Nonlinear Dynamics manuscript No.
(will be inserted by the editor)

Kinship-based Differential Evolution algorithm for unconstrained numerical optimization

Giovanni Formica · Franco Milicchio*

Received: date / Accepted: date

Abstract We propose a modification of the standard Differential Evolution (DE) algorithm in order to significantly make easier and more efficient standard DE implementations. Taking advantages from chaotic map approaches, recently proposed and successfully implemented for swarm intelligence-based algorithms, our DE improvement facilitates the search for the best population and then the optimal solution. More specifically, we work with a genetic memory that stores parents and grand-parents of each individual (its kin) of the population generated by the DE algorithm. In this way, the new population is carried out not only on the basis of the best fitness of a certain individual, but also according to a good score of its kin. Additionally, we carried out a wide numerical campaign in order to assess the performances of our approach, and validated the results with standard statistical techniques.

Keywords Optimization · Differential Evolution algorithm · Chaotic maps

1 Introduction

Heuristic algorithms are considered robust methods for finding optimum solutions and handling multimodal optimization problems, employed in diverse fields, from engineering to deep-learning. Along with their improved version, named meta-heuristic algorithms, these procedures may be classified as evolutionary, or swarm intelligence.

G. Formica
Dipartimento di Architettura, Roma Tre University, via Madonna dei Monti 40, Roma, Italy
Tel.:+39-06-57336253 — Fax:+39-06-57336265
E-mail: giovanni.formica@uniroma3.it
ORCID 0000-0002-6410-8083

F. Milicchio* (Corresponding author)
Dipartimento di Ingegneria, Roma Tre University, via della Vasca Navale 79, Roma, Italy
Tel.:+39-06-57336253 — Fax:+39-06-57336265
E-mail: franco.milicchio@uniroma3.it
ORCID 0000-0002-4875-4894

One of the most reliable heuristic algorithm is the Differential Evolution (DE), the present work is framed into. DE belongs to the general family of evolutionary/genetic algorithms that proved to be successful on a variety of real world problems from diverse domains of science and technology. Contrary to standard evolutionary or genetic algorithms, DE stands out by distinguishing itself from its peers for its properties: simplicity and *self-referential mutation*. A DE optimizer can be implemented in any programming language, mandating only few lines of code and very few control parameters. Moreover, the algorithm, when optimizing a set of variables, does not require the specification or adaptation of absolute step sizes—self-referential mutation property—basing its convergence properties on a numeric scale factor. Performances of differential evolution, in terms of accuracy, speed, and robustness, are remarkable despite its simplicity if compared to its competitors (for details, please refer to the IEEE CEC[1] conference website).

One of the main competitor branches of heuristic algorithms are the so called swarm intelligence, basing their procedures on biological entities that interact and self-organize—bees, ants, birds. Here we only mention few approaches, such as ant colony optimization (ACO), particle swarm optimization (PSO), and artificial bee colony (ABC); other, less popular swarm optimizers include bacterial foraging optimization (BFO), biogeography-based optimization (BBO), artificial fish swarm optimization (AFSO), firefly algorithm (FA).

It is arduous to provide a complete overview of both types of algorithms, as they are continuously improved with more or less significant modifications. For an up-to-date survey in swarm intelligence algorithms, we refer the reader to [26], while for differential evolution please see [11]. Instead, we briefly remind here few aspects of some of them in order to bring out the novel contributions of the present work.

Optimization procedures require control parameters, whose selection towards the optimal solution profoundly affect the effectiveness of the whole evolution. For instance, genetic algorithms such as DE require an adequate control of its settings such as mutation and crossover rates; particle swarm optimization also relies on cognitive and social parameters, as well as inertia weights. Similarly, artificial bee colony necessitates the specification of the number of bees in the colony, while biogeography-based optimization the probability of habitat modification, mutation probability, habitat elitism parameter, and population size (cf. [34] for a deeper insight on such issue).

Recently, new DE algorithms employ *adaptive* strategies to modify control parameters used in the evolutionary process. For instance JADE [51] and SHADE [43] select a given percentage of the best individuals, employing historical data to drive the process. New control parameters may be also generated by utilizing uniform distributions as in jDE [5], or implementing multi-stategy mutation processes (cf. SaDE [32]).

Chaos theories have been also recently introduced, such as chaotic differential evolution [21,31], chaotic genetic algorithms [24,50], chaotic simulated annealing [27,19], chaothic Jaya [13], chaotic firefly algorithm [16,17], chaotic-bat swarm optimization [15], and chaotic biogeography optimization [35,48].

The application of chaotic theories contributes in three different ways in ameliorating the limitations of standard heuristic optimizers: *i*) random numbers may

[1] See for instance `http://cec2019.org/`.

be substituted by chaotic sequences generated by chaotic maps; *ii*) local search approaches can be implemented via chaotic map functions; *iii*) chaotic maps may be employed in the generation of control parameters.

Chaotic maps have recently been applied successfully to various Differential Evolution approaches. One of the major advantages of applying chaos theory to DE algorithms relies in improving convergence rates [28] and overall results when compared to randomized approaches (cf. [6] and [36]). Moreover, a recent research [39] presented an introspection on various DE algorithms in order to assess the benefits of chaotic maps (for furhter reference, see, *e.g.*, [38] and [37]). Variants of Differential Evolution have been tested with the standard IEEE CEC'15 test suite, among them more recent algorithms such as jDE [5, 7] and SHADE [43, 44], showing that chaotic versions are comparatively highly efficient in optimizing the given objective functions. Additionally, it has been shown that chaotic maps amplify the diversity in the population of a differential evolution optimizer (cf. [39]), as well as in other evolutionary algorithms, as detailed, *e.g.*, in [45] for Particle Swarm Optimization algorithms.

Contribution Taking advantages from chaotic map and genetic approaches, we developed a modification of the Differential Evolution algorithm that drastically facilitates the search for the best population and then the optimal solution. More specifically, we developed a new *mutation* equation that take advantage from chaotic maps, and new *crossover* one considering the *genetic memory* of each individual in the DE population, *i.e.*, its kin up to the second degree (parents and grandparents). Therefore, this novel algorithm selects individuals not only on the basis of the best *fitness*, but also according to its previous generation's scores. An extensive numerical testbed has been performed to underline the performances of our approach, and validated via standard statistical tests.

Outline The manuscript is organized as follows. Section 2 introduces the main aspects of Differential Evolution algorithms, while Section 3 describes in details our kinship-based DE approach. Finally, Section 4 report all numerical tests exhaustively, by providing numerical analyses as well as statistical validation of all the results.

2 Differential Evolution Algorithms

In the problem optimization field, Differential Evolution (DE) is an iterative *evolutionary algorithm*—*i.e.*, inspired by the evolution of biological entities—optimizing a problem given an objective and a quality measure functions. Contrary to other classical optimization methods, however, DE does not assume any a priori condition on the objective function, for instance, it does not require the gradient of the objective function to be know, or even differentiability. This fact renders DE one of the most adaptable and important optimizing algorithm, suitable for non-continuous, non-convex, subject to noise, and multimodal parameter spaces.

The initial Differential Evolution algorithm has been introduced by Storn in 1995 [41, 42], and it spawned a plethora of diverse approaches applied in real-life scenarios (*e.g.*, engineering design [23], medicine [1], material design [14], and economy [9]). DE algorithms can be summarized as follows: given a random set of

candidate solution, the *population*, extracted from a search domain, DE iterates mutating the population by scaling the differences between individuals. After an initialization of the population, the DE algorithm can be designed as an iterative process constituted by three steps, carried out in order: mutation, crossover, and population selection. The optimization process ends when a series of halting criteria are met, *e.g.*, the number of maximum iteration has been reached, or the current best solution is within a given threshold from the previously found one.

This section will briefly outline the idea behind Differential Evolution algorithms, introducing its basic steps and variants, and we refer the reader to [11, 12] for a thorough review of DE approaches. Contrary to other evolutionary algorithms, DE presents itself as a very simple procedure that can be implemented in a straightforward manner, with the original DE being controlled by just three parameters, *i.e.*, a scale factor, a crossover probability threshold, and the population size. Despite the ease of coding a DE algorithm, it sports a unique computational performance and accuracy in the optimization of diverse problems, as shown by its top ranking in the annual IEEE Congress on Evolutionary Computation (CEC) conference series. Different versions have been developed of DE algorithms, for instance by employing parameter adaptivity [51, 20, 32] or population control [44, 49]; here we outline the original differential evolution algorithm as introduced in [41, 42].

Initialization Given a d-dimensional search space Ω^d, a differential evolution algorithm searches for a global optimum by creating an initial population at time $t = 0$ constituted by N individuals represented by d-dimensional vectors:

$$x_i^t = (x_{i,1}^t, ..., x_{i,d}^t)\,,$$

where the components of the i-th individual x_i^t at time t are the d decision variables of the search space.

As variables are usually restricted in Ω^d, being part of physical measures that are naturally bounded, the initialization step takes into account each bound by randomly initializing each j-th component $x_{i,j}^0 = x_{min,j} + rand_{i,j}(x_{max,j} - x_{min,j})$, given $x_{min} = (x_{min,1}, ..., x_{min,d})$ and $x_{max} = (x_{max,1}, ..., x_{max,d})$, the minimum and maximum bounds vectors, and being $rand_{i,j} \in [0, 1] \subset \mathbb{R}$.

Mutation The cornerstone of DE is the population mutation step. This phase generates a new vector $\widetilde{x}_i^t$ corresponding to each individual according to an aleatoric rule. Such rules may vary greatly, for instance, an implementation may choose a "random" approach, selecting the new vector as

$$\widetilde{x}_i^t = x_{r_1}^t + F(x_{r_2}^t - x_{r_3}^t)\,, \tag{1}$$

where r_1, r_2, and r_3 are distinct random natural numbers in $[1, N] \subset \mathbb{N}$, and F is a (real) scaling factor. Another method may choose a "best" mutation, by selecting the best individual from the population—the one with the lowest objective function—and combining its components with other two random ones, *i.e.*, $\widetilde{x}_i^t = x_{best}^t + F(x_{r_1}^t - x_{r_2}^t)$.

Crossover Mimicking biological chromosomes, DE performs a crossover phase where new individuals $\widetilde{x}_i^t$ blend their genetic information, *i.e.*, their components, with the old ones; similarly to the mutation step, crossover relies on randomness and on a *crossover probability* parameter $Cr \in [0,1] \subset \mathbb{R}$. The crossover step then generate new "hybrid" individuals $\widetilde{x}'_i$ by selecting at least one random component from $\widetilde{x}_i$, in other words:

$$\widetilde{x}'_{i,j} = \begin{cases} \widetilde{x}_{i,j}\,, & j = K \text{ or } rand_{i,j} \leq Cr \\ x_{i,j}\,, & \text{otherwise}\,. \end{cases} \tag{2}$$

In the above equation $K \in [1, N] \subset \mathbb{N}$ is a random integer number that ensures that the new individual has at least one component from the new one.

