J Glob Optim (2011) 51:607–640
DOI 10.1007/s10898-011-9646-9

Self-adaptive randomized and rank-based differential evolution for multimodal problems

Onay Urfalioglu · Orhan Arikan

Received: 23 March 2010 / Accepted: 3 January 2011 / Published online: 15 January 2011

Abstract Differential Evolution (DE) is a widely used successful evolutionary algorithm (EA) based on a population of individuals, which is especially well suited to solve problems that have non-linear, multimodal cost functions. However, for a given population, the set of possible new populations is finite and a true subset of the cost function domain. Furthermore, the update formula of DE does not use any information about the fitness of the population. This paper presents a novel extension of DE called Randomized and Rank-based Differential Evolution (R2DE) and its self-adaptive version SAR2DE to improve robustness and global convergence speed on multimodal problems by introducing two multiplicative terms in the DE update formula. The first term is based on a random variate of a Cauchy distribution, which leads to a randomization. The second term is based on ranking of individuals, so that R2DE exploits additional information provided by the population fitness. In extensive experiments conducted with a wide range of complexity settings, we show that the proposed heuristics lead to an overall improvement in robustness and speed of convergence compared to several global optimization techniques, including DE, Opposition based Differential Evolution (ODE), DE with Random Scale Factor (DERSF) and the self-adaptive Cauchy distribution based DE (NSDE).

Keywords Differential evolution · Cauchy distribution · Ranking · Randomization · Optimization

This work was funded by the Turkish Scientific and Technical Research Council (TUBITAK).

O. Urfalioglu (✉) · O. Arikan
Department of Electrical and Electronics Engineering, Bilkent University, 06800 Ankara, Turkey
e-mail: onay@ee.bilkent.edu.tr

O. Arikan
e-mail: oarikan@ee.bilkent.edu.tr

1 Introduction

Within the class of Evolutionary Algorithms (EA's), Differential Evolution (DE) [22,33] is one of the most robust, fastest [34] and easily implementable methods. It has only three control parameters, including the population size. A striking property of DE is that it incorporates self-adaptation by automatically scaling the search area on each phase of the global search process, which makes DE an efficient global optimizer. One important application domain of EA's is the optimization of multimodal functions. For many problems, the required number of function evaluations increases exponentially with the search space dimension. Therefore, the efficiency of an EA determines the practical limit at which applications based on those problems can be realized.

In the literature, DE is the subject of improvement in several publications. In two different works, Liu and Lampinen [15] and Brest et al. [5], introduce methods for on-line self-adaptation of DE's control parameters for mutation and crossover. Another self-adaptive DE is proposed in [20]. In a more general framework [25], Qin et al. propose the adaptation of several strategies and their control parameters at the same time. In [38], Teo applies self-adaptation to the population size. In [1], Ali and Törn propose an auxiliary population and the automatic calculation of the mutation scale factor. Tasoulis et al. [37] introduce parallel DE, where the population is divided into sub-populations, and each sub-population is assigned to a different processor node. In [30], Shi et al. propose the so called cooperative co-evolutionary differential evolution, where multiple cooperating sub-populations are used and high dimensional search spaces are partitioned into smaller spaces. Other methods for improving DE are based on hybridization. In [36], Sun et al. propose a hybrid algorithm using an estimation of distribution method. This method is based on a probability model to generate additional solution candidates. Noman and Iba [19] propose a local search to accelerate the fine tuning phase of DE based on fittest individual refinement which is a crossover-based local search. In [9], Fan and Lampinen introduced another local search - DE hybrid, which is called trigonometric mutation, in order to obtain a better tradeoff between convergence speed and robustness. Kaelo and Ali [12] introduce reinforcement-learning-based DE where different schemes for the generation of candidate vectors are proposed.

Another approach called Opposition Based Differential Evolution (ODE) based on oppositional numbers is presented by Rahnamayan et al. [26]. In another work, Das et al. [8], utilize neighborhood information of individuals and introduce schemes which balance the exploration and exploitation abilities of DE. In [7], two variants are proposed for the classical DE: DE with Random Scale Factor (DERSF) and DE with Time Varying Scale Factor (DETVSF). In DERSF, the mutation scale factor for the difference vector is replaced by a uniformly distributed random variable, whereas in DETVSF, the mutation scale factor decreases with the number of iterations. Random scaling is also discussed in [24], Sect. 2.5.2, p. 79. For noisy optimization problems, other DE-variants are proposed in [6]. In [48], a chaos based parameter update scheme is introduced to DE.

DE variants with Normal distributed or Cauchy distributed mutation operators are described and analyzed in [27,32,44,45,47]. Another DE-variant called NSDE proposed in [31] uses Cauchy-distributed scale factors and self-adaptation to dynamically adjust some parameters.

The proposed methods, called Randomized and Rank-based Differential Evolution (R2DE) and its self-adaptive version SAR2DE, integrate two distinct concepts in producing the new population: randomization and the utilization of ranking. DE has the property that the set of possible candidate vectors, which contains all possible results of mutation and crossover given a population, is finite. Furthermore, the support of the distribution of the

candidate vectors is finite too. The effect of the randomization is that these attributes become effectively continuous. The second concept takes advantage of the fitness information of each individual. This information is not used in classical DE's mutation and crossover operators. On a wide range of problems, we show experimentally that these concepts generally improve the efficiency of the global search when applied to DE.

In this work, we compare the performance of the proposed approaches to that of DE, DERSF, ODE and NSDE on scalable multimodal problems. DERSF is chosen in order to compare its uniformly distributed scale factor to our proposed Cauchy distributed scale factor. ODE is chosen to compare the proposed additional heuristics to a completely different heuristic and NSDE is chosen to compare the efficiency of SAR2DE, since NSDE also comprises a Cauchy distributed scale factor. In the experiments, we show the *tendency* of the global search efficiency of each method by increasing the number of dimensions of the search space or varying other complexity parameters, depending on the problem. Since some methods may be slower in a low dimensional setting but may become more efficient than the compared method in a higher dimension, taking only one single dimension or complexity parameter into account is not enough and can lead to wrong conclusions.

The paper is organized as follows. The following Sect. 2 briefly reviews DE. Section 3 introduces the proposed R2DE and SAR2DE methods, followed by Sect. 4, where an overview of all other compared DE-variants is given. In Sect. 5, experimental results are presented, and the paper is concluded in Sect. 6.

2 Brief review of differential evolution

DE is one of the best general purpose evolutionary global optimization methods available. It is known as an efficient global optimization method for continuous problem spaces. The optimization is based on a population of N_p solution candidates $\mathbf{x}_i$, $i \in \{1, \ldots, N_p\}$ where each candidate has a position in the D-dimensional search space. Initially, the solution candidates are generated randomly according to a uniform distribution within the provided intervals of the search space. The population improves by generating new positions iteratively for each candidate. For each individual $\mathbf{x}_{i,G}$, new trial positions $\mathbf{u}$ are determined by

$$\mathbf{v} = \mathbf{x}_{r_1,G} + F \cdot (\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}) \tag{1}$$

$$\mathbf{u} = C\left(\mathbf{x}_{i,G},\ \mathbf{v}\right), \tag{2}$$

where r_1, r_2, r_3 are pairwise different randomly chosen integers from the discrete set $\{1, \ldots, N_p\}$ and $F > 0$ is the mutation scale factor. The vector $\mathbf{v}$ is used together with $\mathbf{x}_{i,G}$ in the crossover operation, denoted by $C()$. The crossover operator copies coordinates from both $\mathbf{x}_{i,G}$ and $\mathbf{v}$ in order to create the candidate vector $\mathbf{u}$. The probability C_r mediates the crossover operation C to copy coordinates from $\mathbf{x}_{i,G}$. In the other case, coordinates from $\mathbf{v}$ are copied with the probability of $1 - C_r$ to $\mathbf{u}$. Only if the new candidate $\mathbf{u}$ proves to have an equal or lower cost then it replaces $\mathbf{x}_{i,G}$, otherwise it is discarded.

DE includes an adaptive range scaling for the generation of solution candidates through the difference term in Eq. (1). This leads to a global search with large step sizes in the case where the solution candidate vectors are widely spread within the search space due to a relatively large mean difference vector. In the case of a converging population, the mean difference vector becomes relatively small and this enables efficient fine tuning at the final phase of the optimization process. The crossover operator has a complicated role in the dynamics of

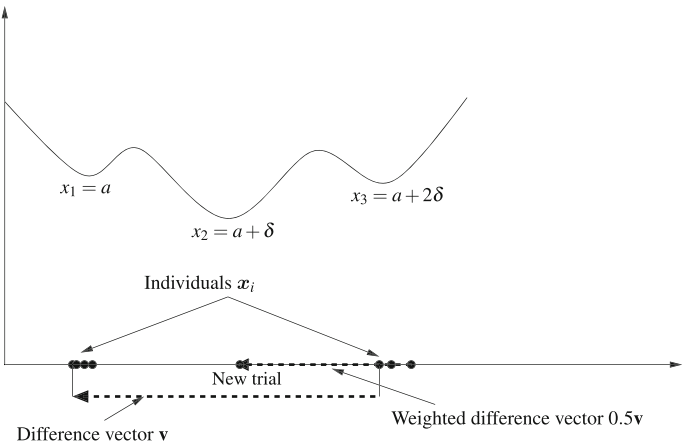

Fig. 1 In this 1-D example of *regularly distributed* local optima at $x_1 = a,\ x_2 = a + \delta,\ x_3 = a + 2\delta$, the additive weighted difference vectors yield, with high probability, new solution candidates which are located in the vicinity of another local optimum (assuming mutation scale factor $F = 0.5$)

the population. In some cases, it can help to increase the diversity of the population or it can also speed up the convergence, depending on the problem.

In case of regularly distributed local optima, due to its differential nature, the mutation scheme of DE is particularly advantageous. We define the regularity of a distribution by

$$\mathbf{x} \text{ and } \mathbf{x} + \delta \text{ are local optima} \Rightarrow \mathbf{x} + 2\delta \text{ is also a local optimum.} \tag{3}$$

For a differentiable function $f(\mathbf{x})$, the following condition is equivalent to (3):

$$\exists \delta : \frac{df(\mathbf{x})}{d\mathbf{x}} = \frac{df(\mathbf{x}+\delta)}{d\mathbf{x}} = \mathbf{0} \Rightarrow \frac{df(\mathbf{x}+2\delta)}{d\mathbf{x}} = \mathbf{0}. \tag{4}$$

During the convergence process, there is a high probability that individuals are located within the basins of the local optima. Therefore, the difference vectors are generated approximately between the basins of two selected local optima. In a mesh-like distribution of the local optima, depending on the mutation scale factor F, the resulting new position of an individual hits the area around the basin of another local optimum with high probability. In a one dimensional example, Fig. 1 illustrates this property of DE's mutation scheme.

On the other hand, this scheme can become inefficient on search spaces with non-regular structures, where local optima have a non-regular distribution. However, this scheme is only one possible aspect of the population dynamics, which is generally a complex matter.

