Optim Lett
DOI 10.1007/s11590-017-1107-z

SHORT COMMUNICATION

On the comparison of initialisation strategies in differential evolution for large scale optimisation

Eduardo Segredo[1] · **Ben Paechter**[1] · **Carlos Segura**[2] · **Carlos I. González-Vila**[3]

Received: 24 March 2016 / Accepted: 6 January 2017

Abstract *Differential Evolution* (DE) has shown to be a promising global optimisation solver for continuous problems, even for those with a large dimensionality. Different previous works have studied the effects that a population initialisation strategy has on the performance of DE when solving large scale continuous problems, and several contradictions have appeared with respect to the benefits that a particular initialisation scheme might provide. Some works have claimed that by applying a particular approach to a given problem, the performance of DE is going to be better than using others. In other cases however, researchers have stated that the overall performance of DE is not going to be affected by the use of a particular initialisation method. In this work, we study a wide range of well-known initialisation techniques for DE. Taking into account the best and worst results, statistically significant differences among considered initialisation strategies appeared. Thus, with the aim of increasing the probability of appearance of high-quality results and/or reducing the probability

✉ Eduardo Segredo
e.segredo@napier.ac.uk

Ben Paechter
b.paechter@napier.ac.uk

Carlos Segura
carlos.segura@cimat.mx

Carlos I. González-Vila
cigonzalez@iter.es

1 School of Computing, Edinburgh Napier University, Edinburgh, Scotland, UK

2 Área de Computación, Centro de Investigación en Matemáticas, Callejón Jalisco s/n, Mineral de Valenciana, Guanajuato 36240, Mexico

3 Instituto Tecnológico y de Energías Renovables (ITER), Santa Cruz de Tenerife, Spain

Published online: 19 January 2017

of appearance of low-quality ones, a suitable initialisation strategy, which depends on the large scale problem being solved, should be selected.

Keywords Differential evolution · Initialisation strategies · Large scale continuous optimisation

1 Introduction

Differential Evolution (DE) is one of the most widely used meta-heuristics to deal with continuous optimisation problems [17]. Due to its simplicity and efficiency, it has not only been applied to benchmark problems, but also to a wide range of real-world applications regarding electrical and power systems, robotics, and bio-informatics, among others [2]. Moreover, DE and its variants have usually been one of the best performing approaches in different contests, such as the competition on *Large Scale Global Optimisation* (LSGO) organised in different editions of the *Congress on Evolutionary Computation* (CEC) [10].

Regarding *large scale* problems, i.e., problems with a large dimensionality, typically more than 100 decision variables [6], a significant number of works have tried to modify different aspects of DE for searching the vast decision space in a more efficient way [2]. For instance, in [12], a novel proposal that groups dependent decision variables in different sets was combined with DE, being the latter responsible for optimising each set of variables separately. Another work proposed a linearly scalable exponential crossover operator that provided promising results considering a recent set of scalable benchmarks [19]. Recently, two novel schemes that improve the trial vector generation strategy of DE were proposed, which showed to increase the performance of that algorithm when dealing with large scale problems [13]. Finally, the analysis of different strategies for initialising the population of DE with the aim of improving its performance with large scale optimisation problems has gained a noticeable popularity in recent years [5,18].

Some controversies have arisen concerning the benefits that a particular initialisation strategy, applied together with DE, might provide when solving large scale problems [5]. The common belief is that some initialisation strategies can improve the performance of DE for solving problems with a high dimensionality. For instance, the main conclusion given in [6] is that, in opposition to the application of basic random number generators as initialisation strategies, other more advanced methods should be considered in order to increase the performance of DE when dealing with large scale problems, with the most suitable initialiser depending on the problem at hand. In a more recent work [5] however, authors claimed that the initialisation approach does not significantly affect the performance of DE. In that paper, the behaviour of several advanced initialisation methods with different features was analysed when dealing with a set of large scale problems. Those initialisation strategies were combined with the best performing configuration found for one of the most widely used DE variants. Although a few differences among initialisation techniques appeared in some cases when functions were analysed separately, all initialisation approaches performed in a statistically similar fashion taking into account the test suite as a whole.

In the current work, we try to shed some light on the above, by performing a novel study which consists of analysing the behaviour of a wide range of initialisation strategies in the overall, best, and worst cases. Those initialisation mechanisms are applied with the aforementioned best parameterisation of DE to the same set of large scale problems considered in [5]. We show that initialisation strategies present a considerably larger number of statistically significant differences for the best and worst cases in comparison to the overall case, and therefore, a proper mechanism should be selected with the aim of increasing the probability of appearance of high-quality results and/or reducing the probability of appearance of low-quality ones, especially in cases where a high number of executions is not feasible.

The rest of this paper is organised as follows. DE and the particular variant applied herein are described in Sect. 2. Section 3 is focused on introducing the initialisation strategies considered for our study. Afterwards, in Sect. 4, the different experiments conducted are exposed, together with their discussion. Finally, some conclusions and lines of future work are shared in Sect. 5.