Selection The final step in a DE algorithm is the selection of the new population based on their objective function evaluation. Hence, the new individual x_i^{t+1} at time $t+1$ is indeed chosen as the new one only if its *fitness* is better than the old one, *i.e.*,

$$x_i^{t+1} = \begin{cases} \widetilde{x}'_i\,, & f(\widetilde{x}'_i) \leq f(x_i) \\ x_i\,, & \text{otherwise}\,, \end{cases} \tag{3}$$

with f being the objective function to be minimized. For a general overview of the algorithm functioning, see the flowchart in Fig. 1.

3 Kinship-based DE Algorithm

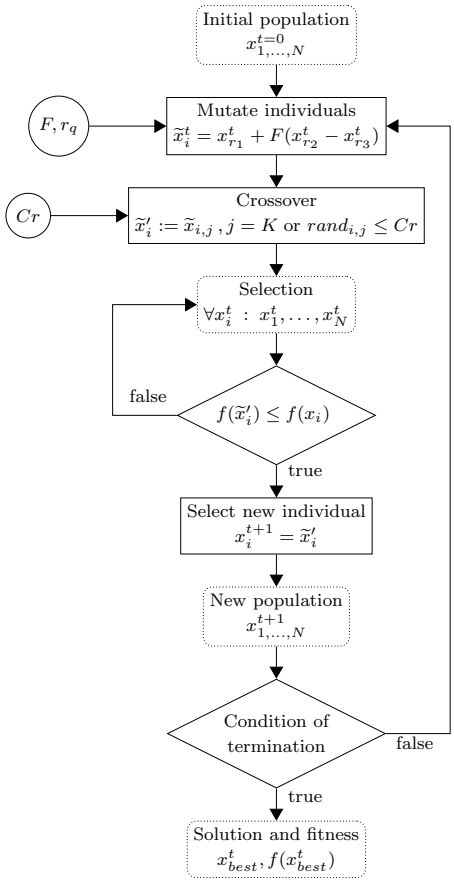
As outlined in Section 2, the control parameters F and Cr are at the heart of a differential evolution, providing several different strategies according to the rules of mutation and crossover phases. As such, literature has explored greatly diverse strategies to adapt F and Cr to different scenarios. Recently, the population number N gained momentum as a possible additional control parameter. This section will briefly outline some possible strategies concerning the population generation, and will introduce our *kinship*-based approach to differential evolution, where parental relationships are exploited in order to boost the convergence speed and avoid local minima.

3.1 Hybrid Algorithms

The mutation phase generates a novel individual based on random selection and the scaling factor F, while the crossover step transforms the population ensuring that it will differ, in each individual, from the older generation. However, the population size in a classical DE remains identical. Recent classical variants propose to sub-sample, cluster, or mutate the population size. For instance, in [6] the population is halved at each time step: the old population is divided into two halves, and a new individual is selected according to the fitness evaluation of two old ones, one from each partition.

One of the major divergent variants of DE proposes the *hybridization* of Differential Evolution algorithms with other optimization techniques, such as Artificial

Fig. 1 Flowchart of a classic Differential Evolution algorithm.

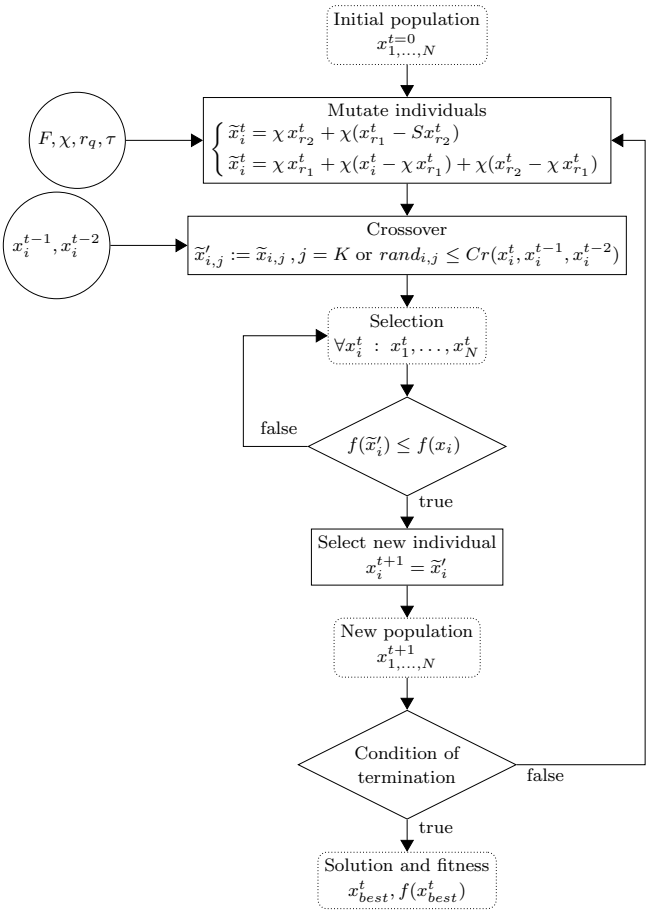


Bee Colony (ABC), Particle Swarm Optimization (PSO), and other approaches [4], yielding better results with respect to their corresponding conventional algorithms. Our kinship algorithm may be viewed under the umbrella of an hybrid optimization technique, as detailed in the ensuing paragraphs.

3.2 The Kinship Differential Evolution

Standard DE optimizers may suffer from stagnation, yielding a population that does not balance exploitation and exploration, *i.e.*, converging swiftly and searching different domain regions, respectively. In order to overcome this issue, we deviate from standard differential evolution algorithms by *i*) employing a *chaotic map* in lieu of pseudo-random numbers in the mutation step, *ii*) considering the *ancestry* of a population in the crossover phase. The ensuing paragraphs will detail our algorithm, pictured in Fig. 2.

Fig. 2 Flowchart of the Kinship Differential Evolution algorithm.



Chaotic Map As outlined in Section 2, random numbers are employed in DE algorithms to generate the initial population, as well as produce a new spawn of individuals from previous ones. A better alternative to PRNG (pseudo-random number generators) present in all computer languages can be found in *chaotic maps* [29,25], where the nonlinearity in the dynamical system yields an elevated sensitivity to initial condition variations; the use of chaotic maps in optimization procedures have enhanced standard (non-chaotic) algorithms, such as DE, ABC, GA, and HS (cf. [31,2,24], and [3]). Recently, chaos theory has been applied to the Jaya algorithm [33], proposing the C-Jaya variation [13] that employs the Chebyshev map:

$$\begin{cases} y^1_{i+1} &= \cos\left(k \arccos(y^2_i)\right) \\ y^2_{i+1} &= 16\,(y^1_i)^5 - 20\,(y^1_i)^3 + 5\,y^1_i\,, \end{cases} \tag{4}$$

with the initial conditions $y_i^1, y_i^2 \in [-1, 1]$, $k > 1$, and (y_0^1, y_0^2). Our differential evolution employs the above chaotic map in order to ensure a balanced diversity in the population.

Mutation The mutation step produces new trial individuals that, with the crossover phase, will generate a new population that will be compared to the old one by means of the objective function. In our case, the mutation equation (1) has been modified to exploit the chaotic map in the mutation equations as follows. As argued in several works employing chaotic maps (see, *e.g.* [47,28,46,13]), we propose two mutation equations in order to balance exploitation and exploration: the former would lead the optimizer to search for local solutions only, while the latter to degenerate into a random search. Given a random number a_i for each individual and a threshold τ, we mutate an individual as

$$\begin{cases} \widetilde{x}_i^t = \chi\, x_{r_2}^t + \chi(x_{r_1}^t - S\, x_{r_2}^t)\,, & a \geq \tau \\ \widetilde{x}_i^t = \chi\, x_{r_1}^t + \chi(x_i^t - \chi\, x_{r_1}^t) + \chi(x_{r_2}^t - \chi\, x_{r_1}^t)\,, & a < \tau\,, \end{cases} \tag{5}$$

where $\chi = chaos_{1,2}$ is the absolute value of one of the two chaotic variables (y^1, y^2), generated by the chaotic map (4) and then selected randomly; (r_1, r_2) are random integer numbers selecting randomly two individuals in the population; S is a scaling factor equal to $S = \chi + 1 > 1$.

Note that the threshold condition, as successfully implemented in recent works on particle swarm optimization [45], is governed by $a_i = rand(0, 1) \in \mathbb{R}$ and by $\tau = t/(\xi\, t_{\max})$, accounting for the current number of iterations t, with $\xi < 1$ an assigned positive coefficient. When t reaches the iteration threshold $\xi\, t_{\max}$ (*i.e.*, $\tau \geq 1$) the second mutation equation $(5)_2$ acts, whereas up to the iteration level $\xi\, t_{\max}$ it is more probable that the first mutation equation $(5)_1$ is selected. In particular, in $(5)_1$ the exploration is favoured by the scaling factor S, while ξ adjustes the balance between global and local search.

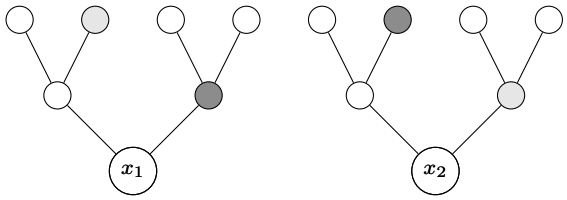
Crossover and Selection Last in the generation phases, the crossover step performs a random selection of the "genes" of individuals in order to generate the new spawn at time $t + 1$. Our kinship approach to the crossover phase mimics the biological avoidance of *inbreeding*, *i.e.*, selecting individuals that are not closely related. In biology, genetic disorders are carried by individuals with two copies of a recessive pathological genetic mutation [18]. In our DE algorithm, we exploit the kinship relationship to exclude unwanted genes in the next population. In details, the *crossover probability* is a function of the *kinship coefficient* κ:

$$\widetilde{x}'_{i,j} = \begin{cases} \widetilde{x}_{i,j}\,, & j = K \text{ or } rand_{i,j} \leq Cr(\kappa) =: {}^1\!/_8(6 - \kappa) \\ x_{i,j}\,, & \text{otherwise}\,. \end{cases} \tag{6}$$

The kinship coefficient κ effectively relates individuals x_i and x_j by their ancestry, hence genes owing to their origin, and is defined as $\kappa \in [0, 6] \subset \mathbb{N}$: $\kappa = n_{i,j}^p + n_{i,j}^g$ is the total number of ancestors in common between two individuals (see Fig. 3), up to the second-order, counting $n_{i,j}^p$ "parents" $(\max n_{i,j}^p = 2\,, \forall i, j)$ and $n_{i,j}^g$ "grandparents" $(\max n_{i,j}^g = 4\,, \forall i, j)$.

In order to keep track of common ancestry, we store only the individuals with the additional property that univocally identifies the ancestors, *i.e.*, six integer

Fig. 3 Two individuals x_1 and x_2 with kinship coefficient $\kappa = 2$; nodes in light and dark gray indicate their common ancestors.



numbers: in fact, as one can see from Equation (6), crossover is computed within the current population, and the kinship coefficient κ needs no reference to the actual older parameters, and can be simply computed with a group of six unique identifiers, one per each ancestor. Such additional storage, however, is for all practical purposes negligible, and accounts for $6N \log_2(N)$ bits. For instance, if we allow the maximum number of individuals among all generations to be $2^{24} \approx 1.7 \cdot 10^7$, we need about 300 MB of additional RAM with respect to a standard DE implementation.