2.1 Separability of functions and the role of Crossover

Some of the cost functions, like Rastrigin or Zeldasine considered in this paper are separable, i.e., each parameter of the function can be optimized independently. A separable function $f_s(\mathbf{x})$ can be written as [11]:

$$f_s(\mathbf{x}) = \prod_{j=1}^{D} f_j(x_j). \tag{5}$$

Applying the Logarithm on both sides of this condition provides the following equivalent form of separability:

$$f_s(\mathbf{x}) = \sum_{j=1}^{D} f_j(x_j). \tag{6}$$

Low values for C_r are the most effective for such functions, as also shown in Sect. 5.4. High values for C_r (e.g. $C_r = 0.9$) are best suited for optimizing functions with dependent parameters (like Rosenbrock). $C_r = 0.9$ is recommended because functions with dependent parameters comprise the general case and there are faster methods for optimizing separable functions that are based on decomposition. See also [24], p. 97 for a short discussion of C_r's role in optimization. However, whether the cost function is separable or not is often not known a priori. Self-adaptive versions of DE such as [31] can adapt C_r to lower values and are therefore superior in such cases.

3 Randomized and rank-based differential evolution (R2DE) and self-adaptive R2DE (SAR2DE)

The modifications of DE which make up R2DE are twofold. Two new multiplicative terms are introduced in the update formula in Eq. (1). The first term is a random variable λ with a heavy tailed distribution. Here, we will only consider the case where λ has a Cauchy distribution, which has the following density:

$$f(\lambda) = \frac{1}{\pi(1+\lambda^2)}, \lambda \in \mathbb{R}. \tag{7}$$

Although the density function of the Cauchy distribution has its maximum at zero, due to its heavy tailed nature, the Cauchy distribution has no finite moments and it is very likely to have samples which differ significantly from zero. The motivation for this term is to 'fill in' the gaps in the set of possible candidate vectors produced by DE's mutation operator. This way, the mentioned set becomes continuous, which also helps to increase the diversity of the population.

The second term α, which is in (0, 1] interval, is defined as:

$$\alpha(\mathbf{x}_{r_1,G}) = 1 - \frac{k(\mathbf{x}_{r_1,G})}{N_p}, \tag{8}$$

where $k(\mathbf{x}_{r_1,G})$ is the rank of the individual $\mathbf{x}_{r_1,G}$. Assuming the global minimum is searched for, the best individual with minimal cost has rank 0, whereas the worst individual has rank $N_p - 1$. This term reflects the fact that, on minimization of multimodal functions, typically, we need to explore a relatively large area to improve upon a relatively small cost function value. Figure 2 shows an example with two cost-levels and correponding step lengths required to reach a new basin with potentially lower cost. The R2DE update formula for the generation of candidate vectors is given by

$$\mathbf{v} = \mathbf{x}_{r_1,G} + F \cdot \lambda_i \cdot \alpha(\mathbf{x}_{r_1,G}) \cdot (\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}) \tag{9}$$

$$\mathbf{u} = C\left(\mathbf{x}_{i,G},\ \mathbf{v}\right), \tag{10}$$

where $\alpha(\mathbf{x}_{r_1,G})$ depends on $\mathbf{x}_{r_1,G}$ and λ_i is sampled independently for each individual $\mathbf{x}_{i,G}$ at each iteration.

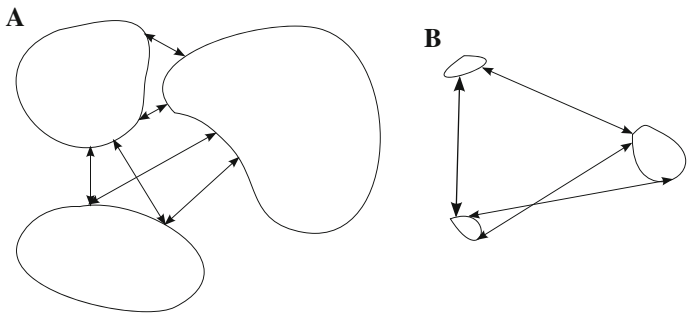

Fig. 2 In these plots, *closed curves* represent regions of a 2-*D* function with constant cost *A* (*left*) and *B* (*right*), where $A > B$. *The arrows* show possible required step lengths for these cost-levels *A* and *B* to reach the basin of a local optimum from the basin of other local optima. For multimodal cost functions, as in this case, the mean distances typically increase for decreasing cost levels. This means individuals having lower cost require greater steps (relatively) to reach the basin of a potentially better local optimum

Due to the ranking, R2DE comprises a slightly higher runtime complexity $O(N_p \log(N_p))$ than DE, which has complexity $O(N_p)$.

We also propose a self-adaptive version of R2DE called SAR2DE, which is motivated by NSDE [31]. In SAR2DE, the vector of function parameters is extended by two additional parameters ε and γ. These two parameters have special update formulas:

$$\varepsilon_{\text{next}} = \begin{cases} 2 \cdot U_\varepsilon & \text{for } 0.1 \geq U_{t,\varepsilon} \\ \varepsilon & \text{else} \end{cases}, \quad \gamma_{\text{next}} = \begin{cases} U_\gamma & \text{for } 0.1 \geq U_{t,\gamma} \\ \gamma & \text{else} \end{cases}, \tag{11}$$

where for each update, the random variables $U_\varepsilon, U_\gamma, U_{t,\varepsilon}, U_{t,\gamma}$ are generated using an independent uniform distribution in [0,1] interval. The ε parameter generalizes the ranking parameter α, and the γ parameter makes the crossover probability adaptive. Finally, the SAR2DE update formula for the generation of the regular part of the candidate vectors is given by

$$\mathbf{v} = \mathbf{x}_{r_1,G} + F \cdot \lambda_i \cdot (\alpha(\mathbf{x}_{r_1,G}))^{\psi \varepsilon_{\text{next}}} \cdot (\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}) \tag{12}$$

$$\mathbf{u} = C\left(\mathbf{x}_{i,G},\ \mathbf{v}\right), \ [\mathbf{u} \text{ is extended by } \varepsilon_{\text{next}} \text{ and } \gamma_{\text{next}}], \tag{13}$$

with $C_r = \gamma_{\text{next}}$ and $\psi = \log(1 + \lambda_i)$. As in DE, $\mathbf{u}$ replaces $\mathbf{x}_{i,G}$ only if it proves to have an equal or lower cost. As a result, only those adaptation parameters ε and γ survive which prove to enable the generation of better candidates. The motivation for $\alpha^{\psi\varepsilon}$ is to be able to adaptively adjust the rank-based weighting to each problem. As a special case, $\varepsilon = 0$ switches off the rank-based weighting. The term $\varepsilon\psi$ regulates the overall scale factor $F\lambda\alpha^{\log(1+\lambda)\varepsilon}$. The main effect of the term $\varepsilon\psi$ is that the overall scale factor has an upper bound for small values of α. Figure 3 shows three plots for the overall scale factor with $\alpha = 0.2, \alpha = e^{-1} \approx 0.368$ and $\alpha = 0.6$, all with $\varepsilon = 1$, depending on the Cauchy-distributed random variable λ. As a result, in contrast to R2DE, the heavy-tail property caused by λ can be 'switched off' for high-ranked individuals which yield relatively large cost function results. On the other hand, it is kept 'on' for the other individuals. This further supports the heuristic of larger step sizes for lower-ranked individuals. In other words, according to Fig. 2, individuals having a small α-value (case A) tend to have a bounded, non-heavy-tailed overall scale factor. On the other hand, individuals having a large α-value (case B) tend to have an unbounded,

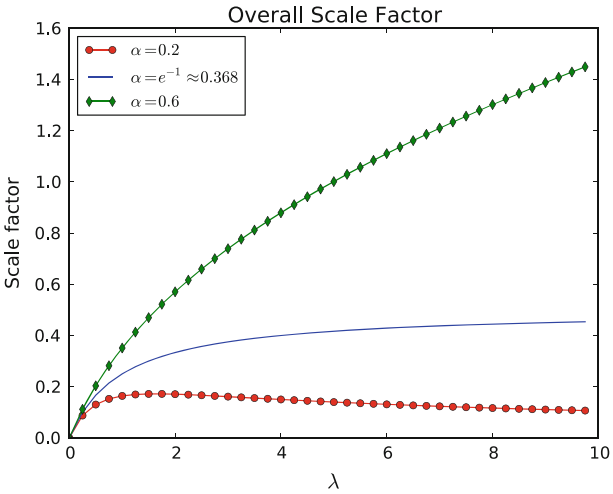

Fig. 3 Three cases for the overall scale factor are plotted, depending on λ. For all cases, it is $\varepsilon = 1$. In case $\alpha = 0.2$, the overall scale factor has an upper bound, so that the heavy-tail property is no longer given. In case $\alpha = e^{-1}$, the overall scale factor is still bounded and converges to $F = 0.5$. Finally, case $\alpha = 0.6$ leads to an unbounded overall scale factor, having the heavy-tail property

heavy-tailed overall scale factor. In order to determine for which α and ε the overall scale factor is upper-bounded, we write the overall scale factor as

$$\lambda\alpha^{\varepsilon \log(1+\lambda)} = \exp[\log(\lambda) + \varepsilon \log(\alpha) \log(1+\lambda)]. \tag{14}$$

Since $\log(1+\lambda) = \log(\lambda)$ for $\lambda \to \infty$, it follows

$$\lim_{\lambda\to\infty} \exp[\log(\lambda) + \varepsilon \log(\alpha) \log(1+\lambda)] = \lim_{\lambda\to\infty} \exp[\log(\lambda)(1+\varepsilon \log(\alpha))] \tag{15}$$

$$= \begin{cases} 0 & \text{for } 1+\varepsilon\log(\alpha) < 0 \Leftrightarrow 0 < \alpha < \exp(-1/\varepsilon) \\ 1 & \text{for } 1+\varepsilon\log(\alpha) = 0 \Leftrightarrow \alpha = \exp(-1/\varepsilon) \\ \infty & \text{for } 1+\varepsilon\log(\alpha) > 0 \Leftrightarrow \alpha > \exp(-1/\varepsilon). \end{cases} \tag{16}$$

As a result, the overall scale factor is upper-bounded for $\alpha \leq \exp(-1/\varepsilon)$. This means that the parameter ε controls which α-values are assigned to the heavy-tail property. For $\varepsilon = 0$, all α-values lead to a heavy-tailed overall scale factor. For $\varepsilon = 2$, which is the maximum value of ε, the heavy-tail property is given for $\alpha > e^{-1/2} \approx 0.607$.

The adaptation of the crossover probability by γ is the same as found in [31]. One of the advantages of an adaptive C_r is that given a separable problem, $C_r = \gamma$ may be adapted to become small, which enables a more effective search.

On all considered benchmark functions, we apply the following transform to limit vector components x_i into a feasible region $[L, R]$.

$$x_i = \begin{cases} L + (L - x_i) \bmod (R - L) & \text{for } x_i < L \\ R - (x_i - R) \bmod (R - L) & \text{for } x_i > R \end{cases} \tag{17}$$

where mod is the modulo operator. As an example, given a feasible region of [0, 1], $x_1 = 1.2$ becomes 0.8.

4 Overview of benchmarked DE-variants

In order to evaluate the proposed methods R2DE and SAR2DE, we conduct comparisons with the methods Opposition-based Differential Evolution (ODE) [26], Differential Evolution with Random Scale Factor (DERSF) [7] and A Self-Adaptive Strategy for Controlling Parameters in Differential Evolution (NSDE) [31]. In the following, we give a short introduction to each of these algorithms.

4.1 Opposition-based Differential Evolution (ODE)

Opposition-based Differential Evolution (ODE) is motivated by opposition-based learning. The main idea behind this is to consider an estimate and its corresponding opposite estimate simultaneously. Instead of a single estimate, two estimates are to be evaluated. From the probability theory follows that 50% of the time a guess is further from the solution than its opposite guess. Therefore, starting with the closer of the two guesses (as judged by its fitness) has the potential to accelerate convergence. The same approach can be applied not only to initial solutions but also continuously to each solution in the current population. ODE is chosen for comparisons due to its additional opposition learning-based heuristic, which often enables a more efficient search than DE.