2 Differential evolution

DE is a stochastic direct search method especially suited for continuous global optimisation [17]. In DE, the decision variables of a given problem are defined by a vector $\mathbf{X} = [x_1, x_2, \ldots, x_i, \ldots, x_D]$, being D the number of decision variables or the dimensionality of the problem, and every $x_{i=1\ldots D}$ a real number. As we previously mentioned, the term large scale problems is used to refer to those optimisation problems with a large dimensionality, typically $D > 100$. The quality of each vector $\mathbf{X}$ is given by the objective function $f(\mathbf{X})(f : \Omega \subseteq \mathbb{R}^D \rightarrow \mathbb{R})$. The goal of the global optimisation, considering a minimisation problem, is thus to find a vector $\mathbf{X}* \in \Omega$ where $f(\mathbf{X}*) \leq f(\mathbf{X})$ holds for all $\mathbf{X} \in \Omega$. In the particular case of box-constrained continuous optimisation problems, the feasible region Ω is defined by particular values for the lower (a_i) and upper (b_i) bounds of each variable, i.e., $\Omega = \prod_{i=1}^{D}[a_i, b_i]$.

Taking into account the most widely used nomenclature for DE [17], i.e., DE/*x*/*y*/*z*, where x is the vector to be mutated, y defines the number of difference vectors used, and z indicates the crossover scheme, in this work we applied the approach DE/rand/1/bin. We selected this variant due to its simplicity and popularity. The operation of this DE variant is as follows. First of all, a population $P = [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_j, \ldots, \mathbf{X}_{NP}]$ with NP individuals, also called vectors in the field of DE, is initialised by using a particular strategy. Each individual comprises D decision variables. The value of the decision variable i belonging to the individual $\mathbf{X}_j$ is denoted by $x_{j,i}$. Then, successive iterations are evolved by executing the following steps. For each vector $\mathbf{X}_j$ in the current population, called *target vector*, a new *mutant vector* $\mathbf{V}_j$ is created using a *mutant vector generation strategy*. Several mutant vector generation strategies have been devised [2]. In our case, we applied the rand/1 scheme, which is probably the most popular one. The mutant vector $\mathbf{V}_j$ for target vector $\mathbf{X}_j$ is thus created as shown in Eq. 1, where r_1, r_2, and r_3 are mutually exclusive integers chosen at random from the range $[1, NP]$. Furthermore, they are all different from the index j. The *mutation scale factor* F allows the exploration and exploitation abilities of DE to be balanced.

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

After applying the mutant vector generation strategy, the mutant vector is combined with the target vector to generate the *trial vector* $\mathbf{U}_j$ through a crossover operator. The combination of the mutant vector generation strategy and the crossover operator is usually referred to as the *trial vector generation strategy*. The most commonly applied operator for combining the target and mutant vectors—and the one considered in this paper—is the *binomial crossover* (*bin*). The crossover operation is controlled by means of the crossover rate CR. The binomial crossover generates a trial vector as shown in Eq. 2. A uniformly distributed random number in the range [0, 1] is given by $rand_{j,i}$, and $i_{rand} \in [1, 2, \ldots, D]$ is an index selected in a random way that ensures that at least one variable is propagated from the mutant vector to the trial one. For the remaining cases, the probability of the variable being inherited from the mutant vector is CR. Otherwise, the variable of the target vector is taken into consideration.

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

The trial vector generation strategy, as described above, might generate vectors outside the feasible region Ω. One of the most widely used schemes is based on randomly reinitialising the infeasible values in their corresponding feasible ranges, and it is the one applied herein. Finally, after generating NP trial vectors, each one is compared against its corresponding target vector. For each pair, the one that minimises the objective function is selected to survive. In case of a tie, in our implementation the trial vector survives.

3 Initialisation strategies for differential evolution

A wide range of initialisation strategies have been proposed in order to improve the results obtained by DE [2,5,18]. In the current work, we compared the same set of initialisation strategies considered in [5], which are introduced herein.

Pseudo-Random Number Generators (PRNGs) and *Chaotic Number Generators* (CNGs) are one of the most frequently used approaches for initialising a population of individuals [11,15]. In the case of PRNGs, one of the most popular methods is *Mersenne Twister* (MT) [9], which is included as a typical PRNG on a large number of programming languages. Particularly, we used the variant that provides a period of 2^{19937} and 623-dimensional equidistribution with 32-bit accuracy. Regarding CNGs, we considered *Tent Map* (TM) [3]. This approach produces a chaotic sequence of numbers uniformly distributed in the range [0, 1], and has shown some benefits, like a higher iterative speed, with respect to other CNGs, such as *Logistic Map* [3].