Finally, the new individual is selected for the next generation following (3), *i.e.*, if and only if its fitness function is preferable to the old one, thus the last phase is in line with standard DE approaches.

4 Results and Comparison

This section is devoted to illustrate numerical performances of the Kinship DE optimization algorithm, in comparison with several popular optimizers.

In details, we chose standard benchmark functions employed to test optimization algorithms (cf. [22,13], and [34]); these functions span different groups such as unimodal, multimodal, and their rotated version, as illustrated in details in Table 1. All tests were conducted on different problem dimensions, namely 10, 30, and 100-dimensional optimization, fixing the population sizes for all dimensions to 20.

We then compared our Kinship DE optimizer with respect to the following algorithms: PSO (Particle Swarm Optimization), DE (Standard Differential Evolution), HS (Harmony Search), ABC (Artificial Bee Colony), TLBO (Teaching-learning-based optimization), Jaya, and C-Jaya (Chaotic Jaya). Such a selection of optimizers available in the literature revealed to be significant as testbed methods in this research field; in particular, we report here the outcomes recently provided in [10,22,13], along with our results.

4.1 Convergence Evaluation

The convergence analysis of all algorithms for dimensions 10, 30, and 100 are shown respectively in Tables 2, 3, and 4, where we show the mean, the standard deviation, and the standard error of the mean for all functions to be optimized. A statistical analysis was performed in order to obtain a fair comparison between

Table 1 Test functions and their parameters; rotated versions are all intended with $y = M \times x$. Acceptance rates are set to 10^{-5}, except for f_1 (10^{-2}), and f_5, f_{12}, f_{14} (50).

Formula	Range	Function
$f_1(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]$	Sphere
$f_2(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	$[-100, 100]$	Quadric
$f_3(x) = \sum_{i=1}^{D} (ix_i)^2$	$[-100, 100]$	Sum Squares
$f_4(x) = \sum_{i=1}^{D} x_i^2 + (\sum_{i=1}^{D} 0.5x_i)^2 + (\sum_{i=1}^{n} 0.5x_i)^4$	$[-10, 10]$	Zakharov
$f_5(x) = \sum_{i=1}^{D-1} [100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2]$	$[-2.048, 2.048]$	Rosenbrock
$f_6(x) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi x_i)) + 20 + e$	$[-32.768, 32.768]$	Ackley
$f_7(x) = \sum_{i=1}^{D} (x_i^2 - 10 \cos(2\pi x_i + 10))$	$[-5.12, 5.12]$	Rastrigin
$f_8(x) = \sum_{i=1}^{D} (\sum_{k=0}^{20} [0.5^k \cos(2\pi 3^k (x_i + 0.5))]) - D \sum_{k=0}^{20} [0.5^k \cos(2\pi 3^k 0.5)]$	$[-0.5, 0.5]$	Weierstrass
$f_9(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600, 600]$	Griewank
$f_{10}(x) = \sum_{i=1}^{D} i\, y_i^2$	$[-100, 100]$	Rotated Sum Square
$f_{11}(x) = \sum_{i=1}^{D} x_i^2 + (\sum_{i=1}^{D} 0.5i\, y_i)^2 + (\sum_{i=1}^{n} 0.5i\, y_i)^4$	$[-10, 10]$	Rotated Zakharov
$f_{12}(x) = \sum_{i=1}^{D-1} [100(y_i^2 - y_{i+1})^2 + (y_i - 1)^2]$	$[-2.048, 2.048]$	Rotated Rosenbrock
$f_{13}(x) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} y_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi y_i)) + 20 + e$	$[-32.768, 32.768]$	Rotated Ackley
$f_{14}(x) = \sum_{i=1}^{D} (y_i^2 - 10 \cos(2\pi y_i + 10))$	$[-5.12, 5.12]$	Rotated Rastrigin
$f_{15}(x) = \sum_{i=1}^{D} (\sum_{k=0}^{20} [0.5^k \cos(2\pi 3^k (y_i + 0.5))]) - D \sum_{k=0}^{20} [0.5^k \cos(2\pi 3^k 0.5)]$	$[-0.5, 0.5]$	Rotated Weierstrass
$f_{16}(x) = \sum_{i=1}^{D} \frac{y_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{y_i}{\sqrt{i}}) + 1$	$[-600, 600]$	Rotated Griewank

our approach and other optimization techniques, hence all tests were conducted 30 times. In our algorithm, contrary to the other optimizers, the halting condition respects the original problem setting, *i.e.*, a hard stop is set if the error is lower than the chosen threshold regardless of the number of iterations.

Convergence Results As one can see from the tabulated results, our algorithm achieves the optimum for all tests outperforming other optimizers except in one specific case. The Rosenbrock function f_5 in dimension 30 obtains a worse result than competing algorithms; with the Rotated Rosenbrock case in dimension 100 (f_{12}), although Table 4 shows numerical values, it should be noted that no optimizer converged to the solution, as detailed in Table 7. Another interesting case is the f_{12} Rotated Rosenbrock in dimension 30, where our approach is substantially on par with other algorithms, with the exception of Artificial Bee Colony (ABC), obtaining convergence with a result of 144 with others laying in the range 20–30 (real optimum 50).

In order to evaluate exploitation of our algorithm, we refer to the first four functions $f_1, \ldots, f_4$ (unimodal), where no local minima are present with one global

Table 2 Results for dimension 10 in terms of mean, standard deviation, and SEM of the attained solution.

Function		PSO	DE	HS	ABC	TLBO	Jaya	C-Jaya	Kinship
f_1	Mean	3.34E-26	8.05E-42	3.39E-01	1.64E-16	8.52E-112	9.65E-27	0.0	1.19E-03
	StdDev	1.01E-25	2.34E-41	1.90E-01	6.95E-17	2.22E-111	1.19E-26	0.0	4.97E-19
	SEM	1.85E-26	4.27E-42	3.46E-02	1.27E-17	4.05E-112	2.17E-27	0.0	1.11E-19
f_2	Mean	2.99E-07	2.16E-01	1.79E+02	1.46E+02	1.29E-47	1.73E-02	0.0	8.06E-08
	StdDev	5.02E-07	3.54E-01	1.42E+02	8.51E+01	5.90E-47	2.51E-02	0.0	6.07E-23
	SEM	9.17E-08	6.47E-02	2.59E+01	1.55E+01	1.08E-47	4.57E-03	0.0	1.36E-23
f_3	Mean	7.32E-26	1.70E-41	1.91E+00	1.98E-16	1.87E-111	2.78E-26	0.0	1.86E-09
	StdDev	2.34E-25	4.09E-41	1.68E+00	1.01E-16	5.20E-111	2.67E-26	0.0	0.0
	SEM	4.26E-26	7.46E-42	3.07E-01	1.85E-17	9.50E-112	4.87E-27	0.0	0.0
f_4	Mean	5.50E-20	1.21E-09	1.54E+00	5.07E-01	3.98E-89	7.21E-13	0.0	2.82E-08
	StdDev	1.82E-19	6.59E-09	1.62E+00	1.09E+00	1.61E-88	1.41E-12	0.0	3.19E-23
	SEM	3.33E-20	1.20E-09	2.96E-01	1.99E-01	2.95E-89	2.57E-13	0.0	7.13E-24
f_5	Mean	4.31E+00	4.89E+00	4.78E+00	1.99E+00	4.64E+00	1.72E-01	8.81E+00	11.3884
	StdDev	1.10E+00	1.41E+00	2.93E+00	1.80E+00	6.59E-01	2.75E-01	2.59E-01	1.06E-14
	SEM	2.01E-01	2.58E-01	5.34E-01	3.29E-01	1.20E-01	5.02E-02	4.73E-02	2.37E-15
f_6	Mean	8.17E-14	4.44E-15	4.46E-01	6.74E-13	4.44E-15	6.22E-14	8.88E-16	4.63E-07
	StdDev	1.71E-13	0.0	2.24E-01	7.57E-13	0.0	6.25E-14	0.0	7.29E-23
	SEM	3.13E-14	0.0	4.09E-02	1.38E-13	0.0	1.14E-14	0.0	1.63E-23
f_7	Mean	1.12E+01	9.95E-02	1.55E-01	3.32E-02	3.94E+00	3.14E+01	0.0	6.21E-07
	StdDev	6.46E+00	3.04E-01	1.08E-01	1.82E-01	2.25E+00	9.31E+00	0.0	5.34E-22
	SEM	1.18E+00	5.54E-02	1.96E-02	3.32E-02	4.10E-01	1.70E+00	0.0	1.19E-22
f_8	Mean	5.00E-02	0.0	4.86E-01	1.54E-15	0.0	5.00E-02	0.0	8.05E-07
	StdDev	2.74E-01	0.0	1.38E-01	6.51E-15	0.0	2.74E-01	0.0	3.89E-22
	SEM	5.00E-02	0.0	2.52E-02	1.19E-15	0.0	5.00E-02	0.0	8.69E-23
f_9	Mean	8.35E-02	1.55E-03	3.99E-01	1.33E-02	7.53E-03	4.56E-01	0.0	2.92E-07
	StdDev	3.46E-02	3.00E-03	1.50E-01	1.21E-02	1.42E-02	1.25E-01	0.0	0.0
	SEM	6.32E-03	5.47E-04	2.73E-02	2.21E-03	2.59E-03	2.28E-02	0.0	0.0
f_{10}	Mean	2.50E-09	4.17E-13	4.59E+01	2.73E-02	6.68E-85	4.01E-18	0.0	4.71E-07
	StdDev	1.37E-08	1.81E-12	4.83E+01	3.98E-02	3.39E-84	1.26E-17	0.0	1.94E-22
	SEM	2.50E-09	3.31E-13	8.82E+00	7.27E-03	6.20E-85	2.29E-18	0.0	6.14E-23
f_{11}	Mean	5.20E-05	1.89E+00	2.18E+00	3.48E+02	2.91E-37	2.65E+02	0.0	3.00E-07
	StdDev	2.63E-04	2.98E+00	2.91E+00	1.16E+02	8.16E-37	9.60E+01	0.0	0.0
	SEM	4.81E-05	5.45E-01	5.32E-01	2.12E+01	1.49E-37	1.75E+01	0.0	0.0
f_{12}	Mean	8.89E+00	8.89E+00	9.26E+00	8.89E+00	8.89E+00	8.88E+00	8.96E+00	23.4709
	StdDev	6.08E-02	1.70E-02	4.36E-01	1.53E-02	1.13E-02	1.45E-02	1.68E-02	17161.2
	SEM	1.11E-02	3.10E-03	7.96E-02	2.79E-03	2.06E-03	2.65E-03	3.07E-03	5426.85
f_{13}	Mean	9.69E-14	4.44E-15	1.84E+00	1.05E-08	4.44E-15	7.19E-14	8.88E-16	2.87E-07
	StdDev	1.96E-13	0.0	7.66E-01	3.40E-08	0.0	5.90E-14	0.0	1.06E-22
	SEM	3.58E-14	0.0	1.40E-01	6.20E-09	0.0	1.08E-14	0.0	3.35E-23
f_{14}	Mean	3.17E+01	3.38E+01	2.98E+01	4.05E+01	2.44E+01	4.05E+01	0.0	4.17E-09
	StdDev	7.73E+00	4.91E+00	4.05E+00	5.97E+00	8.25E+00	4.96E+00	0.0	5.51E-25
	SEM	1.41E+00	8.97E-01	7.39E-01	1.09E+00	1.51E+00	9.06E-01	0.0	1.74E-25
f_{15}	Mean	8.74E-01	2.45E+00	1.70E+00	1.23E+00	0.0	1.08E-04	0.0	3.75E-06
	StdDev	2.58E+00	2.33E+00	7.71E-01	1.10E+00	0.0	1.05E-04	0.0	5.65E-22
	SEM	4.70E-01	4.26E-01	1.41E-01	2.01E-01	0.0	1.91E-05	0.0	1.79E-22
f_{16}	Mean	5.94E-01	4.41E-01	6.71E-01	9.50E-02	1.64E-03	2.22E-02	0.0	1.67E-07
	StdDev	1.56E-01	1.64E-01	8.14E-02	1.07E-01	6.16E-03	7.54E-02	0.0	1.76E-23
	SEM	2.84E-02	3.00E-02	1.49E-02	1.95E-02	1.12E-03	1.38E-02	0.0	5.58E-24