4.2 Differential Evolution with Random Scale Factor (DERSF)

Differential Evolution with Random Scale Factor (DERSF) is one of the first DE-variants incorporating the randomization of the scale factor F. In DERSF, the scale factor is not constant but is a random variable with a uniform distribution. DERSF is chosen to compare the different randomization methods of the scale factor.

4.3 A Self-Adaptive Strategy for Controlling Parameters in Differential Evolution (NSDE)

A Self-Adaptive Strategy for Controlling Parameters in Differential Evolution (NSDE) is based on ideas of self-adaptation. The control parameters F and C_r in DE are constant. In several works, it is shown that optimal values for these parameters heavily depend on the problem. Therefore, self-adaptation of these parameters is achieved by extending the candidate vectors by additional parameters to adapt F and C_r. These additional parameters are subject to evolutionary mutation and selection, where better parameters, which map to better values for F and C_r, survive over time. NSDE is chosen for comparisons with the self-adaptive variant of R2DE, named SAR2DE, since it is based on Cauchy distributed random scale factor too. To obtain a fair comparison, we choose $F = 0.5$ as the center of the Cauchy distribution.

5 Experiments

The experiments contain 19 scalable multimodal global optimization problems and an artificial neural network (ANN) problem (though a smaller number of problems is generally acceptable for this purpose, e.g., [13]). For all experiments, unless mentioned otherwise, the utilized settings for the parameters are given by

- $F = 0.5$ (as in [1,5,15,26,35,42])
- $C_r = 0.9$ (as in [1,5,15,26,35,42])
- mutation strategy: DE/rand/1/bin (classic DE) (as in [5,21,24,26,35,36])
- value-to-reach (VTR) $= f(\mathbf{x}^*) + 10^{-6}$,

where the global optimum of each problem is denoted by $\mathbf{x}^*$. We compare the performance of the proposed R2DE method with those of DE, ODE [26], DERSF [7], DE with the λ-factor, denoted as DE-λ and DE with the α-factor, denoted as DE-α. To provide experimental support for the proposed rank-based factor $\alpha(\mathbf{x})$, we conduct further experiments by applying a 'reversed' rank-based heuristic, using the factor $1 - \alpha(\mathbf{x})$ instead of $\alpha(\mathbf{x})$. Additionally, we also compare the performance of the self-adaptive method NSDE with the proposed SAR2DE.

5.1 Benchmark suite

In the following, 19 multimodal problems are introduced for experiments.

5.1.1 Alpine function

The Alpine function (f_1) consists of multiple global optima and local maxima. One global minimum is at $f_1(\mathbf{0}) = 0$. The number of local optima increases exponentially with the dimension. It is also used as a benchmark function in [26].

$$f_1(\mathbf{x}) = \sum_{j=1}^{D} |x_j \sin(x_j) + 0.1x_j|$$
$$x_j \in [-10, 10], \; f_1(\mathbf{x}^*) = 0. \tag{18}$$

Plese note that there are multiple global optima of the Alpine function, one of them is $\mathbf{x}^* = \mathbf{0}$.

5.1.2 Cosine mixture

The Cosine Mixture function (f_2) has one global optimum. The number of local optima increases exponentially with the dimension. This function was also used as a benchmark function in [2,4].

$$f_2(\mathbf{x}) = -0.1 \sum_{j=1}^{D} \cos(5\pi x_j) + \sum_{j=1}^{D} x_j^2$$
$$x_j \in [-1, 1], \; f_2(\mathbf{x}^* = \mathbf{0}) = -0.1D \tag{19}$$

5.1.3 Epistatic Michalewicz function

The Epistatic Michalewicz function (f_3) (second ICEO) has one global optimum and an exponentially increasing (with dimension) number of local optima. The location of the global optimum coordinates depends on the dimension. This function is also used as a benchmark function in [36].

$$y_{2j-1} = x_{2j-1} \cos(\pi/6) - x_{2j} \sin(\pi/6)$$

$$y_{2j} = x_{2j-1}\sin(\pi/6) + x_{2j}\cos(\pi/6),\ \ j = 1, \ldots, D$$
$$\text{if } D \text{ is odd number: } y_D = x_D$$
$$f_3(\mathbf{y}) = \sum_{j=1}^{D} -\sin(y_j)\cdot\left(\sin\left(jy_j^2/\pi\right)\right)^{20}$$
$$x_j \in [0, \pi]. \tag{20}$$

D	$\mathbf{x}^*$	VTR
5	(2.693, 0.258, 2.074, 1.022, 1.720)	−4.68765
6	(2.693, 0.258, 2.074, 1.022, 2.275, 0.5)	−5.68765
7	(2.693, 0.258, 2.074, 1.022, 2.275, 0.5, 1.458)	−6.68088
8	(2.693, 0.258, 2.074, 1.022, 2.275, 0.5, 2.137, 0.793)	−7.66375
9	(2.693, 0.258, 2.074, 1.022, 2.275, 0.5, 2.137, 0.793, 1.655)	−8.66014
10	(2.693, 0.258, 2.074, 1.022, 2.275, 0.5, 2.137, 0.793, 2.219, 0.532)	−9.66014

5.1.4 Foxholes function

The Foxholes function (f_4) is generally customizable and usually has one global optimum. The location of the global optimum depends on the parameters of the function. It is also used as a benchmark function in [3].

$$f_4(\mathbf{x}) = -\sum_{j=1}^{M}\left(\sum_{k=1}^{D}[(x_k - a_{jk})^2 + c_k]\right)^{-1}$$
$$x_j \in [0, 10],\ \ M = 50 \tag{21}$$

where $c_j, a_{jk} \in [0, 10]$ are user defined numbers, which are initially sampled from a uniform distribution in this paper. The elements of c_k and a_{jk} are given in Appendix A.

D	$\mathbf{x}^*$	VTR
5	(8.625, 5.285, 6.203, 1.657, 1.196)	−4.7986
6	(0.782, 8.543, 4.427, 6.041, 1.068, 4.986)	−3.94122
7	(0.954, 1.456, 2.826, 1.361, 8.020, 8.692, 0.776)	−3.28514

5.1.5 Griewank function

The Griewank function (f_5) has the property that its complexity has a peak at a finite dimension [16], although the total number of local optima increases with the dimension.

$$f_5(\mathbf{x}) = \left(\sum_{j=1}^{D} x_j^2/4000\right) - \left(\prod_{j=1}^{D}\cos(x_j/\sqrt{j})\right) + 1$$
$$x_j \in [-600, 600],\ \ f_5(\mathbf{x}^* = \mathbf{0}) = 0. \tag{22}$$

5.1.6 Inverted cosine wave

The Inverted Cosine Wave function (f_6) has one global optimum and an exponentially increasing (with dimension) number of local optima. It is also used as a benchmark function in [26].

$$f_6(\mathbf{x}) = \sum_{j=1}^{D-1} \left\{ \exp\left(\frac{-\left(x_j^2 + x_{j+1}^2 + 0.5 x_j x_{j+1}\right)}{8} \right) \cdot \cos\left(4\sqrt{x_j^2 + x_{j+1}^2 + 0.5 x_j x_{j+1}}\right) \right\}$$
$$x_j \in [-5, 5], \; f_6(\mathbf{x}^* = \mathbf{0}) = -D + 1. \tag{23}$$

5.1.7 Michalewicz function

The Michalewicz function (f_7) has one global optimum and an exponentially increasing (with dimension) number of local optima. It is also used as a benchmark function in [26].

$$f_7(\mathbf{x}) = \sum_{j=1}^{D} -\sin(x_j) \cdot \left(\sin\left(j x_j^2/\pi\right)\right)^{20}$$
$$x_j \in [0, \pi]. \tag{24}$$

D	$\mathbf{x}^*$	VTR
5	(2.203, 1.571, 1.285, 1.923, 1.72)	−4.68765
6	(2.203, 1.571, 1.285, 1.923, 1.72, 1.571)	−5.68765
7	(2.203, 1.571, 1.285, 1.923, 1.72, 1.571, 1.454)	−6.68088
8	(2.203, 1.571, 1.285, 1.923, 1.72, 1.571, 1.454, 1.756)	−7.66375
9	(2.203, 1.571, 1.285, 1.923, 1.72, 1.571, 1.454, 1.756, 1.656)	−8.66014
10	(2.203, 1.571, 1.285, 1.923, 1.72, 1.571, 1.454, 1.756, 1.656, 1.571)	−9.66014
11	(2.203, 1.571, 1.285, 1.923, 1.72, 1.571, 1.454, 1.756, 1.656, 1.571, 1.498)	−10.6574
12	(2.203, 1.571, 1.285, 1.923, 1.72, 1.571, 1.454, 1.756, 1.656, 1.571, 1.498, 1.697)	−11.6495

5.1.8 Periodic function

The Periodic function (f_8) has one global optimum and an exponentially increasing (with dimension) number of local optima. It is also used as a benchmark function in [2,23].

$$f_8(\mathbf{x}) = \sum_{j=1}^{D} \sin^2(x_j) - 0.1 \exp\left(-\sum_{j=1}^{D} x_j^2\right),$$
$$x_j \in [-10, 10], \; f_8(\mathbf{x}^* = \mathbf{0}) = 0.9. \tag{25}$$

5.1.9 Perm function ($D = 4$)

The Perm function (f_9) has one global optimum. It has an additional parameter β, which also affects the complexity of the function. The smaller β, the more difficult this problem becomes since the global minimum is difficult to distinguish from local minima near permuted solutions. It is also used as a benchmark function in [26].

$$f_9(\mathbf{x}) = \sum_{j=1}^{D} \left[\sum_{k=1}^{D} \left(j^k + \beta \right) \left(\left[\frac{x_j}{j} \right]^k - 1 \right) \right]^2$$
$$x_j \in [-D, D],\ \beta \in \{4, 5, \ldots, 13\},\ f_9(\mathbf{x}^* = (1, 2, \ldots, D)) = 0. \quad (26)$$

5.1.10 Perm0 function ($D = 4$)

The Perm0 function (f_{10}) has one global optimum and an additional parameter β. It has similar characteristics like the Perm function 5.1.9.

$$f_{10}(\mathbf{x}) = \sum_{k=1}^{D} \left[\sum_{j=1}^{D} (j + \beta) \left(x_j^k - \left[\frac{1}{j} \right]^k \right) \right]^2$$
$$x_j \in [-1, 1],\ \beta \in \{70, 80, \ldots, 100\},\ f_{10}(\mathbf{x}^* = (1/1, 1/2, \ldots, 1/D)) = 0. \quad (27)$$

5.1.11 Rastrigin function

The Rastrigin function (f_{11}) is a widely used benchmark function having one global optimum and an exponentially increasing (with dimension) number of local optima. It is also used as a benchmark function in [26,39].

$$f_{11}(\mathbf{x}) = 10D + \sum_{j=1}^{D} x_j^2 - 10\cos(2\pi x_j)$$
$$x_j \in [-5.12, 5.12],\ f_{11}(\mathbf{x}^* = \mathbf{0}) = 0. \quad (28)$$

5.1.12 Salomon function

The Salomon function (f_{12}) is rotation symmetric and comprises no single points but regions (hyperspheres) as local optima. It has one global optimum. It is also used as a benchmark function in [2,28].