The aforementioned types of schemes take into account both randomness and uniformity to generate the initial population. There exist other kinds of schemes however, that only consider uniformity, and therefore, are usually deterministic. From among those strategies, we applied the methods *Sobol Set* (SS) [1] and *Good Lattice Points*

(GLP) [16], which are able to provide a set of points well distributed in the decision space.

Finally, *Opposition-based Learning* (OBL) mechanisms as initialisation methods for DE have gained a significant popularity in recent years [18]. Instead of considering randomness and/or uniformity, OBL generates an initial population and calculates the opposite one with the aim of selecting the fittest individuals from both populations as the starting set. There are different variants of OBL schemes [18]. In addition to the approaches considered in [5], i.e., OBL and *Quasi-Opposition-based Learning* (QOBL), we also applied *Quasi-Reflection Opposition-based Learning* (QROBL) herein, since a recent work [4] stated that the quasi-reflected opposition individual is more likely to be closer to the optimal solution than the opposition and quasi-opposition individuals.

4 Experimental evaluation

This section is devoted to describe the experiments conducted with the version of DE introduced in Sect. 2 integrated with the different initialisation strategies depicted in Sect. 3.

Experimental method

The approach DE/rand/1/bin, as well as the initialisation strategies considered, were implemented by using the *Meta-heuristic-based Extensible Tool for Cooperative Optimisation* (METCO) [7]. Tests were run on *Teide* High Performance Computing facilities, which are composed of 1100 Fujitsu® computer servers, with a total of 17800 computing cores and 36 TB of memory. Since all experiments used stochastic algorithms, each execution was repeated $numRep = 3 \times 10^3$ times, with the aim of comparing the different initialisation strategies with enough statistical confidence. With respect to the former, comparisons were carried out by applying the following statistical analysis [14]. First, a *Shapiro-Wilk test* was performed to check whether the values of the results followed a normal (Gaussian) distribution or not. 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

Experiments were carried out using a set of scalable continuous optimisation problems proposed in CEC'13 for its LSGO competition [8]. It is important to remark that this set of functions is the latest benchmark suite provided for large scale global optimisation in the field of the CEC. Consequently, it was also considered for the LSGO competition organised during CEC'15.[1] The set is composed of 15 functions (f_1–f_{15}) with different features: fully-separable functions (category 1: f_1–f_3), partially additively separable functions (category 2: f_4–f_{11}), overlapping functions (category 3: f_{12}–f_{14}), and a non-separable function (category 4: f_{15}). Following the indications given for

[1] We should note that, although special sessions on LSGO were proposed for CEC'14 and CEC'16, the corresponding competitions were not organised.

Table 1 Different parameterisations of the scheme DE/rand/1/bin

Parameter	Value	Parameter	Value
Stopping criterion	3×10^6 evals.	Mutation scale factor (F)	0.5
Population size (NP)	150 individuals	Crossover rate (CR)	0.9
Initialisation strategies	MT, TM, GLP, SS, OBL, QOBL, QROBL	Decision variables (D)	1000
		Decision variables (f_{13}, f_{14})	905

the different editions of the LSGO competition, in the current work, the number of decision variables D was fixed to 1000 for all the aforementioned functions, with the exception of problems f_{13} and f_{14}, where D was fixed to 905 decision variables due to overlapping subcomponents.

Parameters

The experiments conducted applied a common parameterisation for different configurations of the scheme DE/rand/1/bin, which can be observed in Table 1. The only difference among configurations resides on the initialisation strategy used. In previous work [5], a configuration of the scheme DE/rand/1/bin using those parameter values, from among a candidate pool with more than 80 different configurations of that approach, was able to provide the best overall results for problems f_1–f_{15} with 1000 decision variables. That is the reason why we have selected those values for the parameters NP, F, and CR. Finally, rules of the LSGO competition indicate that the stopping criterion has to be fixed to a maximum number of 3×10^6 function evaluations. In order to perform the analyses, DE was applied with each considered initialisation strategy to each of the 15 benchmark problems, thus giving a total number of 3.15×10^5 runs. Following the recommendations given in [5,10], Eq. 3 was applied to assign a seed $s(i)$ to the i-th run of every DE configuration, regardless of the initialisation method.

$$s(i) = i \quad \forall i \in \{1, 2, \ldots, numRep\} \tag{3}$$

Table 2 shows rankings of the considered initialisation strategies when the best 300 and the worst 300 executions, i.e., those with the lowest and highest values of the objective function at the end of the runs, respectively, are taken into account. Results regarding all the executions are also shown. In order to calculate rankings the following steps were performed. First, the number of approaches that a particular strategy statistically outperformed (↑), as well as the number of times that it was statistically outperformed (↓) by the remaining schemes, considering all problems, were calculated by applying the statistical procedure explained at the beginning of the current section. Approach A statistically outperforms scheme B if there exist 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 objective value than B, since we are dealing with minimisation problems. Afterwards, the score assigned to a strategy is given by the difference between the number of schemes it was able to beat and the number of schemes that were able to beat it.