optimal solution. On the other hand, functions $f_6, \ldots, f_9$ (multimodal) highlight the exploration property of our approach handling more local minima compared to other approaches that fail to reach the optimal value (cf. Table 4). Results, thus, demonstrate that the Kinship DE attains good results compared to other optimizers in terms of convergence.

Table 3 Results for dimension 30 in terms of mean, standard deviation, and SEM of the attained solution.

Function		PSO	DE	HS	ABC	TLBO	Jaya	C-Jaya	Kinship
f_1	Mean	5.04E-13	2.89E-28	7.35E+00	1.57E-13	2.48E-179	2.36E-10	0.0	1.64E-04
	StdDev	1.28E-12	1.74E-28	2.61E+00	1.87E-13	0.0	1.25E-10	0.0	4.35E-20
	SEM	2.34E-13	3.18E-29	4.77E-01	3.41E-14	0.0	2.28E-11	0.0	9.73E-21
f_2	Mean	3.43E+01	1.71E+04	2.98E+03	9.92E+03	2.73E-38	2.74E+04	0.0	4.18E-08
	StdDev	1.75E+01	3.03E+03	8.11E+02	2.17E+03	6.58E-38	5.84E+03	0.0	3.04E-23
	SEM	3.20E+00	5.53E+02	1.48E+02	3.96E+02	1.20E-38	1.07E+03	0.0	6.79E-24
f_3	Mean	6.43E-12	3.04E-27	1.01E+02	5.48E-11	9.19E-178	3.32E-09	0.0	2.68E-07
	StdDev	1.60E-11	1.34E-27	3.70E+01	7.99E-11	0.0	2.12E-09	0.0	2.44E-13
	SEM	2.92E-12	2.44E-28	6.75E+00	1.46E-11	0.0	3.87E-10	0.0	5.45E-14
f_4	Mean	3.81E-04	1.95E+02	1.39E+02	6.79E+03	9.00E-96	1.87E+03	0.0	9.02E-08
	StdDev	5.70E-04	6.89E+01	4.31E+01	1.20E+03	3.59E-95	3.55E+02	0.0	5.60E-13
	SEM	1.04E-04	1.26E+01	7.86E+00	2.19E+02	6.55E-96	6.48E+01	0.0	1.25E-13
f_5	Mean	2.54E+01	2.47E+01	5.46E+01	1.50E+01	2.25E+01	1.72E+01	2.88E+01	4528.95
	StdDev	1.57E+00	6.05E-01	2.71E+01	5.44E+00	5.73E-01	1.12E+01	2.78E-01	2.06E+04
	SEM	2.86E-01	1.10E-01	4.94E+00	9.93E-01	1.05E-01	2.05E+00	5.07E-02	4.62E+03
f_6	Mean	1.46E-07	1.59E-14	1.33E+00	5.63E-08	6.10E-15	8.80E-06	8.88E-16	3.56E-07
	StdDev	2.63E-07	1.53E-15	2.81E-01	2.58E-08	1.80E-15	5.86E-06	0.0	2.43E-22
	SEM	4.80E-08	2.79E-16	5.13E-02	4.72E-09	3.29E-16	1.07E-06	0.0	5.43E-23
f_7	Mean	4.15E+01	3.62E+01	2.48E+00	5.92E-03	1.22E+01	2.04E+02	0.0	1.96E-08
	StdDev	1.48E+01	4.16E+00	7.80E-01	3.07E-02	6.97E+00	2.29E+01	0.0	6.07E-24
	SEM	2.70E+00	7.59E-01	1.42E-01	5.61E-03	1.27E+00	4.19E+00	0.0	1.36E-24
f_8	Mean	4.45E-01	0.0	2.47E+00	7.45E-06	0.0	6.43E-01	0.0	4.17E-07
	StdDev	8.03E-01	0.0	5.10E-01	4.62E-06	0.0	9.27E-01	0.0	9.72E-23
	SEM	1.47E-01	0.0	9.31E-02	8.44E-07	0.0	1.69E-01	0.0	2.17E-23
f_9	Mean	1.02E-02	0.0	1.07E+00	2.92E-03	8.25E-10	8.19E-02	0.0	2.11E-07
	StdDev	9.77E-03	0.0	1.99E-02	6.45E-03	4.52E-09	1.24E-01	0.0	1.70E-22
	SEM	1.78E-03	0.0	3.62E-03	1.18E-03	8.25E-10	2.27E-02	0.0	3.80E-23
f_{10}	Mean	1.05E+01	9.01E-03	8.09E+02	4.77E+04	2.98E-153	3.40E-02	0.0	1.86E-07
	StdDev	2.66E+01	1.22E-02	3.88E+02	2.23E+04	6.12E-153	2.91E-02	0.0	2.78E-12
	SEM	4.86E+00	2.23E-03	7.08E+01	4.07E+03	1.12E-153	5.31E-03	0.0	5.08E-13
f_{11}	Mean	2.21E+02	5.15E+03	4.73E+02	8.25E+03	3.89E-27	9.43E+03	0.0	1.99E-07
	StdDev	4.14E+02	7.05E+02	1.39E+02	8.99E+02	1.23E-26	1.12E+03	0.0	2.65E-22
	SEM	7.57E+01	1.29E+02	2.53E+01	1.64E+02	2.25E-27	2.05E+02	0.0	4.85E-23
f_{12}	Mean	2.91E+01	2.89E+01	3.02E+01	1.44E+02	2.89E+01	2.90E+01	2.90E+01	21.1314
	StdDev	1.36E-01	2.88E-02	4.52E-01	8.71E+01	1.58E-02	3.42E-02	1.60E-02	4.42E-14
	SEM	2.48E-02	5.25E-03	8.26E-02	1.59E+01	2.89E-03	6.25E-03	2.91E-03	8.07E-15
f_{13}	Mean	5.23E-06	2.45E-13	2.95E+00	1.85E+00	5.39E-15	1.51E-05	8.88E-16	7.74E-08
	StdDev	8.89E-06	2.18E-13	3.48E-01	6.22E-01	1.60E-15	6.05E-06	0.0	1.33E-22
	SEM	1.62E-06	3.97E-14	6.35E-02	1.14E-01	2.92E-16	1.11E-06	0.0	2.42E-23
f_{14}	Mean	1.69E+02	2.10E+02	1.87E+02	2.69E+02	1.27E+02	2.60E+02	0.0	2.84E-08
	StdDev	1.19E+01	1.07E+01	1.07E+01	1.51E+01	5.07E+01	1.49E+01	0.0	5.71E-23
	SEM	2.18E+00	1.95E+00	1.95E+00	2.75E+00	9.26E+00	2.72E+00	0.0	1.04E-23
f_{15}	Mean	2.97E+00	3.40E+01	7.02E+00	3.73E+01	0.0	3.62E+01	0.0	3.48E-06
	StdDev	3.49E+00	2.34E+00	8.93E-01	1.21E+00	0.0	1.26E+00	0.0	4.72E-21
	SEM	6.36E-01	4.28E-01	1.63E-01	2.21E-01	0.0	2.30E-01	0.0	8.62E-22
f_{16}	Mean	4.56E-01	1.18E-03	1.06E+00	1.89E-01	0.0	3.76E-01	0.0	3.74E-08
	StdDev	3.63E-01	5.32E-04	2.99E-02	8.87E-02	0.0	7.05E-02	0.0	7.62E-23
	SEM	6.63E-02	9.72E-05	5.45E-03	1.62E-02	0.0	1.29E-02	0.0	1.39E-23

Convergence Speed Optimizers are employed in several real-world problems that require evaluating complex time-demanding functions, thus, the *speed* of an optimizing algorithm shall be considered not in terms of wall-clock time—varying depending of several factors, from computer architectures to compilers used to run the source code—but in terms of the number of *function evaluations*, or FEs.

Tables 5, 6, and 7 show the mean number of function evaluations over the 30 runs, as highlighted in the beginning of this section; along with the mean FEs,

Table 4 Results for dimension 100 in terms of mean, standard deviation, and SEM of the attained solution.