$$f_{12}(\mathbf{x}) = -\cos(2\pi||\mathbf{x}||) + 0.1||\mathbf{x}|| + 1$$
$$x_j \in [-100, 100],\ f_{12}(\mathbf{x}^* = \mathbf{0}) = 0. \quad (29)$$

5.1.13 Schaffer1 function

The Schaffer1 function (f_{13}) is rotation symmetric and comprises no single points but regions (hyperspheres) as local optima. It has one global optimum. It is also used as a benchmark function in [2,17].

$$f_{13}(\mathbf{x}) = 0.5 + \frac{\sin^2(||\mathbf{x}||) - 0.5}{1 + 0.001||\mathbf{x}||^2}$$
$$x_j \in [-100, 100],\ f_{13}(\mathbf{x}^* = \mathbf{0}) = 0. \quad (30)$$

5.1.14 Schaffer2 function

The Schaffer2 function (f_{14}) is rotation symmetric and comprises no single points but regions (hyperspheres) as local optima. It has one global optimum. It is also used as a benchmark function in [2,17].

$$f_{14}(\mathbf{x}) = ||\mathbf{x}||^{0.25}\left[\sin\left(\sin\left((50||\mathbf{x}||)^{0.1}\right)\right)+1\right]$$
$$x_j \in [-100, 100],\ f_{14}(\mathbf{x}^* = \mathbf{0}) \approx 0.00012. \tag{31}$$

5.1.15 Shifted Schaffer2 function

The Shifted Schaffer2 function (f_{15}) is the shifted version of the Schaffer2 function 5.1.14.

$$u = 100(\sqrt{2}/5 - 1) \approx -71.71573$$
$$s = \sum_{j=1}^{D}(x_j - u)^2$$
$$f_{15}(\mathbf{x}) = s^{0.25}\left[\sin\left(\sin\left((50s)^{0.1}\right)\right)+1\right]$$
$$x_j \in [-100, 100],\ f_{15}(\mathbf{x}^* = (-71.71573, \ldots, -71.71573)) \approx 0.00012. \tag{32}$$

5.1.16 Schubert function

The Schubert function (f_{16}) has multiple local and global optima [2,14]. The number of local optima increases exponentially with the dimension.

$$f_{16}(\mathbf{x}) = \prod_{j=1}^{D}\sum_{k=1}^{5} k\cos((k+1)x_j + k)$$
$$x_j \in [-10, 10]. \tag{33}$$

Plese note that there are multiple global optima of the Schubert function.

D	$\mathbf{x}^*$	VTR
2	Varies	−186.7309
3	Varies	−2709.1
4	Varies	−39303.6
5	Varies	−570215.8
6	Varies	−8.2726

5.1.17 Schwefel's problem (2.26)

The Schwefel function (2.26) (f_{17}) has one global optimum and an exponentially increasing (with dimension) number of local optima. It is also used as a benchmark function in [2].

$$f_{17}(\mathbf{x}) = \sum_{j=1}^{D} -x_j \sin(\sqrt{|x_j|})$$
$$x_j \in [-500, 500],\ f_{17}(\mathbf{x}^* = (420.9687, \ldots, 420.9687)) \approx -D \cdot 418.9829. \tag{34}$$

5.1.18 Zeldasine function

The Zeldasine function (f_{18}) has multiple local and global optima. The optima count increases exponentially with the dimension. It is also used as a benchmark function in [2,46].

$$f_{18}(\mathbf{x}) = -A \prod_{j=1}^{D} \sin(x_j - z) - \prod_{j=1}^{D} \sin(B \cdot (x_j - z))$$
$$A = 2.5,\ B = 5,\ z = \pi/6$$
$$x_j \in [-10, 10],\ f_{18}(\mathbf{x}^*) = 0. \tag{35}$$

Plese note that there are multiple global optima of the Zeldasine function.

5.1.19 Rosenbrock function

The Rosenbrock function (f_{19}) is a widely used benchmark function. According to [29], it is unimodal for $D \leq 3$ and multimodal for higher dimensions. Due to a saddle point, it is very difficult to locate the global minimum. It is also used as a benchmark function in [2,18].

$$f_{19}(\mathbf{x}) = \sum_{j=1}^{D-1} (1 - x_j)^2 + 100 \left(x_{j+1} - x_j^2\right)^2$$
$$x_j \in [-30, 30],\ f_{19}(\mathbf{x}^* = (1, \ldots, 1)) = 0. \tag{36}$$

5.1.20 Robust estimation of artificial neural network (ANN) parameters

The performance of the proposed R2DE method is investigated on the estimation of parameters of an Artificial Neural Network (ANN), which is commonly used in engineering applications. For this purpose, we utilize feed forward networks which can be described by the 1-D–1-D mapping $\Psi(x)$:

$$y = \Psi(x) = \sum_{j=1}^{N} w_j e(v_j, \tau_j, x), \tag{37}$$

where $e(v_j, \tau_j, x)$ is a sigmoidal 'basis function' or neuron:

$$e(v_j, \tau_j, x) = \frac{1}{1 + \exp(-v_j x + \tau_j)}. \tag{38}$$

The scalars w_j and v_j represent weights and τ_j is a threshold. Here, we consider a 1-D to 1-D ANN with one input neuron, N hidden layer neurons and one output neuron, which has $3N$ parameters in total. In this experiment, for the training of the ANN, we used the **sinc** data set, which contains the input/output pairs $(x_i, y_i),\ i = 1, \ldots, M$ generated by

$$x_i = -\frac{6.1}{M} i + 12,\ y_i = \frac{\sin(x_i)}{x_i} + v_i,\ i = 1, \ldots, M, \tag{39}$$

where v_i is a zero mean normal distributed random variable with standard deviation $\sigma_v = 0.001$ and $M = 30$. Additionally, the data set also contains a varying number of outlier points $(\tilde{x}_i,\ \tilde{y}_i)$ generated by:

$$\tilde{x}_i = -6.1 + 12U(0, 1),\ \tilde{y}_i = -2 + 4U(0, 1), \tag{40}$$

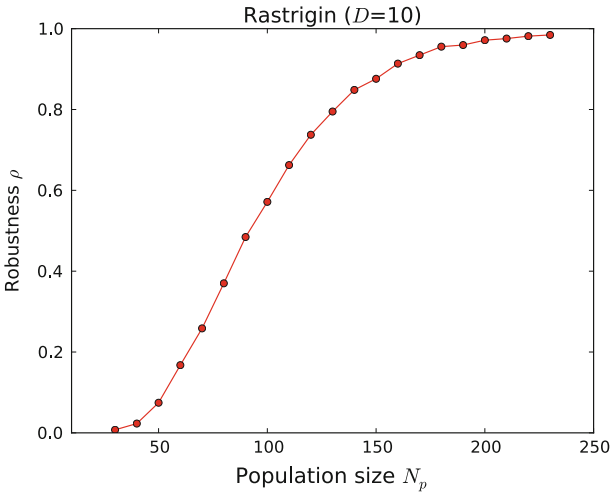

Fig. 4 Typical dependence of the robustness ρ on the population size N_p, drawn on the example of the Rastrigin cost function

where $U(0, 1)$ is a uniform random number generator for sampling numbers within [0,1]. The minimization for an ANN with $N = 4$ hidden layer neurons with 12 parameters is based on the following robust cost function [40,41,43]:

$$\Omega_R(\boldsymbol{\theta}) = \sum_{i=1}^{N} -\log\left\{\frac{0.5e^{\frac{(y_i - f(\boldsymbol{\theta}, y_i))^2}{2\cdot 10\cdot\sigma_v^2}}}{\sqrt{2\pi\cdot 10\cdot\sigma_v^2}} + \frac{0.5}{4}\right\}, \tag{41}$$

where $\boldsymbol{\theta}$ contains the parameters of the ANN and $f(\boldsymbol{\theta}, y_i)$ represents the ANN mapping. The VTR's were chosen so that the MSE using the *inliers* (points which conform to the underlying model, the opposite of outliers) only is below 0.05. The settings of the EA's are the same as in 5, and 50 independent runs are carried out on each EA and data set.

5.2 Comparison methodology

For each problem, 100 independent optimization runs were carried out at different complexity settings such as search space dimension or other specific cost function parameters. The task is to achieve a robustness (also known as success rate) of $\rho \approx 0.99$, i.e., at most one of the 100 runs may fail to find the global optimum in average. The global optimum is declared as found when the VTR is reached. Figure 4 shows the typical relationship between robustness and population size, where each measurement of ρ is the result of 2,000 independent runs. It can be seen that the sensitivity of ρ over N_p decreases, i.e., the first derivative $\frac{d\rho}{dN_p}$ becomes smaller for $\rho \approx 0.99$. We also assume that the error $\delta\rho$ of the measurement decreases at the same time. As a result, to find a $\hat{N}_p$ with $\rho \approx 0.99|_{N_p=\hat{N}_p}$, we propose to conduct 100 runs with at most 1 failure. The step size δN_p to find $\hat{N}_p$ should be chosen such that the difference of the mean function evaluations (MFE) by adjusting N_p is not greater than the standard deviation of the MFE:

$$|\mathrm{MFE}(N_p + \delta N_p) - \mathrm{MFE}(N_p)| \leq \sigma_{\mathrm{MFE}(N_p)}. \tag{42}$$

For each complexity setting on each cost function, the population size is manually adjusted to minimize the required MFE and to meet the robustness constraint of $\rho \approx 0.99$. This approach for comparison shows the scalability of each EA-method and reveals the dependence of the required population size on the robustness. We believe that this enables a compact but exhaustive analysis of the methods.

5.3 Obtained results

The Figs. 5 and 6 show the results of the comparisons between the proposed R2DE method and DE, DERSF, ODE, DE-λ and DE-α. Plots of convergence characteristics at highest complexity settings for DE and R2DE are shown in Figs. 7 and 8. Regarding the required MFE, R2DE outperforms DE at high complexity settings on 15 out of 19 problems, DERSF is outperformed in 15 out of 19 problems and ODE is outperformed in 13 out of 19 problems. In our experiments, DERSF outperforms DE on 8 out of 19 problems, but the required MFE's are generally close to DE. ODE outperforms DE in 10 out of 19 problems.

The results (MFE's) of the ANN-parameter estimation problem are shown in Fig. 9 at a robustness of $\beta = 0.98$. More detailed results including the standard deviations of the measured MFE's and corresponding t-test based hypotheses rejections are shown in Table 1. In this experiment, R2DE significantly outperforms DE. One important property of the ANN-based cost function is the permutation symmetry of the neuron-level parameter blocks [10]. Additionally, each neuron parameter block comprises a point symmetry. This means there are several partitions in the search space each with a global optimum. For K neurons in the hidden layer, there are $2^K K!$ global optima. Principally, this corresponds to the Zeldasine function (f_{18}), where R2DE also shows very good results.

It can be seen from the results that the difference of the MFE's increases with the complexity settings. Table 2 shows detailed measurements including the population sizes and the standard deviations of the MFE's. Note that R2DE generally requires a greater population of individuals to achieve the same robustness. On the other hand, it requires a much smaller number of iterations for global convergence, and outperforms DE, DERSF and ODE on the majority of the presented problems. In Table 3, results from the DE-λ and DE-α methods are compared. It can be seen that the randomization of the scale factor yields stronger improvements than the introduction of the rank-based factor. However, it should be noted that the rank-based factor rescales the Cauchy distribution, so that their combination in R2DE shows the most consistent improvement over DE, compared to DE-α and DE-λ. In Table 4, we provide experimental support for the proposed rank-based factor $\alpha(\mathbf{x})$ by applying a 'reversed' rank-based heuristic, using the factor $1 - \alpha(\mathbf{x})$ instead of $\alpha(\mathbf{x})$. Using the first 11 test-functions, it is clearly shown that the 'reversed' rank-based heuristic leads to significantly and consistently inferior results.