Table 2 Ranking of initialisation strategies considering the best 300 and the worst 300 executions for problems f_1–f_{15}

Set	Best 300				Worst 300				All			
Strategy	↑	↓	Score	Rank	↑	↓	Score	Rank	↑	↓	Score	Rank
OBL	26	8	18	1	26	7	19	1	13	1	12	1
MT	21	9	12	2	17	17	0	2	13	3	10	2
TM	16	14	2	3	18	20	−2	3	12	13	−1	3
QOBL	14	16	−2	4	13	19	−6	5	5	11	−6	5
QROBL	11	20	−9	5	16	20	−4	4	7	10	−3	4
GLP	14	23	−9	5	20	27	−7	6	6	9	−3	4
SS	15	27	−12	6	18	18	0	2	5	14	−9	6

Results are also shown considering all executions

Finally, a ranking is established by sorting strategies in descending order taking into account the scores assigned.

If we consider the whole set of executions, the best performing overall approach was OBL, although followed by MT with a similar score. In this case, it can be observed that scores assigned to both aforementioned approaches were lower than those assigned to the first-ranked scheme in the best and worst cases. This means that, if all executions are taken into account, the number of differences among initialisation strategies (122 cases out of 630) significantly decreases in comparison to the number of differences that appears regarding the best (234 cases) and the worst results (256 cases). Furthermore, since two methods obtained similar scores, no initialisation strategy was able to provide a clear advantage with respect to the remaining ones when considering the set of problems as a whole. As a result, it might seem that the initialisation technique does not affect the overall performance of DE when solving large scale problems. The above agrees with the conclusions given in [5].

Nevertheless, it can be observed that OBL was the best performing initialisation strategy in cases when the best and the worst 300 executions were analysed, since it obtained significantly better scores than the remaining approaches. The method OBL was statistically better in a larger number of cases than the remaining strategies, while it was statistically worse in a lower number of cases when compared to the remaining schemes. By the application of OBL as an initialisation technique, we therefore might increase/reduce the probability of appearance of high-quality/low-quality executions when using DE for solving large scale continuous optimisation problems. A more in-depth analysis of this method however, should be carried out for each problem, with the aim of providing more evidence of its advantages and drawbacks.

4.1 Analysis of the scheme OBL considering all the results

This section focuses on comparing the approach OBL with respect to the remaining strategies when considering all the executions. Table 3 shows, for each problem, the p-values obtained from the statistical comparison between the scheme OBL and the

Table 3 Statistical comparison between OBL and the remaining strategies considering problems f_1–f_{15} and all executions

f	Init.	p-value	Dif.	f	Init.	p-value	Dif.	f	Init.	p-value	Dif.
f_1	MT	1.84e-02	*	f_2	MT	8.58e-01	↔	f_3	MT	6.08e-01	↔
	GLP	8.20e-02	↔		GLP	4.25e-01	↔		GLP	5.11e-01	↔
	SS	6.89e-02	↔		SS	**5.96e-15**	↑		SS	9.45e-01	↔
	TM	4.56e-02	*		TM	2.34e-02	↓		TM	4.50e-01	↔
	QROBL	1.13e-01	↔		QROBL	**1.91e-76**	↑		QROBL	1.23e-01	↔
	QOBL	2.55e-01	↔		QOBL	**3.78e-76**	↑		QOBL	8.03e-01	↔
f_4	MT	8.73e-01	↔	f_5	MT	5.16e-01	↔	f_6	MT	3.59e-01	↔
	GLP	3.85e-01	↔		GLP	5.89e-01	↔		GLP	4.28e-01	↔
	SS	6.16e-01	↔		SS	2.09e-01	↔		SS	4.49e-01	↔
	TM	9.54e-01	↔		TM	3.71e-01	↔		TM	3.37e-01	↔
	QROBL	2.83e-01	↔		QROBL	9.49e-01	↔		QROBL	6.17e-01	↔
	QOBL	6.38e-01	↔		QOBL	9.75e-01	↔		QOBL	3.51e-01	↔
f_7	MT	8.27e-01	↔	f_8	MT	2.13e-01	↔	f_9	MT	5.46e-01	↔
	GLP	4.67e-01	↔		GLP	2.55e-01	↔		GLP	3.61e-01	↔
	SS	4.33e-01	↔		SS	9.22e-01	↔		SS	9.57e-01	↔
	TM	3.28e-01	↔		TM	8.62e-01	↔		TM	5.30e-01	↔
	QROBL	2.70e-01	↔		QROBL	5.55e-01	↔		QROBL	2.81e-01	↔
	QOBL	1.58e-01	↔		QOBL	5.20e-01	↔		QOBL	6.64e-01	↔
f_{10}	MT	8.16e-01	↔	f_{11}	MT	1.51e-01	↔	f_{12}	MT	7.07e-01	↔
	GLP	3.25e-01	↔		GLP	3.67e-01	↔		GLP	**2.92e-07**	↑
	SS	4.90e-01	↔		SS	9.14e-01	↔		SS	**4.68e-08**	↑
	TM	**0.00e+00**	↑		TM	5.36e-01	↔		TM	7.03e-01	↔
	QROBL	7.30e-01	↔		QROBL	9.19e-01	↔		QROBL	**4.03e-07**	↑
	QOBL	9.15e-01	↔		QOBL	1.14e-01	↔		QOBL	**1.10e-03**	↑
f_{13}	MT	9.96e-01	↔	f_{14}	MT	3.74e-01	↔	f_{15}	MT	2.33e-01	↔
	GLP	2.62e-01	↔		GLP	4.73e-01	↔		GLP	**0.00e+00**	↑
	SS	3.25e-01	↔		SS	4.25e-01	↔		SS	**1.26e-214**	↑
	TM	8.02e-01	↔		TM	4.48e-01	↔		TM	**0.00e+00**	↑
	QROBL	2.87e-01	↔		QROBL	5.80e-01	↔		QROBL	**1.58e-12**	↑
	QOBL	7.31e-01	↔		QOBL	4.87e-01	↔		QOBL	**9.05e-08**	↑