Function		PSO	DE	HS	ABC	TLBO	Jaya	C-Jaya	Kinship
f_1	Mean	8.98E+00	1.43E-03	1.31E+04	2.99E-03	1.14E-162	2.38E+01	0.0	1.45E-04
	StdDev	3.62E+00	2.53E-04	1.11E+03	2.76E-03	2.22E-162	8.75E+00	0.0	2.55E-19
	SEM	6.61E-01	4.62E-05	2.03E+02	5.04E-04	4.06E-163	1.60E+00	0.0	5.70E-20
f_2	Mean	2.17E+04	3.55E+05	3.15E+05	1.68E+05	1.85E-17	4.00E+05	0.0	8.93E-07
	StdDev	5.94E+03	2.42E+04	3.40E+04	1.59E+04	3.71E-17	4.36E+04	0.0	1.12E-21
	SEM	1.08E+03	4.41E+03	6.21E+03	2.90E+03	6.77E-18	7.96E+03	0.0	2.50E-22
f_3	Mean	3.68E+02	5.52E-02	5.22E+05	1.14E-01	5.69E-161	1.41E+03	0.0	9.86E-08
	StdDev	1.63E+02	1.33E-02	5.20E+04	8.46E-02	8.77E-161	6.96E+02	0.0	4.25E-23
	SEM	2.98E+01	2.43E-03	9.50E+03	1.54E-02	1.60E-161	1.27E+02	0.0	9.51E-24
f_4	Mean	2.47E+03	1.20E+05	5.94E+04	1.38E+05	1.28E-25	1.32E+05	0.0	4.41E-07
	StdDev	4.22E+02	6.64E+03	5.85E+03	7.83E+03	5.17E-25	1.63E+04	0.0	3.40E-22
	SEM	7.70E+01	1.21E+03	1.07E+03	1.43E+03	9.43E-26	2.97E+03	0.0	7.60E-23
f_5	Mean	9.78E+01	9.64E+01	1.34E+03	2.48E+02	9.50E+01	1.72E+02	9.88E+01	9.98E+01
	StdDev	1.05E+00	5.08E-01	9.55E+01	4.67E+01	1.06E+00	4.65E+01	2.10E-01	3.91E-14
	SEM	1.92E-01	9.27E-02	1.74E+01	8.52E+00	1.93E-01	8.49E+00	3.84E-02	8.75E-15
f_6	Mean	1.85E+00	5.56E-03	1.22E+01	2.01E+00	7.99E-15	5.38E+00	8.88E-16	1.78E-06
	StdDev	3.06E-01	6.54E-04	3.40E-01	2.40E-01	0.0	1.01E+00	0.0	1.94E-22
	SEM	5.60E-02	1.19E-04	6.20E-02	4.38E-02	0.0	1.85E-01	0.0	4.35E-23
f_7	Mean	1.47E+02	6.06E+02	1.99E+02	5.15E+01	3.17E+00	6.94E+02	0.0	2.03E-08
	StdDev	3.08E+01	2.03E+01	1.71E+01	8.72E+00	1.22E+01	1.50E+02	0.0	0.0
	SEM	5.62E+00	3.71E+00	3.11E+00	1.59E+00	2.24E+00	2.74E+01	0.0	0.0
f_8	Mean	8.64E+00	3.18E-01	3.68E+01	2.28E+00	0.0	3.51E+01	0.0	2.92E-06
	StdDev	3.14E+00	2.76E-02	2.17E+00	6.57E-01	0.0	4.83E+00	0.0	0.0
	SEM	5.73E-01	5.03E-03	3.95E-01	1.20E-01	0.0	8.82E-01	0.0	0.0
f_9	Mean	1.10E+00	9.46E-04	1.23E+02	6.62E-02	0.0	1.26E+00	0.0	3.81E-07
	StdDev	3.98E-02	3.47E-04	1.12E+01	5.51E-02	0.0	1.31E-01	0.0	7.29E-23
	SEM	7.26E-03	6.34E-05	2.05E+00	1.01E-02	0.0	2.40E-02	0.0	1.63E-23
f_{10}	Mean	5.74E+04	8.77E+05	7.81E+05	9.28E+06	7.21E-151	7.01E+04	0.0	2.64E-07
	StdDev	1.77E+04	3.09E+05	7.14E+04	6.40E+05	1.12E-150	3.25E+04	0.0	5.91E-22
	SEM	3.22E+03	5.63E+04	1.30E+04	1.17E+05	2.05E-151	5.94E+03	0.0	5.91E-23
f_{11}	Mean	2.60E+04	1.32E+05	8.00E+04	1.32E+05	7.81E+02	1.63E+05	0.0	4.75E-09
	StdDev	9.39E+03	8.24E+03	6.10E+03	7.51E+03	4.67E+02	1.42E+04	0.0	4.99E-24
	SEM	1.71E+03	1.50E+03	1.11E+03	1.37E+03	8.53E+01	2.60E+03	0.0	4.99E-25
f_{12}	Mean	1.15E+02	3.34E+02	8.46E+02	2.47E+04	9.88E+01	1.40E+02	9.89E+01	419.845
	StdDev	3.60E+00	1.07E+02	9.12E+01	2.71E+03	2.04E-02	4.24E+01	2.16E-02	1.49E-12
	SEM	6.58E-01	1.95E+01	1.67E+01	4.96E+02	3.73E-03	7.75E+00	3.94E-03	1.49E-13
f_{13}	Mean	3.91E+00	2.12E+00	1.24E+01	2.06E+01	7.76E-15	3.06E+00	8.88E-16	5.84E-07
	StdDev	2.33E-01	1.89E-01	4.46E-01	1.62E-01	9.01E-16	3.23E-01	0.0	1.36E-21
	SEM	4.26E-02	3.44E-02	8.14E-02	2.96E-02	1.65E-16	5.89E-02	0.0	1.36E-22
f_{14}	Mean	8.17E+02	1.14E+03	9.64E+02	1.44E+03	2.95E+00	1.07E+03	0.0	5.17E-08
	StdDev	3.02E+01	3.15E+01	2.67E+01	3.97E+01	6.13E+00	2.83E+01	0.0	8.38E-23
	SEM	5.51E+00	5.74E+00	4.87E+00	7.25E+00	1.12E+00	5.17E+00	0.0	8.38E-24
f_{15}	Mean	4.94E+01	1.51E+02	8.81E+01	1.53E+02	0.0	1.47E+02	0.0	1.76E-06
	StdDev	1.06E+01	1.96E+00	2.92E+00	2.00E+00	0.0	2.57E+00	0.0	1.49E-21
	SEM	1.94E+00	3.58E-01	5.33E-01	3.65E-01	0.0	4.70E-01	0.0	1.49E-22
f_{16}	Mean	1.09E+00	8.66E-01	1.22E+02	8.22E-01	0.0	1.24E+00	0.0	7.80E-08
	StdDev	3.41E-02	3.39E-02	1.06E+01	1.45E-01	0.0	1.21E-01	0.0	1.32E-21
	SEM	6.23E-03	6.18E-03	1.94E+00	2.65E-02	0.0	2.21E-02	0.0	1.32E-22

we reproduce the *success rate* (SR) for each function, *i.e.*, the percentage of times the algorithm was able to produce the optimal solution over the total number of executions.

Our Kinship DE improved sensibly the number of required FEs compared to other optimizers, as one can see from Figs. 4 and 5, where we present the number of function evaluations in a logarithmic scale, comparing our approach the TLBO and C-Jaya, the third and second best algorithms in terms of speed. We point out

Table 5 Timing for dimension 10 in terms of mean number of Function Evaluations (FEs) and Success Rate (SR). Dot symbol indicates non-convergence.

Function		PSO	DE	HS	ABC	TLBO	Jaya	C·Jaya	Kinship
f_1	MeanFEs	2.69E+03	2.70E+03	6.16E+03	2.82E+03	1.12E+03	4.05E+03	2.01E+02	6.92E+00
	SR(%)	100	100	3.3	100	1000	100	100	100
f_2	MeanFEs	1.52E+04	·	·	·	3.46E+03	·	3.09E+02	9.22E+00
	SR(%)	100	0.0	0.0	0.0	100	0.0	100	100
f_3	MeanFEs	5.27E+03	4.36E+03	·	7.49E+03	1.75E+03	6.47E+03	2.86E+02	9.65E+00
	SR(%)	100	100	0.0	100	100	100	100	100
f_4	MeanFEs	5.83E+03	1.33E+04	1.96E+04	·	2.38E+03	1.24E+04	2.94E+02	8.02E+00
	SR(%)	100	100	3.3	0.0	100	100	100	100
f_5	MeanFEs	1.12E+02	8.91E+02	1.48E+03	9.25E+02	2.47E+02	1.04E+03	7.53E+01	3.46
	SR(%)	100	100	100	100	100	100	100	100
f_6	MeanFEs	7.80E+03	5.95E+03	·	1.04E+04	2.36E+03	9.19E+03	3.96E+02	1.12E+01
	SR(%)	100	100	0.0	100	100	100	100	100
f_7	MeanFEs	·	7.20E+03	1.89E+04	1.01E+04	1.36E+04	·	2.67E+02	8.73E+00
	SR(%)	0.0	90	3.3	93.3	10	0.0	100	100
f_8	MeanFEs	1.13E+04	7.38E+03	·	1.26E+04	3.81E+03	1.52E+04	4.57E+02	1.36E+01
	SR(%)	96.7	100	0.0	100	100	96.7	100	100
f_9	MeanFEs	·	1.08E+04	·	1.21E+04	9.31E+03	·	3.24E+02	8.75E+00
	SR(%)	0.0	76.7	0.0	30	66.7	0.0	100	100
f_{10}	MeanFEs	8.12E+03	1.09E+04	·	·	2.21E+03	8.60E+03	3.01E+02	1.10E+01
	SR(%)	100	100	0.0	0.0	100	100	100	100
f_{11}	MeanFEs	1.30E+04	·	·	·	5.83E+03	·	2.86E+02	8.74E+00
	SR(%)	90	0.0	0.0	0.0	100	0.0	100	100
f_{12}	MeanFEs	1.39E+02	1.15E+03	5.03E+02	1.81E+03	2.76E+02	9.27E+02	7.33E+01	3.78
	SR(%)	100	100	100	100	100	100	100	100
f_{13}	MeanFEs	8.28E+03	7.26E+03	·	1.53E+04	2.43E+03	9.21E+03	3.50E+02	1.19E+01
	SR(%)	100	100	0.0	100	100	100	100	100
f_{14}	MeanFEs	8.12E+02	3.44E+03	1.53E+03	8.23E+03	8.79E+02	5.53E+03	9.93E+01	9.41E+00
	SR(%)	96.7	100	100	96.7	100	96.7	100	100
f_{15}	MeanFEs	1.89E+04	·	·	·	5.01E+03	·	5.04E+02	1.49E+01
	SR(%)	3.3	0.0	0.0	0.0	100	0.0	100	100
f_{16}	MeanFEs	9.06E+03	·	·	·	5.53E+03	1.48E+04	2.76E+02	9.85E+00
	SR(%)	3.3	0.0	0.0	0.0	76.7	73.3	100	100

that our algorithm fails to attain the optimal value for f_5 and f_{12} (Rosenbrock, and Rotated Rosenbrock, respectively) in dimension 100, functions proved to be non-optimized by any of the tested algorithms.

Another interesting property is the *convergence behavior* of the Kinship DE, as pictured in Fig. 6 for the 10-dimensional Ackley (f_6) and Rastrigin (f_7). Here we represented all the population individuals in terms of absolute values of their fitness with respect to their generation, *i.e.*, their FE number. As easily seen both in the tables and in Fig. 6, our approach converges rapidly within the first generations, obtaining an acceptable solution within the first ten generations.

4.2 Statistical Analysis

The statistical analysis of the results detailed in the previous sections show that our Kinship DE, compared with other approaches, has consistent better performances in terms of the number of function evaluations. In details, the average rankings of all the algorithms are pictured in Fig. 7 grouped by problem dimension, and in order to assess its validity, we performed the Cochran's Q test, by defining as

Table 6 Timing for dimension 30 in terms of mean number of Function Evaluations (FEs) and Success Rate (SR). Dot symbol indicates non-convergence.