In Table 5, comparisons of the self-adaptive methods NSDE and the proposed SAR2DE are shown. As a general observation, NSDE outperforms SAR2DE on 4 functions, whereas SAR2DE outperforms NSDE on 10 functions at the most complex settings. On the remaining 5 functions, there is no statistically significant difference between the two methods.

In order to better classify the results, we cluster the set of utilized cost functions and provide respective results about which method performs best at each cost function in Table 6. The functions are grouped by the following properties: rotation symmetry, multiple global optima, rough sphere, (exactly or approximately) regular local optima and general. The 'rough

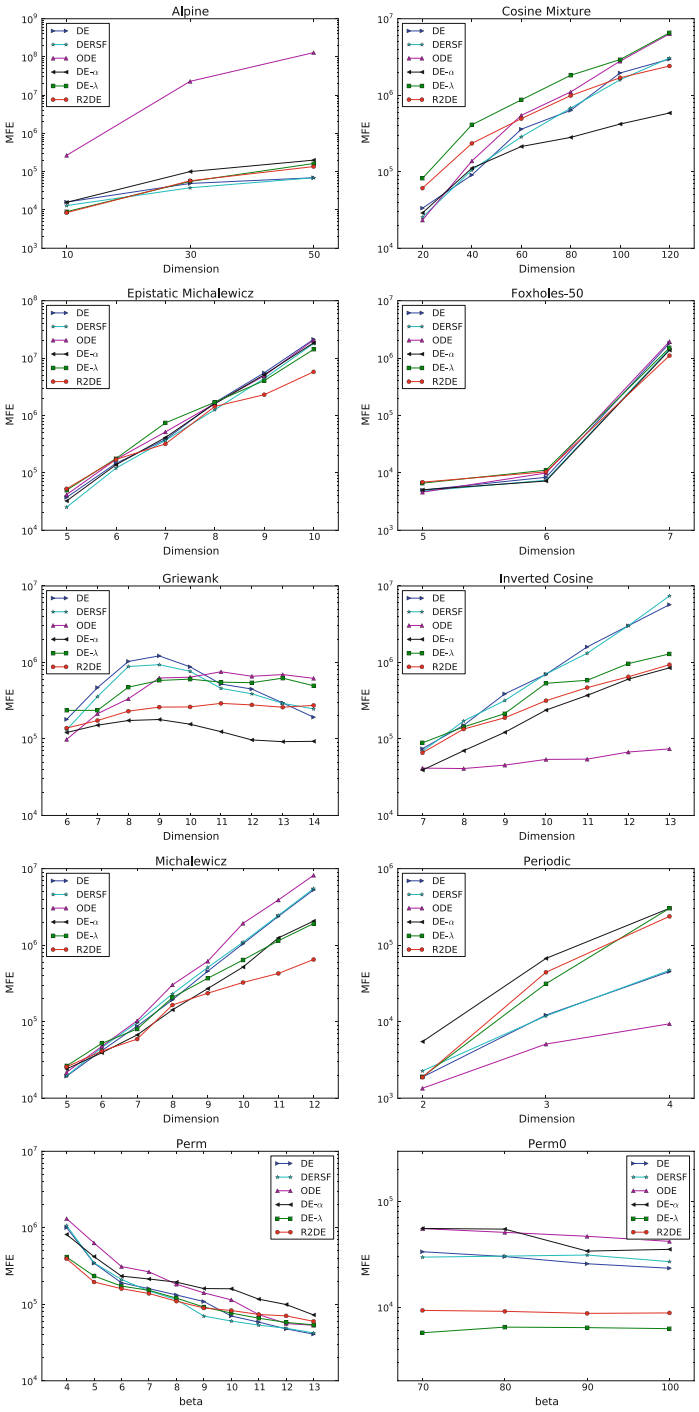

Fig. 5 Required mean function evaluations (MFE's) to find the global optimum with a robustness of $\rho \approx 0.99$

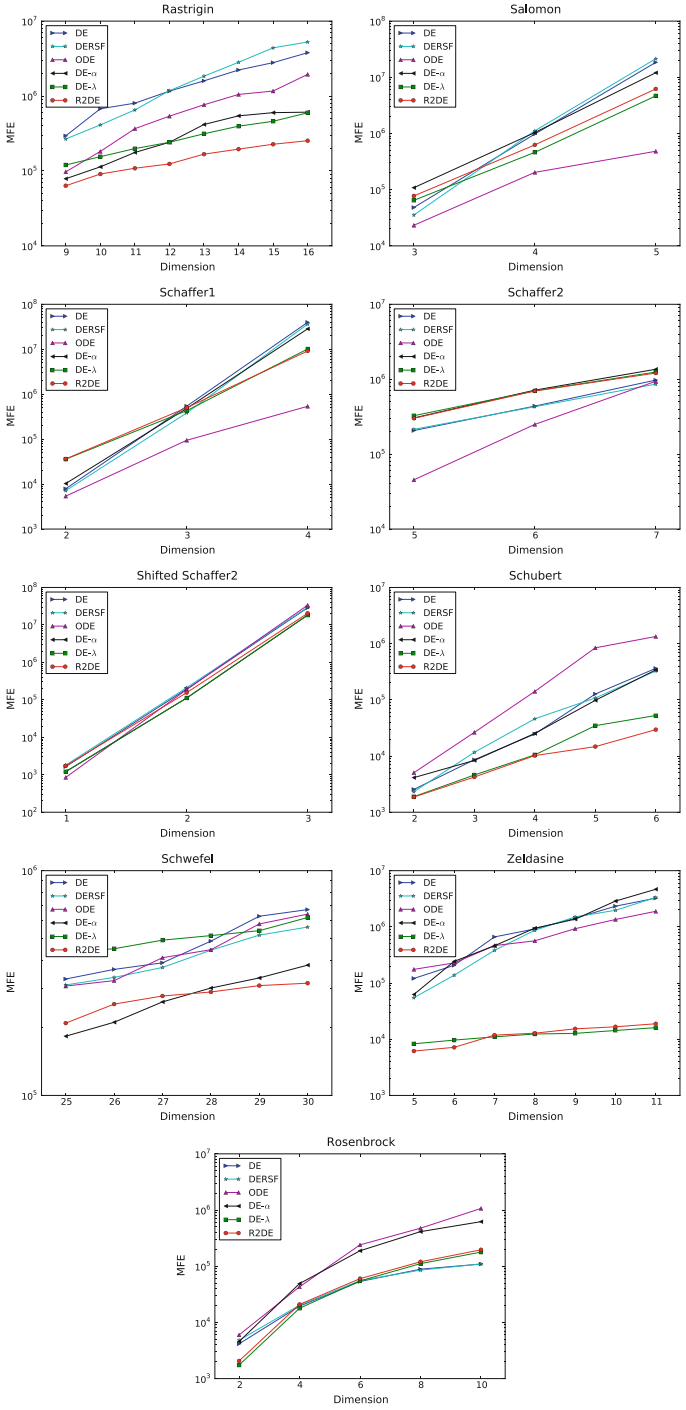

Fig. 6 Required mean function evaluations (MFE's) to find the global optimum with a robustness of $\rho \approx 0.99$

