018), 221–234.

[9] Carlos Segura, Carlos A. Coello Coello, Eduardo Segredo, and Arturo Hernández Aguirre. 2016. A Novel Diversity-Based Replacement Strategy for Evolutionary Algorithms. *IEEE Transactions on Cybernetics* 46, 12 (Dec 2016), 3233–3246.

[10] Carlos Segura, Carlos A. Coello Coello, Eduardo Segredo, and Coromoto León. 2015. On the adaptation of the mutation scale factor in differential evolution. *Optimization Letters* 9, 1 (2015), 189–198.

[11] 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.

[12] Gregorio Toscano, Ricardo Landa, Giomara Lárraga, and Guillermo Leguizamón. 2017. On the use of stochastic ranking for parent selection in differential evolution for constrained optimization. *Soft Computing* 21, 16 (2017), 4617–4633.

[13] Josef Tvrdík, Radka Poláková, Jiří Veselský, and Petr Bujok. 2013. Adaptive Variants of Differential Evolution: towards Control-Parameter-Free Optimizers. In *Handbook of Optimization: From Classical to Modern Approach*, Ivan Zelinka, Václav Snášel, and Ajith Abraham (Eds.). Springer, Berlin, Heidelberg, 423–449.

[14] Qingzheng Xu, Lei Wang, Na Wang, Xinhong Hei, and Li Zhao. 2014. A review of opposition-based learning from 2005 to 2012. *Engineering Applications of Artificial Intelligence* 29 (2014), 1 – 12.

[15] J. Zhang and A. C. Sanderson. 2009. JADE: adaptive Differential Evolution With Optional External Archive. *IEEE Transactions on Evolutionary Computation* 13, 5 (Oct 2009), 945–958.