Function		PSO	DE	HS	ABC	TLBO	Jaya	C·Jaya	Kinship
f_1	MeanFEs	2.43E+04	1.69E+04	·	1.60E+04	3.05E+03	3.77E+04	4.92E+02	7.19E+00
	SR(%)	100	100	0.0	100	100	100	100	100
f_2	MeanFEs	·	·	·	·	1.79E+04	·	8.39E+02	9.79E+00
	SR(%)	0.0	0.0	0.0	0.0	100	0.0	100	100
f_3	MeanFEs	4.49E+04	2.71E+04	·	4.53E+04	4.83E+03	6.04E+04	7.01E+02	1.02E+01
	SR(%)	100	100	0.0	100	100	100	100	100
f_4	MeanFEs	·	·	·	·	1.45E+04	·	8.39E+02	8.51E+00
	SR(%)	0.0	0.0	0.0	0.0	100	0.0	100	100
f_5	MeanFEs	1.48E+03	1.22E+04	3.42E+04	1.66E+04	1.01E+03	1.73E+04	2.07E+02	4.08
	SR(%)	100	100	50	100	100	96.7	100	100
f_6	MeanFEs	5.93E+04	3.37E+04	·	6.23E+04	6.05E+03	7.77E+04	9.07E+02	1.16E+01
	SR(%)	100	100	0.0	100	100	76.7	100	100
f_7	MeanFEs	·	·	·	6.40E+04	3.18E+04	·	6.81E+02	8.88E+00
	SR(%)	0.0	0.0	0.0	93.3	10	0.0	100	100
f_8	MeanFEs	7.39E+04	4.29E+04	·	7.75E+04	9.39E+03	·	1.16E+03	1.40E+01
	SR(%)	50	100	0.0	70	100	0.0	100	100
f_9	MeanFEs	4.12E+04	2.69E+04	·	5.04E+04	5.31E+03	5.75E+04	6.83E+02	9.15E+00
	SR(%)	36.7	100	0.0	80	100	30	100	100
f_{10}	MeanFEs	·	·	·	·	5.59E+03	·	7.44E+02	9.89E+00
	SR(%)	0.0	0.0	0.0	0.0	100	0.0	100	100
f_{11}	MeanFEs	·	·	·	·	3.85E+04	·	8.13E+02	8.39E+00
	SR(%)	0.0	0.0	0.0	0.0	100	0.0	100	100
f_{12}	MeanFEs	1.90E+03	1.66E+04	1.18E+04	6.59E+04	1.12E+03	1.99E+04	2.15E+02	3.99
	SR(%)	100	100	100	20	100	100	100	100
f_{13}	MeanFEs	7.00E+04	4.21E+04	·	·	6.03E+03	7.93E+04	9.00E+02	1.12E+01
	SR(%)	83.3	100	0.0	0.0	100	6.67E+00	100	100
f_{14}	MeanFEs	·	·	·	·	6.13E+04	·	2.92E+02	8.64E+00
	SR(%)	0.0	0.0	0.0	0.0	13.3	0.0	100	100
f_{15}	MeanFEs	·	·	·	·	1.07E+04	·	1.24E+03	1.29E+01
	SR(%)	0.0	0.0	0.0	0.0	100	0.0	100	100
f_{16}	MeanFEs	·	·	·	·	5.94E+03	·	7.20E+02	9.40E+00
	SR(%)	0.0	0.0	0.0	0.0	100	0.0	100	100

converged a series of 30 runs on a single dimension optimization problem yielding a success rate of at least 95%, highlighting that the ranking is statistical significant.

Table 7 Timing for dimension 100 in terms of mean number of Function Evaluations (FEs) and Success Rate (SR). Dot symbol indicates non-convergence.

Function		PSO	DE	HS	ABC	TLBO	Jaya	C·Jaya	Kinship
f_1	MeanFEs	·	7.18E+04	·	6.39E+04	3.60E+03	·	5.92E+02	8.40E+00
	SR(%)	0.0	100	0.0	100	100	0.0	100	100
f_2	MeanFEs	·	·	·	·	3.57E+04	·	1.05E+03	9.90E+00
	SR(%)	0.0	0.0	0.0	0.0	100	0.0	100	100
f_3	MeanFEs	·	·	·	·	5.82E+03	·	8.84E+02	1.15E+01
	SR(%)	0.0	0.0	0.0	0.0	100	0.0	100	100
f_4	MeanFEs	·	·	·	·	5.03E+04	·	1.41E+03	1.04E+01
	SR(%)	0.0	0.0	0.0	0.0	100	0.0	100	100
f_5	MeanFEs	·	·	·	·	·	·	·	·
	SR(%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
f_6	MeanFEs	·	·	·	·	6.59E+03	·	9.81E+02	1.09E+01
	SR(%)	0.0	0.0	0.0	0.0	100	0.0	100	100
f_7	MeanFEs	·	·	·	·	7.79E+03	·	7.75E+02	9.40E+00
	SR(%)	0.0	0.0	0.0	0.0	93.3	0.0	100	100
f_8	MeanFEs	·	·	·	·	1.01E+04	·	1.34E+03	1.30E+01
	SR(%)	0.0	0.0	0.0	0.0	100	0.0	100	100
f_9	MeanFEs	·	·	·	·	4.99E+03	·	7.84E+02	8.80E+00
	SR(%)	0.0	0.0	0.0	0.0	100	0.0	100	100
f_{10}	MeanFEs	·	·	·	·	6.19E+03	·	9.08E+02	8.70E+00
	SR(%)	0.0	0.0	0.0	0.0	100	0.0	100	100
f_{11}	MeanFEs	·	·	·	·	·	·	1.11E+03	7.90E+00
	SR(%)	0.0	0.0	0.0	0.0	0.0	0.0	100	100
f_{12}	MeanFEs	·	·	·	·	·	·	·	·
	SR(%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
f_{13}	MeanFEs	·	·	·	·	6.65E+03	·	9.79E+02	1.09E+01
	SR(%)	0.0	0.0	0.0	0.0	100	0.0	100	100
f_{14}	MeanFEs	·	·	·	·	3.95E+04	·	3.37E+02	8.30E+00
	SR(%)	0.0	0.0	0.0	0.0	100	0.0	100	100
f_{15}	MeanFEs	·	·	·	·	1.06E+04	·	1.42E+03	1.13E+01
	SR(%)	0.0	0.0	0.0	0.0	100	0.0	100	100
f_{16}	MeanFEs	·	·	·	·	6.57E+03	·	7.77E+02	8.90E+00
	SR(%)	0.0	0.0	0.0	0.0	100	0.0	100	100

Fig. 4 MeanFEs values in log·scale for the most performing algorithms of Table 6, timing of dim 30, 100 runs.

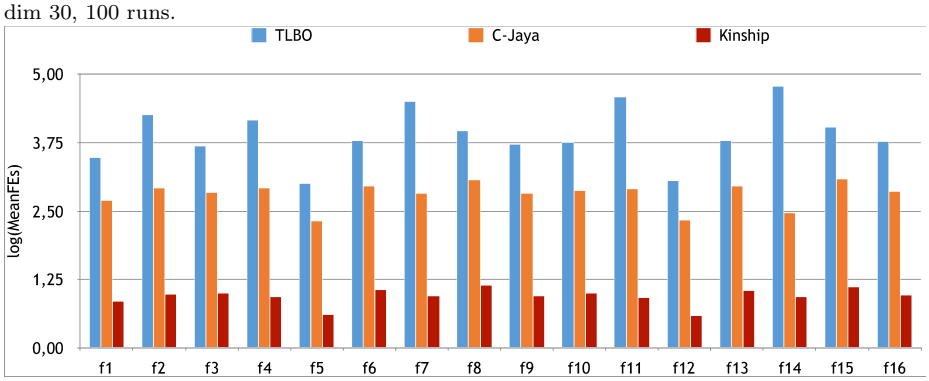


Fig. 5 MeanFEs values in log-scale for the most performing algorithms of Table 7, timing of dim 100, 100 runs (omitted values mean failure in convergence).

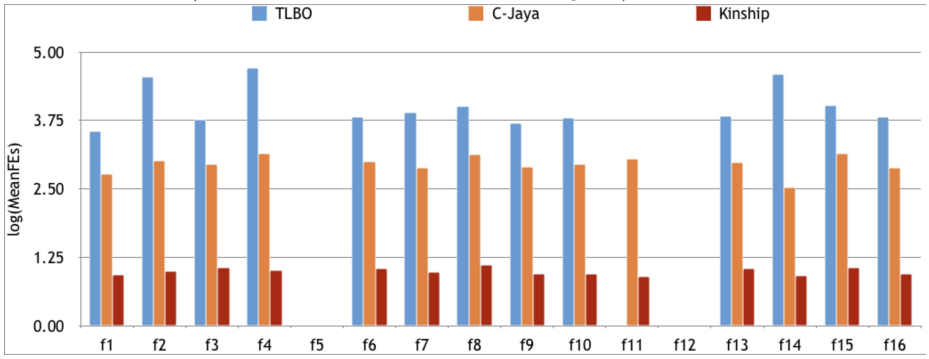


Fig. 6 Convergence behaviour of dim 10 for Ackley and Rastrigin functions (see Table 5).

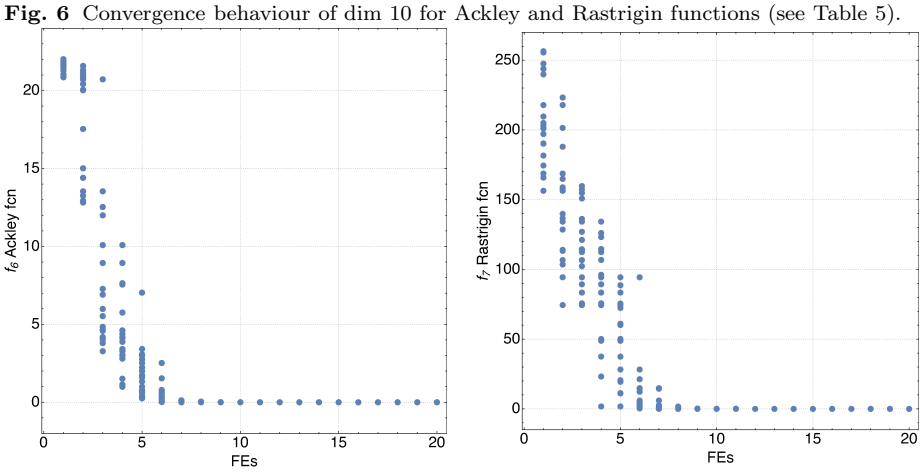


Fig. 7 Average ranking of all algorithms by problem dimension.

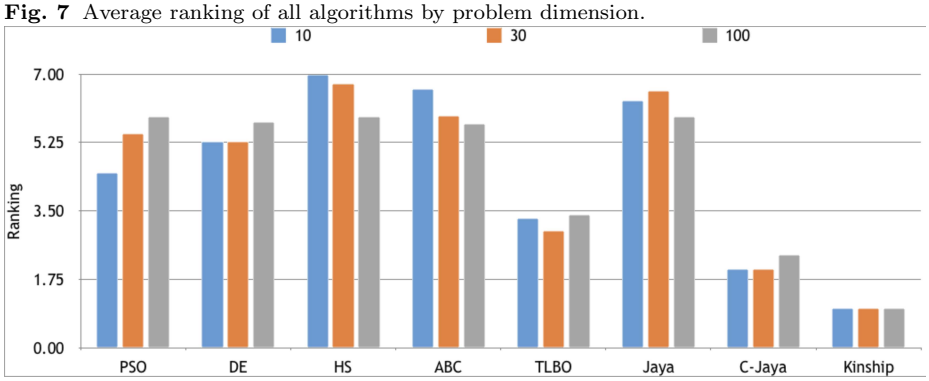


Table 8 Comparison of results for different DE variants in dimension 30. In particular, Std. DE indicates the standard DE/rand/1/bin algorithm.