Data in boldface show those cases where OBL statistically outperformed other initialisation strategy, i.e., where an ↑ is also shown in columns called "Dif"

remaining approaches. It also shows cases for which OBL was able to statistically outperform other strategy (↑), cases where other strategy outperformed OBL (↓), and cases where statistically significant differences between OBL and the corresponding method did not arise (↔). Finally, in cases where statistically significant differences between OBL and the corresponding scheme appeared, but one approach obtained the lowest mean, while the other one provided the lowest median, an '*' is shown. It can be observed that in 10 out of 15 problems, no statistically significant differences

appeared between OBL and the remaining initialisation strategies. This confirms our previous statement concerning the lack of differences between initialisation methods when the overall results are considered. Generally speaking, there was no strategy that clearly provided better results than the remaining ones regardless of the addressed problem. Despite that, some differences arose in some cases. OBL was better than several approaches when solving functions f_2, f_{10}, f_{12}, and f_{15}, with each one belonging to a different category. For instance, in the case of the non-separable function f_{15}, OBL showed a clear superiority together with MT. Only in the case of function f_2 OBL was statistically worse than another scheme (TM).

4.2 Analysis of the scheme OBL considering the best results

This section is devoted to compare the initialisation scheme OBL in regard to the remaining approaches when the best 300 executions are considered. Results of this analysis are shown in Table 4.

It is important to remark that, taking into account the best 300 executions, differences between OBL and the remaining methods appeared in 11 out of 15 problems, being this a significant increase concerning the previous analysis of the overall results. OBL did not present statistically significant differences with any other approach for functions f_4, f_5, f_9, and f_{11}. At the same time, in 8 out of 15 problems, OBL was able to outperform other strategies, and in 4 out of those 8 functions, it was not worse than any other initialisation strategy. Considering function f_{14}, for example, OBL was statistically better, together with MT and SS, than the remaining schemes. This means that, if users would like to increase the probability of appearance of high-quality results when solving problem f_{14}, they should initialise the population of DE with one of those three strategies. Moreover, depending on the problem being solved, the most suitable initialisation strategy changes. For instance, taking into account function f_8, the best performing scheme was OBL, together with MT and GLP, while in the case of f_7, QOBL provided the best performance.

4.3 Analysis of the scheme OBL considering the worst results

In this section, we carry out a similar analysis than the one exposed in the previous section, but in this case, we compare the strategy OBL with respect to the remaining approaches when considering the worst 300 executions. Results of this study are shown in Table 5.

In the worst case, differences between OBL and the remaining methods appeared in 14 out of 15 problems. As in the best case, this is a significant increase of differences in comparison to the study considering all executions. Only in the case of function f_8, OBL did not present statistically significant differences with any other approach. In 11 out of 15 problems, OBL was able to outperform other strategies. In fact, in 10 out of those 11 functions, it was not worse than any other initialisation strategy. For instance, considering function f_2, OBL was statistically better, together with MT and TM, than the remaining schemes. This means that, in the worst case, DE would attain better results for problem f_2 by applying one of those three strategies. Additionally, depending on