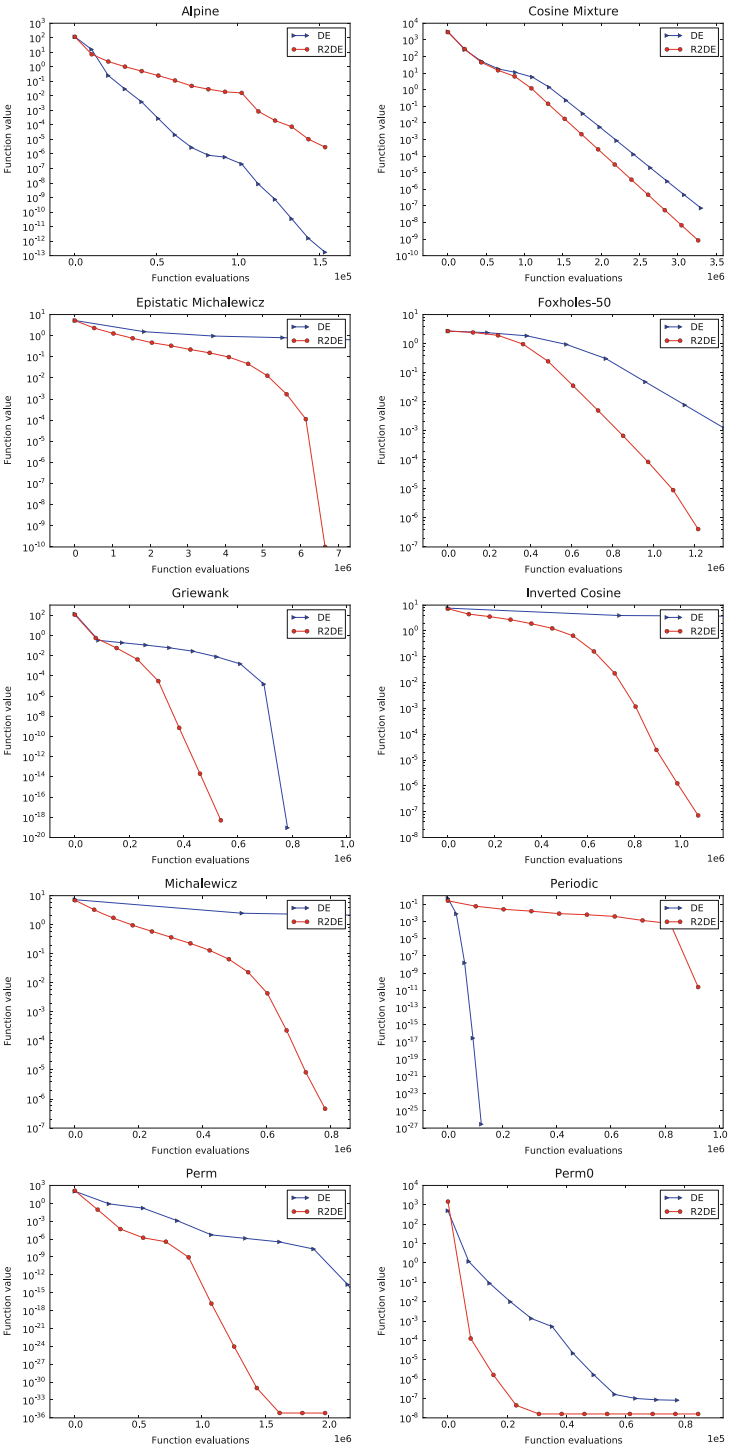

Fig. 7 Convergence plots of the cost functions at highest complexity settings

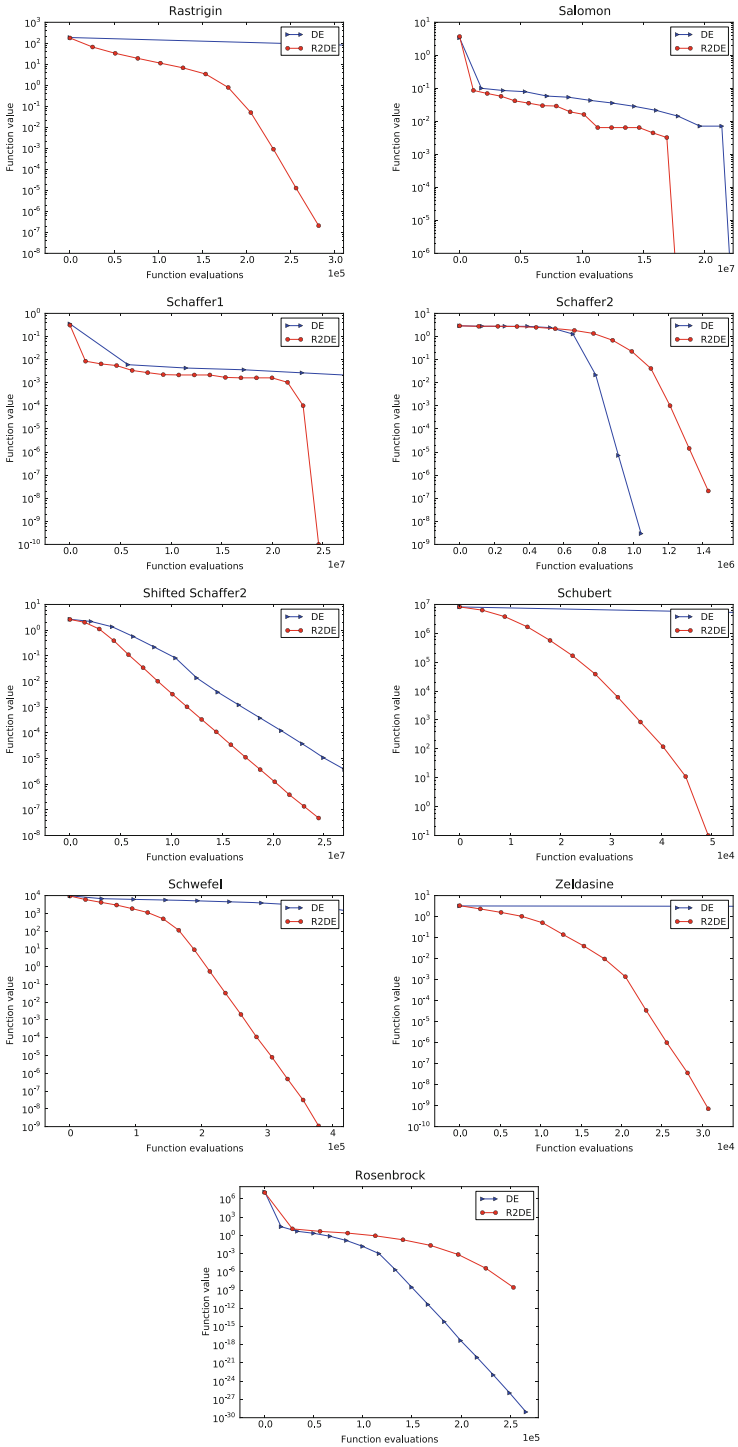

Fig. 8 Convergence plots of the cost functions at highest complexity settings

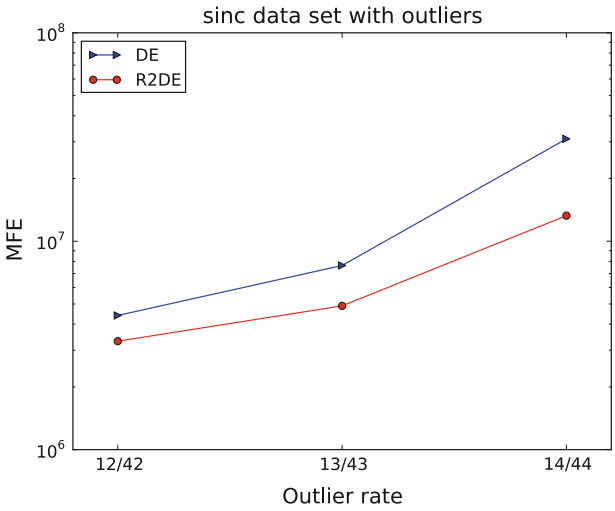

Fig. 9 MFE's to solve the robust estimation of ANN

Table 1 Comparison table of robust ANN estimation results on the **sinc** data set

Outliers total points	VTR	[Population size] MFE $\pm\sigma_{\text{MFE}}$		NOT rejected hypotheses by t-test
		DE	R2DE	
12/42	77.74	[450] 4403700 ± 1355800	[400] **3314540 ± 2219280**	H_{R2DE}
13/43	82.35	[700] 7648350 ± 2886830	[800] **4897040 ± 2793990**	H_{R2DE}
14/44	86.86	[4900] 30988800 ± 1290950	[2500] **13266400 ± 2262700**	H_{R2DE}

The t-test results (p-value = 0.01) are for the three hypotheses H_E : ($\text{MFE}_{DE} = \text{MFE}_{R2DE}$), H_{DE} : ($\text{MFE}_{DE} < \text{MFE}_{R2DE}$) and H_{R2DE} : ($\text{MFE}_{DE} > \text{MFE}_{R2DE}$) at robustness $\rho \approx 0.98$. Note that the bracketed numbers in the second, third and fourth columns denote the population sizes. The smallest MFE values for each setting are printed in *boldface*

sphere' property corresponds to functions which have the form $f(\mathbf{x}) = ||\mathbf{x}||^2 + \mu(\mathbf{x})$, where $\mu(\mathbf{x})$ is a multimodal function.

The classification of the results obtained by comparing the methods DE-λ and DE-α yields the following conclusions. On problems having multiple global optima or a rotation symmetry, DE-λ consistently proves to be the superior method. On the other hand, problems having the 'rough sphere' property or having regular local optima are best solved by the DE-α method. This fact underlines the motivation for the rank-based heuristic given in Sect. 3, since such functions comprise a local optima pattern where the global optimum can be reached by iteratively jumping from one basin to a better basin.

Comparing the methods DE, ODE, DESRF and R2DE yields the following conclusions. On unshifted rotation symmetric functions, other DE-variants outperform R2DE. Applying a shift to the Schaffer2 function (f_{15}) increases its complexity significantly, where R2DE again outperforms the other DE-variants. On functions with multiple global optima, R2DE outperforms the other methods in two of three cases. On functions having the 'rough sphere' property, R2DE consistently outperforms all other methods.

Table 2 Comparison table including the t-test results (p-value = 0.01) for the three hypotheses H_E : ($\text{MFE}_{DE} = \text{MFE}_{R2DE}$), H_{DE} : ($\text{MFE}_{DE} < \text{MFE}_{R2DE}$) and H_{R2DE} : ($\text{MFE}_{DE} > \text{MFE}_{R2DE}$) at robustness $\rho \approx 0.99$

Cost function	[Population size] MFE $\pm \sigma_{\text{MFE}}$				NOT rejected hypotheses by t-test
	DE	DERSF	ODE	R2DE	
f_1 [$D = 10$]	[30] 16089 ± 2542	[27] 13037 ± 1819	[220] 260159 ± 62749	[20] **8520 ± 1417**	H_{R2DE}
f_1 [$D = 30$]	[40] **49388 ± 4599**	[35] 38092 ± 2586	[280] 2476045 ± 908141	[30] 57987 ± 9991	H_{DE}
f_1 [$D = 50$]	[40] **69627 ± 5065**	[27] 69670 ± 6206	[480] 16913390 ± 7899190	[40] 136409 ± 17611	H_{DE}
f_2 [$D = 80$]	[200] **634256** ± 13480	[210] 679708 ± 11322	[860] 1104721 ± 73667	[900] 988875 ± 10483	H_{DE}
f_2 [$D = 100$]	[360] 1953780 ± 35149	[320] **1591844 ± 26728**	[1700] 2791100 ± 140068	[1300] 1693520 ± 15631	H_{R2DE}
f_2 [$D = 120$]	[430] 2961340 ± 48614	[440] 3057647 ± 50470	[4000] 6340334 ± 329518	[1700] **2409120 ± 18336**	H_{R2DE}
f_3 [$D = 8$]	[1300] 1678720 ± 155133	[1000] 1259391 ± 133929	[1000] **1237320 ± 167068**	[3600] 1435820 ± 67710	H_{R2DE}
f_3 [$D = 9$]	[2300] 5540380 ± 430929	[1800] 4428976 ± 355923	[1600] 3791740 ± 436196	[4600] **2300320 ± 105980**	H_{R2DE}
f_3 [$D = 10$]	[3600] 21254400 ± 1773940	[3000] 17932503 ± 1716194	[2800] 20591350 ± 3287375	[8000] **5761920 ± 353785**	H_{R2DE}
f_4 [$D = 5$]	[50] 5012 ± 271	[50] 4810 ± 218	[60] **4607 ± 393**	[70] 6889 ± 417	H_{DE}
f_4 [$D = 6$]	[70] 8375 ± 377	[60] **7440 ± 319**	[110] 9925 ± 689	[90] 10414 ± 588	H_{DE}
f_4 [$D = 7$]	[12000] 1810440 ± 55112	[11000] 1429787 ± 38618	[17000] 1950325 ± 247863	[7600] **1104510 ± 36670**	H_{R2DE}
f_5 [$D = 7$]	[140] 468772 ± 59409.9	[120] 357900 ± 58138	[230] 213946 ± 53154	[400] **174176 ± 18154**	H_{R2DE}
f_5 [$D = 8$]	[190] 1031690 ± 195713	[180] 885897 ± 199982	[300] 335013 ± 76555	[500] **230980 ± 23255**	H_{R2DE}
f_5 [$D = 9$]	[210] 1219020 ± 316313	[190] 935039 ± 258036	[460] 627003 ± 178729	[540] **197249 ± 33333**	H_{R2DE}
f_6 [$D = 11$]	[220] 1596920 ± 247696	[210] 1320702 ± 200149	[140] **54514 ± 16617**	[900] 468909 ± 34793	H_{R2DE}
f_6 [$D = 12$]	[280] 3018530 ± 575157	[290] 30221223 ± 545070	[160] **67493 ± 22380**	[1100] 649385 ± 40855	H_{R2DE}
f_6 [$D = 13$]	[360] 5700880 ± 1134270	[420] 74229564 ± 13381861	[160] **74274 ± 19250**	[1400] 933912 ± 64102	H_{R2DE}
f_7 [$D = 10$]	[370] 1042410 ± 116718	[370] 10903912 ± 122197	[700] 1929928 ± 418844	[720] **324763 ± 17185**	H_{R2DE}
f_7 [$D = 11$]	[440] 2377040 ± 292389	[430] 24424743 ± 287297	[760] 3890133 ± 1083132	[790] **426244 ± 22838**	H_{R2DE}
f_7 [$D = 12$]	[510] 5270890 ± 630430	[500] 54952587 ± 599286	[850] 8187608 ± 2552836	[940] **648882 ± 34567**	H_{R2DE}