Function		JADE	SHADE	jDE	SaDE	Std. DE	Kinship
f_1	Mean	1.8E-60	1.0E-70	2.5E-28	4.5E-20	9.8E-14	1.64E-04
	StdDev	8.4E-60	4.4E-70	3.5E-28	6.9E-20	8.4E-14	4.35E-20
f_2	Mean	5.7E-61	5.4E-64	5.2E-14	9.0E-37	6.6E-11	4.18E-08
	StdDev	2.7E-60	3.3E-63	1.1E-13	9.0E-37	8.8E-11	3.04E-23
f_5	Mean	8.0E-02	8.0E-02	1.3E+01	2.1E+01	2.1E+00	4528.95
	StdDev	5.6E-01	5.6E-01	1.4E+01	7.8E+00	1.5E+00	2.06E+04
f_7	Mean	1.0E-04	1.6E-02	1.5E-04	1.2E-03	1.8E+02	1.96E-08
	StdDev	6.0E-05	7.4E-03	2.0E-04	6.5E-04	1.3E+01	6.07E-24
f_{13}	Mean	8.2E-10	2.5E-10	3.5E-04	2.7E-03	1.1E-01	7.74E-08
	StdDev	6.9E-10	9.4E-11	1.0E-04	5.1E-04	3.9E-02	1.33E-22
f_{16}	Mean	9.9E-08	1.5E-14	1.9E-05	7.8E-04	2.0E-01	3.74E-08
	StdDev	6.0E-07	9.3E-14	5.8E-05	1.2E-03	1.1E-01	7.62E-23

Comparison with other DE algorithms In order to assess the contribution of the proposed Kinship DE scheme with respect to recent variants of Differential Evolution family of algorithms, we performed a Wilcoxon signed-rank test, with the function statistics for dimension 30 reported in Table 8.

In details, the DE algorithms we chose are JADE [51], SHADE [43], jDE [5], SaDE [32], and the standard Differential Evolution algorithm (DE/rand/1/bin), where all but the last are adaptive DE optimizers. In particular, the JADE algorithm employs the mutation strategy dubbed DE/current-to-p-best, that improves DE/current-to-best by selecting a given percentage (p) of the best individuals; JADE also offers adaptation of control parameters, and optionally, driving their modification on the basis of historical data (archive). SHADE is derived from JADE, and works with historical memory of all successful tested control parameters, sampling the parameter space in a neighborhood of one of the old stored pairs; additionally, the percentage of the strategy DE/current-to-p-best is associated to each individual, and not constant. Based on the DE/rand/1/bin strategy, jDE is an adaptive DE algorithm with fixed population and adapting parameters: the adaptation phase employs uniform distributions to compute, at each generation, the new control parameters. As a multi-mutation algorithm, SaDE implements two different strategies, in particular the DE/rand/1 and DE/current-to-best/1 strategies; the probability of creating new individuals is based on previous history, and, in order to achieve a better convergence speed, applies a quasi-Newton search to part of the population.

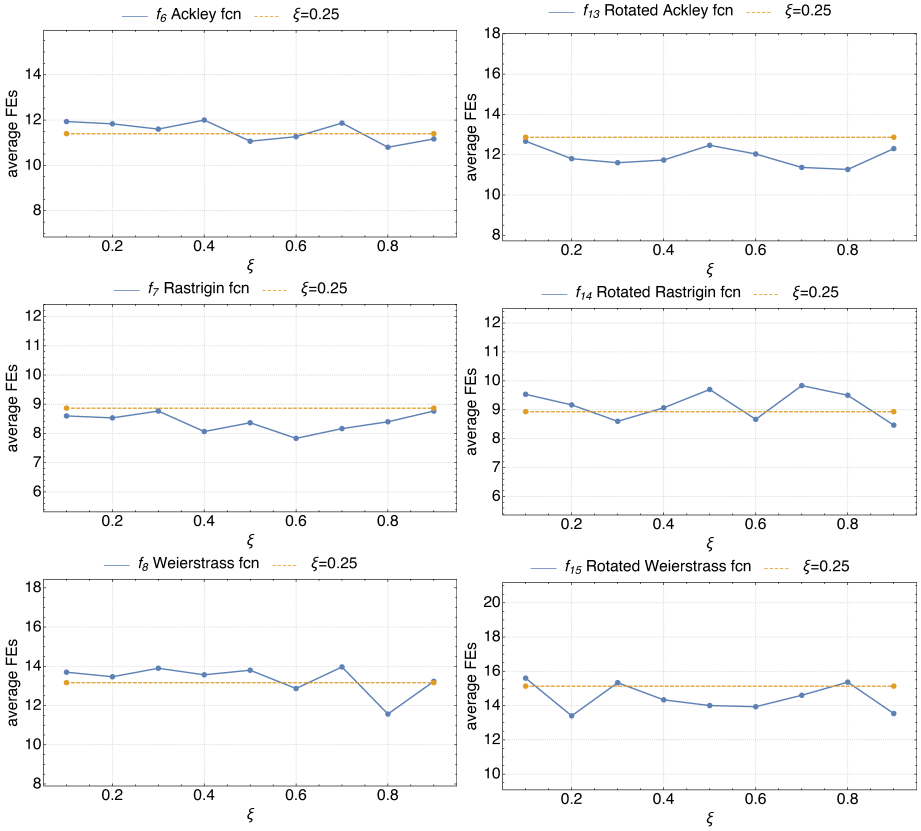
The results of the Wilcoxon signed-rank test showed that Kinship DE performs comparably or better than other DE variants. One single exception is the Rosenbrock f_5 test-function, where Kinship performs worse than the others. Note that we adopted in our computation a fixed threshold of acceptance rates equal to 10^{-5}, except for f_1 (10^{-2}), and f_5 (50). This is evident, for instance, in the sphere f_1 test-function, where Kinship algorithm stops immediately after reaching the given threshold.

4.3 Sensitivity Analysis

We performed three different sensitivity analyses in order to inspect the results variability, while altering key aspects of our kinship algorithm.

A first analysis consists of six functions—Ackley, Rastrigin, Weierstrass, and their respective rotated versions— tested with different values of coefficient $\xi < 1$: the numerical value of ξ, controlling the threshold parameter $\tau = t/(\xi\, t_{\max})$ in equation (5), gauges the preference of the kinship optimizer towards favoring global or local searches. Fig. 8 shows that for each optimized function the performances in terms of function evaluations (FEs) are weakly sensitive to variations of ξ, with the maximal distance resulting in 12% from default value $\xi = 1/4$.

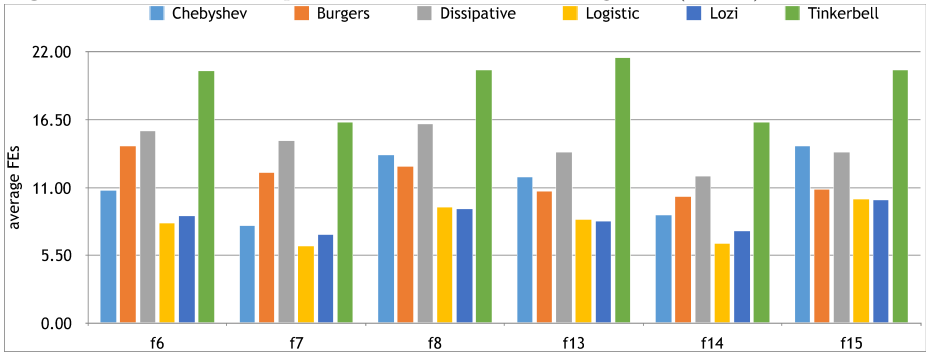
Fig. 8 Average number of FEs plotted against ξ threshold parameter (dim 10); the dotted line represents the number of FEs for $\xi = 1/4$ used in Section 3.



A second sensitivity analysis is devoted to test the influence of the use of the chaotic maps within our algorithm. In particular, we adopted six different chaotic maps: Chebyshev, Burgers, Dissipative, Logistic, Lozi, and Tinkerbell. For more information about chaotic maps, their applications and theory, we refer the reader to [39] and [40]. Of note, two chaotic maps, namely the Chebyshev [8] and

the Logistic one [30] are employed in secure communications, in particular, for authentication and encryption. As Fig. 9 shows, the Chebyshev map, chosen as the default, obtains average performances with the respect to the other maps: as our focus is on the proposed Kinship mutation strategy, such a result validates that the performances of the whole algorithm can be minimally attributed to the choice of a specific chaotic map, rather than the Kinship mutation strategy in itself.

Fig. 9 Different chaotic maps and their effects on the average FEs (dim 10).



Finally, we analyze the effects of changes in the kinship relationship function computing the *kinship coefficient* κ. Additionally to the default function detailed in Section 3, we implemented four alternatives, saturate, single, floor, and non-relatives, respectively:

$$\kappa = \begin{cases} 6\,, & n^p_{i,j} = 2\,, \\ n^p_{i,j}\,, & \text{otherwise}\,, \end{cases} \tag{7}$$

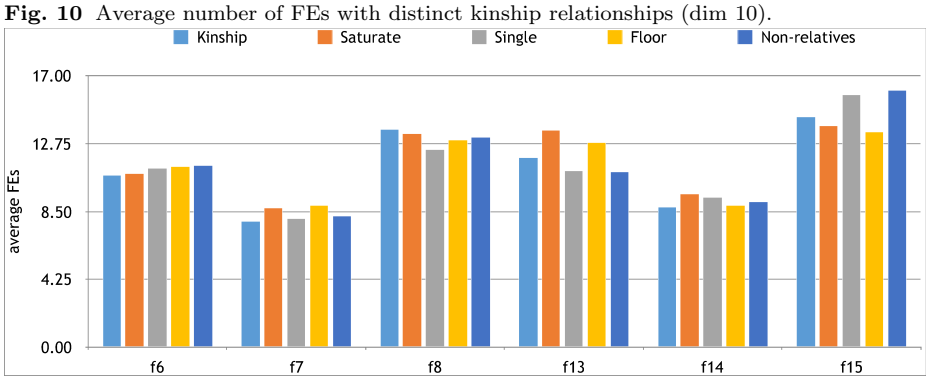
$$\kappa = \begin{cases} 6\,, & n^p_{i,j} = 2\,, \\ 5\,, & n^p_{i,j} = 1\,, \\ n^p_{i,j}\,, & \text{otherwise}\,, \end{cases} \tag{8}$$

$$\kappa = \begin{cases} 6\,, & n^p_{i,j} = 2\,, \\ 0\,, & \text{otherwise}\,, \end{cases} \tag{9}$$

$$\kappa = \begin{cases} 4\,, & n^p_{i,j} \geq 0\,, \\ 3\,, & \text{otherwise}\,. \end{cases} \tag{10}$$

The first relationship (7) saturates to the maximal value when two individuals share both parents, while leaving the weight of the single-parent contribution unvaried; the second one (8) favors single parents, *i.e.*, attributing a similar importance to individuals with one or two progenitors. As shown in equation (9), the third relationship accounts for only individuals with both identical parents, while the last equation (10) assigns almost identical weights to individuals with no relationship, and to those sharing one or two progenitors. The default kinship function, counting the number of ancestors up to the second grade, has the average performances between the other relationships, underlying how the overall algorithm is not highly dependent on the choice of a particular kinship relation.