Table 4 Statistical comparison between OBL and the remaining strategies considering problems f_1–f_{15} and the best 300 executions

f	Init.	p-value	Dif.	f	Init.	p-value	Dif.	f	Init.	p-value	Dif.
f_1	MT	3.06e-01	↔	f_2	MT	**4.78e-02**	↑	f_3	MT	**2.96e-02**	↑
	GLP	4.32e-01	↔		GLP	2.96e-01	↔		GLP	1.45e-01	↔
	SS	3.98e-02	↓		SS	**5.23e-11**	↑		SS	**3.90e-04**	↑
	TM	**1.69e-02**	↑		TM	3.41e-07	↓		TM	3.06e-01	↔
	QROBL	**2.95e-03**	↑		QROBL	**1.93e-47**	↑		QROBL	5.78e-02	↔
	QOBL	9.45e-01	↔		QOBL	**1.18e-46**	↑		QOBL	3.79e-01	↔
f_4	MT	7.75e-01	↔	f_5	MT	9.55e-01	↔	f_6	MT	5.80e-02	↔
	GLP	6.88e-02	↔		GLP	5.55e-01	↔		GLP	1.41e-02	↓
	SS	2.83e-01	↔		SS	5.94e-01	↔		SS	6.58e-01	↔
	TM	8.10e-01	↔		TM	9.28e-01	↔		TM	8.47e-01	↔
	QROBL	1.93e-01	↔		QROBL	1.34e-01	↔		QROBL	8.09e-02	↔
	QOBL	4.21e-01	↔		QOBL	1.44e-01	↔		QOBL	9.00e-01	↔
f_7	MT	3.45e-01	↔	f_8	MT	2.16e-01	↔	f_9	MT	4.32e-01	↔
	GLP	7.59e-01	↔		GLP	5.84e-02	↔		GLP	7.32e-01	↔
	SS	5.21e-02	↔		SS	**2.81e-03**	↑		SS	6.85e-01	↔
	TM	4.08e-01	↔		TM	**2.86e-02**	↑		TM	2.35e-01	↔
	QROBL	7.20e-01	↔		QROBL	**4.62e-03**	↑		QROBL	4.70e-01	↔
	QOBL	3.34e-02	↓		QOBL	**2.97e-02**	↑		QOBL	8.28e-01	↔
f_{10}	MT	7.37e-02	↔	f_{11}	MT	3.47e-01	↔	f_{12}	MT	2.97e-02	↓
	GLP	1.31e-01	↔		GLP	1.02e-01	↔		GLP	**3.37e-14**	↑
	SS	6.75e-01	↔		SS	5.33e-01	↔		SS	**2.81e-10**	↑
	TM	9.35e-03	*		TM	6.13e-01	↔		TM	1.87e-02	↓
	QROBL	5.25e-01	↔		QROBL	6.51e-01	↔		QROBL	**3.76e-07**	↑
	QOBL	3.93e-01	↔		QOBL	4.46e-01	↔		QOBL	**1.31e-04**	↑
f_{13}	MT	4.12e-02	↓	f_{14}	MT	9.61e-01	↔	f_{15}	MT	4.46e-01	↔
	GLP	3.37e-02	↓		GLP	**5.42e-05**	↑		GLP	**1.05e-99**	↑
	SS	**3.79e-02**	↑		SS	2.65e-01	↔		SS	**5.45e-90**	↑
	TM	3.08e-01	↔		TM	**1.05e-04**	↑		TM	**1.05e-99**	↑
	QROBL	8.23e-02	↔		QROBL	**6.99e-03**	↑		QROBL	**4.48e-12**	↑
	QOBL	1.06e-01	↔		QOBL	**8.76e-03**	↑		QOBL	**3.63e-09**	↑

Data in boldface show those cases where OBL statistically outperformed other initialisation strategy, i.e., where an ↑ is also shown in columns called "Dif"

the problem being solved, as in the best case, the most suitable initialisation strategy changes. Taking into account function f_9, for example, the best performing scheme was OBL, together with TM, GLP and QROBL, while in the case of f_{14}, SS provided the best performance.

Finally, it is worth mentioning that, if we consider the best and worst cases simultaneously, there exist two problems (f_3 and f_{15}) for which OBL, and other schemes, were the best performing initialisation approaches. The above means that those initialisation strategies allow the probability of appearance of high-quality results to be

Table 5 Statistical comparison between OBL and the remaining strategies considering problems f_1–f_{15} and the worst 300 executions