Table 2 continued

Cost function	[Population size] MFE $\pm \sigma_{\text{MFE}}$				NOT rejected hypotheses by t-test
	DE	DERSF	ODE	R2DE	
f_8 [$D=2$]	[30] 1884 ± 369	[30] 2271 ± 318	[30] **1345 ± 361**	[30] 1872 ± 381	(All)
f_8 [$D=3$]	[50] 12173 ± 2169	[50] 11862 ± 2195	[70] **5089 ± 1389**	[200] 44224 ± 15101	H_{DE}
f_8 [$D=4$]	[60] 45151 ± 7552	[60] 46987 ± 9004	[100] **9370 ± 2304**	[400] 237884 ± 151059	H_{DE}
f_9 [β=6]	[450] 190814 ± 43699	[520] 210402 ± 46852	[620] 310954 ± 42458	[610] **159930 ± 32072**	H_{R2DE}
f_9 [β=5]	[800] 345192 ± 67983	[840] 347998 ± 79898	[1200] 633235 ± 82343	[720] **195538 ± 40033**	H_{R2DE}
f_9 [β=4]	[2100] 1007370 ± 229005	[2400] 10686137 ± 187884	[2300] 1316887 ± 201815	[1400] **394501 ± 87772**	H_{R2DE}
f_{10} [β=90]	[90] 25742 ± 4878	[110] 31149 ± 6418	[120] 46905 ± 8563	[30] **8714 ± 3822**	H_{R2DE}
f_{10} [β=80]	[100] 30132 ± 6011	[120] 30415 ± 5678	[130] 51092 ± 9690	[30] **9109 ± 4659**	H_{R2DE}
f_{10} [β=70]	[110] 33469 ± 7035	[110] 29740 ± 5618	[140] 55451 ± 9765	[30] **9300 ± 3974**	H_{R2DE}
f_{11} [$D=14$]	[200] 2225850 ± 602941	[220] 2822185 ± 601672	[650] 1050284 ± 359612	[350] **195531 ± 9376.89**	H_{R2DE}
f_{11} [$D=15$]	[220] 2790510 ± 524350	[280] 4400991 ± 985675	[550] 1166029 ± 403007	[380] **227305 ± 11667.7**	H_{R2DE}
f_{11} [$D=16$]	[240] 3787110 ± 825896	[300] 5265738 ± 1062972	[700] 1946828 ± 843550	[400] **253272 ± 12782.2**	H_{R2DE}
f_{12} [$D=3$]	[80] 48221 ± 11370	[70] 35385 ± 9182	[140] **23260 ± 4433**	[260] 77467 ± 31062	H_{DE}
f_{12} [$D=4$]	[260] 978086 ± 256098	[300] 1116525 ± 272336	[1000] **203268 ± 28343**	[900] 626805 ± 349025	H_{R2DE}
f_{12} [$D=5$]	[870] 18486100 ± 5249350	[1000] 21438168 ± 6414258	[1800] **482929 ± 73853**	[2200] 6219090 ± 5605270	H_{R2DE}
f_{13} [$D=2$]	[50] 7879 ± 1199	[50] 7151 ± 1094	[60] **5401 ± 899**	[240] 36348 ± 7843	H_{DE}
f_{13} [$D=3$]	[250] 541562 ± 128005	[200] 383208 ± 90219	[620] **94382 ± 24159**	[800] 501400 ± 243659	(All)
f_{13} [$D=4$]	[1400] 39717100 ± 9196540	[1400] 36234203 ± 9114016	[2400] **538106 ± 145054**	[3000] 9149130 ± 7372620	H_{R2DE}
f_{14} [$D=5$]	[310] 206640 ± 8490	[320] 214326 ± 9195	[240] **45380 ± 5900**	[460] 301130 ± 18151	H_{DE}
f_{14} [$D=6$]	[410] 437757 ± 20672	[400] 429438 ± 20578	[800] **249121 ± 28316**	[720] 696002 ± 46962	H_{DE}
f_{14} [$D=7$]	[510] 972680 ± 54007	[500] **859845 ± 53416**	[2600] 938896 ± 54693	[860] 1208090 ± 102368	H_{DE}

Table 2 continued

Cost function	[Population size] MFE $\pm$ σ_{MFE}				NOT rejected hypotheses by *t*-test
	DE	DERSF	ODE	R2DE	
f_{15} $[D=1]$	[30] 1650 $\pm$ 4190	[30] 1811 $\pm$ 170	[20] **848** $\pm$ **102**	[30] 1729 $\pm$ 214	H_{DE}
f_{15} $[D=2]$	[1300] 194181 $\pm$ 4190	[1300] 211311 $\pm$ 5110	[1700] 188490 $\pm$ 12954	[1000] **155410** $\pm$ **3919**	H_{R2DE}
f_{15} $[D=3]$	[13·10^4] 28674100 $\pm$ 580579	[13·10^4] 29842666 $\pm$ 612112	[21·10^4] 33682740 $\pm$ 1798810	[9·10^4] **20496600** $\pm$ **334727**	H_{R2DE}
f_{16} $[D=4]$	[40] 24724 $\pm$ 5307	[60] 46020 $\pm$ 8542	[120] 108539 $\pm$ 33160	[40] **10188** $\pm$ **1836**	H_{R2DE}
f_{16} $[D=5]$	[80] 126971 $\pm$ 23519	[80] 107899 $\pm$ 18959	[250] 648749 $\pm$ 275575	[50] **14717** $\pm$ **2584**	H_{R2DE}
f_{16} $[D=6]$	[120] 363317 $\pm$ 76073	[120] 327399 $\pm$ 69089	[260] 1031369 $\pm$ 348838	[70] **29494** $\pm$ **5866**	H_{R2DE}
f_{17} $[D=28]$	[170] 485841 $\pm$ 69941	[160] 442896 $\pm$ 67731	[160] 444909 $\pm$ 105765	[360] **288518** $\pm$ **14304**	H_{R2DE}
f_{17} $[D=29]$	[190] 626901 $\pm$ 113176	[170] 516598 $\pm$ 97429	[180] 579308 $\pm$ 115438	[370] **308051** $\pm$ **13207**	H_{R2DE}
f_{17} $[D=30]$	[190] 671090 $\pm$ 121122	[170] 561189 $\pm$ 110700	[180] 640701 $\pm$ 156928	[370] **315795** $\pm$ **12798**	H_{R2DE}
f_{18} $[D=9]$	[130] 1423160 $\pm$ 455462	[150] 1501999 $\pm$ 444180	[220] 927464 $\pm$ 305604	[40] **15318** $\pm$ **3253**	H_{R2DE}
f_{18} $[D=10]$	[150] 2329340 $\pm$ 985982	[160] 1970661 $\pm$ 637702	[250] 1352585 $\pm$ 448338	[40] **16656** $\pm$ **3140**	H_{R2DE}
f_{18} $[D=11]$	[160] 3244070 $\pm$ 132901	[180] 3334054 $\pm$ 1282642	[280] 1890824 $\pm$ 566861	[40] **18850** $\pm$ **3449**	H_{R2DE}
f_{19} $[D=6]$	[150] **53502** $\pm$ **9510**	[150] 55479 $\pm$ 8461	[900] 184770 $\pm$ 21690	[180] 60114 $\pm$ 11544	H_{DE}
f_{19} $[D=8]$	[150] 88579 $\pm$ 11734	[130] **85229** $\pm$ **16772**	[1200] 365508 $\pm$ 32025	[200] 119984 $\pm$ 16221	H_{DE}
f_{19} $[D=10]$	[130] **109379** $\pm$ **11199**	[120] 109645 $\pm$ 15410	[2000] 821200 $\pm$ 51276	[220] 196075 $\pm$ 25754	H_{DE}

Note that the bracketed numbers in the second, third and fourth columns denote the population sizes. The smallest MFE values for each problem and setting are printed in *boldface*

Table 3 Comparison table for DE-λ and DE-α including the t-test results (p-value = 0.01) for the three hypotheses H_E : ($\text{MFE}_{DE-\lambda} = \text{MFE}_{DE-\alpha}$), $H_{DE-\lambda}$: ($\text{MFE}_{DE-\lambda} < \text{MFE}_{DE-\alpha}$) and $H_{DE-\alpha}$: ($\text{MFE}_{DE-\lambda} > \text{MFE}_{DE-\alpha}$) at robustness $\rho \approx 0.99$

Cost function	[Population size] MFE $\pm\sigma_{\text{MFE}}$		NOT rejected hypotheses by t-test
	DE-λ	DE-α	
f_1 [$D=10$]	[20]**9026 ± 1161**	[40]15820 ± 3379	$H_{DE-\lambda}$
f_1 [$D=30$]	[40]**55697 ± 7011**	[70]100913 ± 24305	$H_{DE-\lambda}$
f_1 [$D=50$]	[100]**163693 ± 13032**	[90]200999 ± 50608	$H_{DE-\lambda}$
f_2 [$D=80$]	[1400]1821850 ± 18341	[220]**281142 ± 5030**	$H_{DE-\alpha}$
f_2 [$D=100$]	[1900]2927040 ± 29399	[260]**419853 ± 6786**	$H_{DE-\alpha}$
f_2 [$D=120$]	[3600]6550250 ± 55177	[300]**585834 ± 9542**	$H_{DE-\alpha}$
f_3 [$D=8$]	[3900]1701140 ± 101500	[1300]**1611690 ± 155089**	$H_{DE-\alpha}$
f_3 [$D=9$]	[7000]**4051600 ± 253748**	[2200]5123950 ± 413014	$H_{DE-\lambda}$
f_3 [$D=10$]	[1700]**14196300 ± 1062500**	[3200]18487800 ± 1822640	$H_{DE-\lambda}$
f_4 [$D=5$]	[70]6605 ± 424	[40]**5063 ± 260**	$H_{DE-\alpha}$
f_4 [$D=6$]	[100]11139 ± 664	[60]**7200 ± 364**	$H_{DE-\alpha}$
f_4 [$D=7$]	[11000]1513490 ± 49831	[9000]**1374300 ± 48004**	$H_{DE-\alpha}$
f_5 [$D=7$]	[500]236535 ± 23174	[180]**151358 ± 21451**	$H_{DE-\alpha}$
f_5 [$D=8$]	[840]475616 ± 46144	[200]**174374 ± 26320**	$H_{DE-\alpha}$
f_5 [$D=9$]	[1000]581400 ± 62510	[220]**179626 ± 29946**	$H_{DE-\alpha}$
f_6 [$D=11$]	[940]584642 ± 140394	[290]**371081 ± 52692**	$H_{DE-\alpha}$
f_6 [$D=12$]	[1400]966448 ± 72141	[360]**605239 ± 78378**	$H_{DE-\alpha}$
f_6 [$D=13$]	[1700]1292480 ± 103123	[420]**850395 ± 118731**	$H_{DE-\alpha}$
f_7 [$D=10$]	[1100]639584 ± 35803	[500]**517100 ± 57467**	$H_{DE-\alpha}$
f_7 [$D=11$]	[1600]**1140210 ± 73695**	[800]1243100 ± 112959	$H_{DE-\lambda}$
f_7 [$D=12$]	[2100]**1915220 ± 112176**	[900]2062950 ± 196482	$H_{DE-\lambda}$
f_8 [$D=2$]	[30]**1904 ± 386**	[60]5482 ± 1694	$H_{DE-\lambda}$
f_8 [$D=3$]	[130]**31279 ± 8820**	[140]67039 ± 20451	$H_{DE-\lambda}$
f_8 [$D=4$]	[440]**302971 ± 159374**	[250]457585 ± 142376	$H_{DE-\lambda}$
f_9 [β=6]	[580]**172991 ± 27167**	[660]233389 ± 84555	$H_{DE-\lambda}$
f_9 [β=5]	[800]**234160 ± 38003**	[1200]419892 ± 115837	$H_{DE-\lambda}$
f_9 [β=4]	[1300]**414440 ± 63574**	[2100]817719 ± 301176	$H_{DE-\lambda}$
f_{10} [β=90]	[20]**6385 ± 2569**	[140]33945 ± 5245	$H_{DE-\lambda}$
f_{10} [β=80]	[20]**6455 ± 2578**	[200]54842 ± 12469	$H_{DE-\lambda}$
f_{10} [β=70]	[20]**5705 ± 2698**	[210]55652 ± 11504	$H_{DE-\lambda}$
f_{11} [$D=14$]	[560]**395931 ± 23380**	[310]543982 ± 59938	$H_{DE-\lambda}$
f_{11} [$D=15$]	[610]**460288 ± 27274**	[310]600238 ± 67900	$H_{DE-\lambda}$
f_{11} [$D=16$]	[730]597542 ± 29252	[290]609412 ± 65534	(All)
f_{12} [$D=3$]	[210]**64682 ± 21257**	[270]108584 ± 28792	$H_{DE-\lambda}$
f_{12} [$D=4$]	[700]**462392 ± 258076**	[580]1041140 ± 303856	$H_{DE-\lambda}$
f_{12} [$D=5$]	[1500]**4668870 ± 373085**	[2200]12069200 ± 2568920	$H_{DE-\lambda}$
f_{13} [$D=2$]	[220]35853 ± 7063	[70]**10284 ± 2005**	$H_{DE-\alpha}$
f_{13} [$D=3$]	[700]**430472 ± 176140**	[220]482346 ± 119952	$H_{DE-\lambda}$
f_{13} [$D=4$]	[3200]**10132500 ± 8565000**	[3900]28461000 ± 6270310	$H_{DE-\lambda}$