Fig. 10 Average number of FEs with distinct kinship relationships (dim 10).



5 Final Remarks

In this manuscript we presented a novel approach to standard Differential Evolution (DE) optimization algorithms that exploits the concept of *kinship*. By ensuring that the genetic memory is propagated through generations, *i.e.*, each individual stores its parents and grandparents, our algorithm improved the performances over various techniques by attaining optimal solutions with fewer function evaluations.

We took advantage of chaos theory, by implementing a chaotic map—recently proposed for swarm intelligence—that facilitate the search for the best population in the mutation phase of our Kinship DE optimization algorithm.

Several numerical tests showed how our approach is performing in comparison with other optimizers, in terms of both function evaluations and success rates. In particular, we performed a statistical analysis by means of Cochran's Q test to ensure that the ranking is statistical significant, and a Wilcoxon signed-rank test revealed comparably or better performances of our approach with respect to other adaptive DE algorithms.

Finally, we emphasized through a wide sensitivity analysis how the capabilities of the whole algorithm can be attributed to the Kinship mutation strategy in itself, rather than the use of a specific Kinship relationship or a specific chaotic map.

6 Conflict of Interests

The authors declare that they have no conflict of interest.

References

1. Hussein A Abbass. An evolutionary artificial neural networks approach for breast cancer diagnosis. *Artificial intelligence in Medicine*, 25(3):265–281, 2002.
2. Bilal Alatas. Chaotic bee colony algorithms for global numerical optimization. *Expert Systems with Applications*, 37(8):5682–5687, 2010.
3. Bilal Alatas. Chaotic harmony search algorithms. *Applied Mathematics and Computation*, 216(9):2687–2699, 2010.
4. Christian Blum, Jakob Puchinger, Günther R Raidl, and Andrea Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151, 2011.
5. Janez Brest, Saso Greiner, Borko Boskovic, Marjan Mernik, and Viljem Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, 10:646–657, 2006.
6. Janez Brest and Mirjam Sepesy Maučec. Self-adaptive differential evolution algorithm using population size reduction and three strategies. *Soft Computing*, 15(11):2157–2174, 2011.
7. Janez Brest, Ales Zamuda, Borko Boskovic, Mirjam Sepesy Maucec, and Viljem Zumer. Dynamic optimization using self-adaptive differential evolution. *2009 IEEE Congress on Evolutionary Computation*, pages 415–422, 2009.
8. Santanu Chatterjee, Sandip Roy, Ashok Kumar Das, Samiran Chattopadhyay, Neeraj Kumar, and Athanasios V. Vasilakos. Secure biometric-based authentication scheme using chebyshev chaotic map for multi-server environment. *IEEE Transactions on Dependable and Secure Computing*, 15:824–839, 2018.
9. Nikunj Chauhan, Vadlamani Ravi, and D Karthik Chandra. Differential evolution trained wavelet neural networks: Application to bankruptcy prediction in banks. *Expert Systems with Applications*, 36(4):7659–7665, 2009.
10. Debao Chen, Feng Zou, Zheng Li, Jiangtao Wang, and Suwen Li. An improved teaching-learning-based optimization algorithm for solving global optimization problem. *Information Sciences*, 297:171–190, 2015.
11. Swagatam Das, Sankha Subhra Mullick, and P.N. Suganthan. Recent advances in differential evolution – an updated survey. *Swarm and Evolutionary Computation*, 27:1–30, 2016.
12. Swagatam Das and Ponnuthurai Nagaratnam Suganthan. Differential evolution: a survey of the state-of-the-art. *IEEE transactions on evolutionary computation*, 15(1):4–31, 2011.
13. Anouar Farah and Akram Belazi. A novel chaotic jaya algorithm for unconstrained numerical optimization. *Nonlinear Dynamics*, 93(3):1451–1480, 2018.
14. Giovanni Formica, Franco Milicchio, and Walter Lacarbonara. Hysteretic damping optimization in carbon nanotube nanocomposites. *Composite Structures*, 194:633 – 642, 2018.
15. Amir H Gandomi and Xin-She Yang. Chaotic bat algorithm. *Journal of Computational Science*, 5(2):224–232, 2014.
16. Amir Hossein Gandomi, X-S Yang, S Talatahari, and Amir Hossein Alavi. Firefly algorithm with chaos. *Communications in Nonlinear Science and Numerical Simulation*, 18(1):89–98, 2013.
17. SS Gokhale and VS Kale. An application of a tent map initiated chaotic firefly algorithm for optimal overcurrent relay coordination. *International Journal of Electrical Power & Energy Systems*, 78:336–342, 2016.
18. Anthony JF Griffiths, Susan R Wessler, Richard C Lewontin, William M Gelbart, David T Suzuki, Jeffrey H Miller, et al. *An introduction to genetic analysis*. Macmillan, 2005.
19. Wei-Chiang Hong. Traffic flow forecasting by seasonal svr with chaotic simulated annealing algorithm. *Neurocomputing*, 74(12-13):2096–2107, 2011.
20. Sk. Minhazul Islam, Swagatam Das, Saurav Ghosh, Subhrajit Roy, and Ponnuthurai Nagaratnam Suganthan. An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42:482–500, 2012.
21. Dongli Jia, Guoxin Zheng, and Muhammad Khurram Khan. An effective memetic differential evolution algorithm based on chaotic local search. *Information Sciences*, 181(15):3175–3187, 2011.

22. Mustafa Servet Kiran and Oguz Findik. A directed artificial bee colony algorithm. *Appl. Soft Comput.*, 26:454–462, 2015.
23. T Warren Liao. Two hybrid differential evolution algorithms for engineering design optimization. *Applied Soft Computing*, 10(4):1188–1199, 2010.
24. Zhanshan Sam Ma. Chaotic populations in genetic algorithms. *Applied Soft Computing*, 12(8):2409–2424, 2012.
25. Mukul Majumdar, Tapan Mitra, and Kazuo Nishimura. *Optimization and chaos*, volume 11. Springer Science & Business Media, 2000.
26. Michalis Mavrovouniotis, Changhe Li, and Shengxiang Yang. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation*, 33:1–17, 2017.
27. Ji Mingjun and Tang Huanwen. Application of chaos in simulated annealing. *Chaos, Solitons & Fractals*, 21(4):933–941, 2004.
28. U. Mlakar, J. Brest, and I. Fister. A study of chaotic maps in differential evolution applied to gray-level image thresholding. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, 2016.
29. Edward Ott. *Chaos in dynamical systems*. Cambridge university press, 2002.
30. Narendra K. Pareek, Vinod Patidar, and Krishan K. Sud. Image encryption using chaotic logistic map. *Image Vision Comput.*, 24:926–934, 2006.
31. Chunhua Peng, Huijuan Sun, Jianfeng Guo, and Gang Liu. Dynamic economic dispatch for wind-thermal power system using a novel bi-population chaotic differential evolution algorithm. *International Journal of Electrical Power & Energy Systems*, 42(1):119–126, 2012.
32. A. Kai Qin and Ponnuthurai Nagaratnam Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. *2005 IEEE Congress on Evolutionary Computation*, 2:1785–1791 Vol. 2, 2005.
33. R Venkata Rao and GG Waghmare. A new optimization algorithm for solving complex constrained design optimization problems. *Engineering Optimization*, 49(1):60–83, 2017.
34. R.V. Rao, V.J. Savsani, and D.P. Vakharia. Teaching–learning-based optimization: An optimization method for continuous non-linear large scale problems. *Information Sciences*, 183(1):1–15, 2012.
35. Shahrzad Saremi, Seyedali Mirjalili, and Andrew Lewis. Biogeography-based optimisation with chaos. *Neural Computing and Applications*, 25(5):1077–1097, 2014.
36. Soham Sarkar, Swagatam Das, and Sheli Sinha Chaudhuri. Multilevel image thresholding based on tsallis entropy and differential evolution. *Lecture Notes in Computer Science*, pages 17–24, 2012.
37. Roman Senkerik, Michal Pluhacek, Ivan Zelinka, Donald Davendra, and Jakub Janostik. Preliminary study on the randomization and sequencing for the chaos embedded heuristic. In *AECIA*, 2015.
38. Roman Senkerik, Michal Pluhacek, Ivan Zelinka, Adam Viktorin, and Zuzana Komínková Oplatková. Hybridization of multi-chaotic dynamics and adaptive control parameter adjusting jde strategy. 2016.
39. Roman Senkerik, Adam Viktorin, Michal Pluhacek, Tomas Kadavy, and Zuzana Komínková Oplatková. Differential evolution and chaotic series. pages 1–5, 2018.
40. Julien Clinton Sprott. *Chaos and time-series analysis*, volume 69. Oxford University Press, 2003.
41. R. Storn and K.V. Price. Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, ICSI, USA, 1995.
42. Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.
43. Ryoji Tanabe and Alex S. Fukunaga. Success-history based parameter adaptation for differential evolution. *2013 IEEE Congress on Evolutionary Computation*, pages 71–78, 2013.
44. Ryoji Tanabe and Alex S. Fukunaga. Improving the search performance of shade using linear population size reduction. *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 1658–1665, 2014.
45. Dongping Tian and Zhongzhi Shi. Mpso: Modified particle swarm optimization and its applications. *Swarm and Evolutionary Computation*, 41:49 – 68, 2018.

46. A. Viktorin, M. Pluhacek, and R. Senkerik. Success-history based adaptive differential evolution algorithm with multi-chaotic framework for parent selection performance on cec2014 benchmark set. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 4797–4803, 2016.
47. Adam Viktorin, Dusan Hrabec, and Michal Pluhacek. Multi-chaotic differential evolution for vehicle routing problem with profits. In *ECMS*, 2016.
48. Xiaohua Wang and Haibin Duan. A hybrid biogeography-based optimization algorithm for job shop scheduling problem. *Computers & Industrial Engineering*, 73:96–114, 2014.
49. Ieong Wong, Wenjia Liu, Chih-Ming Ho, and Xianting Ding. Continuous adaptive population reduction (capr) for differential evolution optimization. *SLAS technology*, 22 3:289–305, 2017.
50. Xuefeng F Yan, Dezhao Z Chen, and Shangxu X Hu. Chaos-genetic algorithms for optimizing the operating conditions based on rbf-pls model. *Computers & Chemical Engineering*, 27(10):1393–1404, 2003.
51. J. Zhang and A. C. Sanderson. Jade: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958, 2009.