f	Init.	p-value	Dif.	f	Init.	p-value	Dif.	f	Init.	p-value	Dif.
f_1	MT	**5.99e-03**	↑	f_2	MT	3.64e-01	↔	f_3	MT	5.52e-01	↔
	GLP	3.17e-01	↔		GLP	**3.46e-05**	↑		GLP	9.69e-01	↔
	SS	**1.07e-02**	↑		SS	**7.89e-12**	↑		SS	8.73e-02	↔
	TM	2.78e-01	↔		TM	2.15e-01	↔		TM	**9.76e-04**	↑
	QROBL	5.69e-02	↔		QROBL	**1.14e-46**	↑		QROBL	7.93e-02	↔
	QOBL	**1.44e-03**	↑		QOBL	**6.32e-47**	↑		QOBL	8.97e-01	↔
f_4	MT	2.26e-01	↔	f_5	MT	3.40e-01	↔	f_6	MT	**1.66e-02**	↑
	GLP	**3.07e-02**	↑		GLP	**1.11e-05**	↑		GLP	6.30e-01	↔
	SS	9.43e-01	↔		SS	**6.11e-06**	↑		SS	8.42e-02	↔
	TM	4.84e-01	↔		TM	**2.63e-02**	↑		TM	3.81e-01	↔
	QROBL	8.38e-01	↔		QROBL	7.29e-02	↔		QROBL	**4.82e-02**	↑
	QOBL	5.32e-01	↔		QOBL	7.61e-01	↔		QOBL	6.03e-01	↔
f_7	MT	3.89e-01	↔	f_8	MT	3.06e-01	↔	f_9	MT	**1.39e-02**	↑
	GLP	5.06e-01	↔		GLP	6.97e-01	↔		GLP	1.22e-01	↔
	SS	3.78e-01	↔		SS	5.76e-01	↔		SS	**5.33e-04**	↑
	TM	7.63e-01	↔		TM	8.04e-01	↔		TM	7.74e-02	↔
	QROBL	4.41e-02	↓		QROBL	3.64e-01	↔		QROBL	1.57e-01	↔
	QOBL	2.18e-01	↔		QOBL	6.42e-01	↔		QOBL	**1.34e-02**	↑
f_{10}	MT	1.03e-03	↓	f_{11}	MT	7.09e-02	↔	f_{12}	MT	1.15e-01	↔
	GLP	4.91e-02	↓		GLP	7.72e-01	↔		GLP	**3.40e-04**	↑
	SS	1.76e-02	↓		SS	2.19e-01	↔		SS	**2.81e-03**	↑
	TM	**1.05e-99**	↑		TM	6.34e-02	↔		TM	4.53e-01	↔
	QROBL	6.77e-04	↓		QROBL	1.08e-01	↔		QROBL	1.56e-01	↔
	QOBL	1.65e-01	↔		QOBL	**2.24e-05**	↑		QOBL	2.17e-01	↔
f_{13}	MT	9.64e-01	↔	f_{14}	MT	1.76e-01	↔	f_{15}	MT	4.90e-01	↔
	GLP	5.48e-03	↓		GLP	5.55e-01	↔		GLP	**1.56e-69**	↑
	SS	8.38e-01	↔		SS	1.72e-03	↓		SS	**2.10e-10**	↑
	TM	8.04e-01	↔		TM	1.02e-01	↔		TM	**2.75e-90**	↑
	QROBL	8.95e-01	↔		QROBL	7.64e-02	↔		QROBL	**4.78e-14**	↑
	QOBL	6.62e-02	↔		QOBL	2.05e-01	↔		QOBL	**7.86e-10**	↑

Data in boldface show those cases where OBL statistically outperformed other initialisation strategy, i.e., where an ↑ is also shown in columns called "Dif"

increased, and at the same time, are able to reduce the probability of appearance of low-quality results when solving those two particular functions.

5 Conclusions and future work

Some controversies have arisen in recent years regarding the benefits of using a given initialisation strategy for solving large scale problems with DE. Some works have stated

that certain initialisation mechanisms are able to increase the performance of DE. Other authors however, have claimed that the initialisation strategy does not significantly change the way DE performs.

Analysing the overall case, we showed that differences among the initialisation strategies considered were not statistically significant for a considerable number of problems. Bearing the above in mind, it might seem that the initialisation technique does not generally affect the performance of DE when solving large scale problems. However, when studying the best and worst cases, which had not been previously analysed, significant differences appeared among initialisation mechanisms, with OBL being the best performing approach for a wide range of problems. As a result, with the aim of increasing the probability of appearance of high-quality results and/or decreasing the probability of appearance of low-quality solutions when DE is used to solve large scale problems, a suitable initialisation strategy, which depends on the problem at hand, should be selected. For those cases where we do not have enough *a priori* information about the problem being solved, for instance, when dealing with black-box optimisation problems, OBL seems to be a promising scheme. Finally, we should note that the above is even more important in those scenarios where only a few executions can be performed, for example, when solving large scale real-world problems with time-consuming evaluation functions.

Although QOBL and QROBL are extensions of OBL, the experimental evaluation carried out in this work showed that they were not able to provide better results than the latter. It is likely that this is because different variants of DE, with different parameter values, components, and stopping criteria, are studied depending on the considered work. Another possibility might be the dimensionality used for defining the problems. Due to the above reasons, it would be interesting to study whether more sophisticated initialisation strategies, such as QROBL and QOBL, among others, are able to provide some benefits with respect to the traditional OBL scheme. Another line of future work would be to analyse the behaviour of different mechanisms based on OBL, as well as other initialisation schemes, in the worst and best cases, by combining them with different variants of DE applied to several sets of problems. This might allow the conclusions extracted in this work to be generalised.

Finally, we should note that another factor that could affect the quality of the algorithm initialisation is the population size. As it was already mentioned, a significant number of configurations of one of the most widely used DE variants were analysed in a previous work considering the overall case. Those configurations were obtained by combining different values for the parameters of DE, including the population size. The best performing configuration was applied with one of the biggest population sizes considered. In opposition to the resolution of problems with lower dimensionalities, where smaller populations perform better, the analyses carried out in the aforementioned work concluded that an increase of the population size to some threshold values is more suitable when dealing with large scale optimisation. Nevertheless, those studies also concluded that differences among the initialisation strategies considered were not statistically significant when using larger population sizes. Bearing the above in mind, and considering that the said best performing configuration was also applied in the current work, the effects that the population size, together with the initialisation strategy, may have on the quality of the algorithm initialisation, were not analysed

herein. Since the overall case has already been analysed however, it would be very interesting to carry out a study about the performance of DE through the combination of different initialisation strategies and different population sizes taking into account the best and worst cases.

Acknowledgements The authors wish to acknowledge the contribution of Teide High-Performance Computing facilities to the results of this research. TeideHPC facilities are provided by the Instituto Tecnológico y de Energías Renovables (ITER, S.A). URL: http://teidehpc.iter.es.

References

1. Bratley, P., Fox, B.L.: Algorithm 659: implementing sobol's quasirandom sequence generator. ACM Trans. Math. Softw. **14**(1), 88–100 (1988)
2. Das, S., Mullick, S.S., Suganthan, P.: Recent advances in differential evolution—-an updated survey. Swarm Evolut. Comput. **27**, 1–30 (2016)
3. Dong, N., Wu, C.H., Ip, W.H., Chen, Z.Q., Chan, C.Y., Yung, K.L.: An opposition-based chaotic GA/PSO hybrid algorithm and its application in circle detection. Comput. Math. Appl. **64**(6), 1886–1902 (2012)
4. Ergezer, M., Simon, D.: Mathematical and experimental analyses of oppositional algorithms. IEEE Trans. Cybern. **44**(11), 2178–2189 (2014)
5. Kazimipour, B., Li, X., Qin, A.: Effects of population initialization on differential evolution for large scale optimization. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 2404–2411 (2014). doi:10.1109/CEC.2014.6900624
6. Kazimipour, B., Li, X., Qin, A.K.: Initialization methods for large scale global optimization. In: 2013 IEEE Congress on Evolutionary Computation (CEC), pp. 2750–2757 (2013). doi:10.1109/CEC.2013.6557902
7. León, C., Miranda, G., Segura, C.: METCO: a parallel plugin-based framework for multi-objective optimization. Int. J. Artif. Intell. Tools **18**(4), 569–588 (2009)
8. Li, X., Tang, K., Omidvar, M., Yang, Z., Qin, K.: 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, (2013)
9. Matsumoto, M., Nishimura, T.: Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Model. Comput. Simul. **8**(1), 3–30 (1998)
10. Qin, A., Li, X.: Differential evolution on the CEC-2013 single-objective continuous optimization testbed. In: 2013 IEEE Congress on Evolutionary Computation (CEC), pp. 1099–1106 (2013). doi:10.1109/CEC.2013.6557689
11. Rajashekharan, L., Shunmuga Velayutham, C.: Is Differential Evolution Sensitive to Pseudo Random Number Generator Quality?—An Investigation. In: Berretti, S., Thampi, S. M., Srivastava., Praveen Ranjan (eds) Intelligent systems technologies and applications: vol 1, pp. 305–313. Springer International Publishing, Cham (2016)
12. Sayed, E., Essam, D., Sarker, R., Elsayed, S.: Decomposition-based evolutionary algorithm for large scale constrained problems. Inf. Sci. **316**, 457–486 (2015)
13. Segura, C., Coello, C.A.C., Hernández-Díaz, A.G.: Improving the vector generation strategy of differential evolution for large-scale optimization. Inf. Sci. **323**, 106–129 (2015)
14. Segura, C., Coello, C.A.C., Segredo, E., Aguirre, A.H.: A novel diversity-based replacement strategy for evolutionary algorithms. IEEE Trans. Cybern. **46**(12), 3233–3246 (2015)
15. Skanderova, L., Řehoř, A.: Comparison of Pseudorandom Numbers Generators and Chaotic Numbers Generators used in Differential Evolution. In: Zelinka, I., Suganthan, Ponnuthurai, N., Chen, G., Snasel, V., Abraham, A., Rössler, O (eds) Nostradamus 2014: prediction, modeling and analysis of complex systems, pp. 111–121. Springer International Publishing, Cham (2014)

16. Sloan, I.H.: Lattice methods for multiple integration. J. Comput. Appl. Math. **12**, 131–143 (1985)
17. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. of Glob. Optim. **11**(4), 341–359 (1997)
18. Xu, Q., Wang, L., Wang, N., Hei, X., Zhao, L.: A review of opposition-based learning from 2005 to 2012. Eng. Appl. Artif. Intell. **29**, 1–12 (2014)
19. Zhao, S.Z., Suganthan, P.N.: Empirical investigations into the exponential crossover of differential evolutions. Swarm Evolut. Comput. **9**, 27–36 (2013)