Table 3 continued

Cost function	[Population size] MFE $\pm\sigma_{\text{MFE}}$		NOT rejected hypotheses by t-test
	DE-λ	DE-α	
f_{14} [$D=5$]	[550]**326002** ± **21842**	[470]304137 ± 23644	$H_{DE-\lambda}$
f_{14} [$D=6$]	[770]**698775** ± **41897**	[650]714941 ± 57742	$H_{DE-\lambda}$
f_{14} [$D=7$]	[920]**1246810** ± **103926**	[770]1357950 ± 136098	$H_{DE-\lambda}$
f_{15} [$D=1$]	[20]**1211** ± **107**	[30]1689 ± 148	$H_{DE-\lambda}$
f_{15} [$D=2$]	[1000]**112180** ± **3985**	[1800]182016 ± 6864	$H_{DE-\lambda}$
f_{15} [$D=3$]	[110000]**18323800** ± **421798**	[160000]23966400 ± 582319	$H_{DE-\lambda}$
f_{16} [$D=4$]	[40]**10487** ± **1977**	[40]25273 ± 5282	$H_{DE-\lambda}$
f_{16} [$D=5$]	[80]**34734** ± **6538**	[70]98028 ± 18977	$H_{DE-\lambda}$
f_{16} [$D=6$]	[100]**52674** ± **11799**	[110]341402 ± 73845	$H_{DE-\lambda}$
f_{17} [$D=28$]	[530]514195 ± 29507	[240]**301997** ± **30513**	$H_{DE-\alpha}$
f_{17} [$D=29$]	[540]540869 ± 30714	[370]**333285** ± **32820**	$H_{DE-\alpha}$
f_{17} [$D=30$]	[600]618888 ± 38051	[370]**379944** ± **36327**	$H_{DE-\alpha}$
f_{18} [$D=9$]	[40]**12724** ± **2649**	[140]1369540 ± 387731	$H_{DE-\lambda}$
f_{18} [$D=10$]	[40]**14311** ± **3100**	[180]2877480 ± 1057040	$H_{DE-\lambda}$
f_{18} [$D=11$]	[40]**16090** ± **3601**	[200]4667010 ± 2052570	$H_{DE-\lambda}$
f_{19} [$D=6$]	[160]**54771** ± **6825**	[550]188419 ± 48222	$H_{DE-\lambda}$
f_{19} [$D=8$]	[200]**111354** ± **9494**	[700]412461 ± 68292	$H_{DE-\lambda}$
f_{19} [$D=10$]	[220]**178330** ± **13249**	[750]620700 ± 72078	$H_{DE-\lambda}$

Note that the bracketed numbers in the second, third and fourth columns denote the population sizes. The smallest MFE values for each problem and setting are printed in *boldface*

Table 4 Comparison table for R2DE and R2DE with 'reversed' rank-based heuristic (R2DE-I) using the factor $1-\alpha(\boldsymbol{x})$ instead of $\alpha(\boldsymbol{x})$ at robustness $\rho \approx 0.99$

Cost function	[Population size] MFE $\pm\sigma_{\text{MFE}}$	
	R2DE	R2DE-I
f_1 [$D=10$]	[20] **8520** ± **1417**	[30] 16639 ± 3225
f_2 [$D=20$]	[160] **61009** ± **1467**	[700] 185934 ± 4197
f_3 [$D=5$]	[400] **52660** ± **2815**	[1600] 114784 ± 9846
f_4 [$D=5$]	[70] **6889** ± **417**	[120] 9912 ± 781
f_5 [$D=6$]	[400] **159048** ± **14008**	[8500] 1139170 ± 75559
f_6 [$D=7$]	[250] **66040** ± **6181**	[3500] 411950 ± 30223
f_7 [$D=5$]	[400] **52660** ± **2815**	[1600] 114784 ± 9846
f_8 [$D=2$]	[30] **1872** ± **381**	[250] 7235 ± 1187
f_9 [$D=4,\beta=13$]	[240] **59844** ± **13613**	[800] 131280 ± 46875
f_{10} [$D=4,\beta=100$]	[30] **8802** ± **3822**	[40] 47278 ± 34781
f_{11} [$D=9$]	[180] **63451** ± **4352**	[1200] 194256 ± 8052

Note that the bracketed numbers in the second and third columns denote the population sizes. The smallest MFE values are printed in *boldface*

Table 5 Comparison table for NSDE and SAR2DE including the t-test results (p-value = 0.01) for the three hypotheses H_E : ($\text{MFE}_{NSDE} = \text{MFE}_{SAR2DE}$), H_{NSDE} : ($\text{MFE}_{NSDE} < \text{MFE}_{SAR2DE}$) and H_{SAR2DE} : ($\text{MFE}_{NSDE} > \text{MFE}_{SAR2DE}$) at robustness $\rho \approx 0.99$

Cost function	[Population size] MFE $\pm \sigma_{\text{MFE}}$		NOT rejected hypotheses by t-test
	NSDE	SAR2DE	
f_1 [$D=10$]	[10] 4621 ± 514	[10] **4111 ± 730**	H_{SAR2DE}
f_1 [$D=30$]	[10] **14357 ± 1615**	[15] 17460 ± 1994	H_{NSDE}
f_1 [$D=50$]	[10] **31270 ± 4879**	[20] 34212 ± 3496	H_{NSDE}
f_2 [$D=80$]	[70] 140410 ± 2323	[110] **114005 ± 1363**	H_{SAR2DE}
f_2 [$D=100$]	[80] 196713 ± 2583	[130] **159025 ± 1735**	H_{SAR2DE}
f_2 [$D=120$]	[80] 227508 ± 3608	[140] **195912 ± 1777**	H_{SAR2DE}
f_3 [$D=8$]	[180]**188129 ± 15661**	[320] 197338 ± 19497	H_{NSDE}
f_3 [$D=9$]	[210] 250507 ± 20457	[340]**236912 ± 17929**	H_{SAR2DE}
f_3 [$D=10$]	[240] 529099 ± 33953	[370]**467166 ± 35140**	H_{SAR2DE}
f_4 [$D=5$]	[30] **4373 ± 539**	[40] 4688 ± 648	H_{NSDE}
f_4 [$D=6$]	[60] **11103 ± 845**	[120] 16890 ± 1367	H_{NSDE}
f_4 [$D=7$]	[9000] 2702430 ± 71217	[10000] **2450100 ± 125543**	H_{SAR2DE}
f_5 [$D=7$]	[60] 43035 ± 4196	[90] **37323 ± 3796**	H_{SAR2DE}
f_5 [$D=8$]	[60] 44751 ± 4214	[90] **37650 ± 4063**	H_{SAR2DE}
f_5 [$D=9$]	[60] 45402 ± 4898	[90] **37204 ± 4142**	H_{SAR2DE}
f_6 [$D=11$]	[70] 144500 ± 18875	[120] **128069 ± 14648**	H_{SAR2DE}
f_6 [$D=12$]	[80] 193044 ± 22938	[150] **183614 ± 19198**	H_{SAR2DE}
f_6 [$D=13$]	[100] 278254 ± 33453	[180] **245677 ± 22993**	H_{SAR2DE}
f_7 [$D=10$]	[110] 42417 ± 1683	[150] **38607 ± 1312**	H_{SAR2DE}
f_7 [$D=11$]	[110] 46780 ± 1877	[160] **44689 ± 1787**	H_{SAR2DE}
f_7 [$D=12$]	[110] 55139 ± 2142	[160] **52384 ± 2014**	H_{SAR2DE}
f_8 [$D=2$]	[30] **3043 ± 827**	[60] 5067 ± 1424	H_{NSDE}
f_8 [$D=3$]	[70] **26949 ± 7018**	[120] 31161 ± 9331	H_{NSDE}
f_8 [$D=4$]	[180] **154681 ± 46398**	[280] 183140 ± 61538	H_{NSDE}
f_9 [β=6]	[240] 199277 ± 25671	[380] **186436 ± 31307**	H_{SAR2DE}
f_9 [β=5]	[400] 360540 ± 49711	[590] **291690 ± 42360**	H_{SAR2DE}
f_9 [β=4]	[700] 705159 ± 100991	[800] **436704 ± 64915**	H_{SAR2DE}
f_{10} [β=90]	[10] **8393 ± 3637**	[20] 10779 ± 4547	H_{NSDE}
f_{10} [β=80]	[10] 9185 ± 3745	[20] 10558 ± 4617	(All)
f_{10} [β=70]	[10] 9331 ± 3774	[20] 10616 ± 4314	(All)
f_{11} [$D=14$]	[50] 31728 ± 1312	[80] 31788 ± 1302	(All)
f_{11} [$D=15$]	[50] 34131 ± 1486	[70] **29680 ± 1295**	H_{SAR2DE}
f_{11} [$D=16$]	[50] 36630 ± 1405	[80] 36404 ± 1385	(All)
f_{12} [$D=3$]	[80] **50284 ± 15371**	[150] 59931 ± 26129	H_{NSDE}
f_{12} [$D=4$]	[200] 383558 ± 259494	[280] 323540 ± 278935	(All)
f_{12} [$D=5$]	[600] 2790140 ± 2244530	[1100] 2494510 ± 2309880	(All)
f_{13} [$D=2$]	[100] 27108 ± 5816	[140] 27388 ± 7002	(All)
f_{13} [$D=3$]	[300] 380097 ± 158502	[350] **305490 ± 201493**	H_{SAR2DE}
f_{13} [$D=4$]	[1300] 4259400 ± 2164820	[2000] 3681140 ± 2430140	(All)

Table 5 continued

Cost function	[Population size] MFE $\pm \sigma_{\text{MFE}}$		NOT rejected hypotheses by t-test
	NSDE	SAR2DE	
f_{14} $[D=5]$	[450] **719514** $\pm$ **94907**	[700] 1018090 $\pm$ 169516	H_{NSDE}
f_{14} $[D=6]$	[900] **3059110** $\pm$ **429092**	[1200] 3986230 $\pm$ 1108980	H_{NSDE}
f_{14} $[D=7]$	[1200] **7903030** $\pm$ **1302780**	[1500] 10990800 $\pm$ 10080900	H_{NSDE}
f_{15} $[D=1]$	[30] 2583 $\pm$ 165	[30] **2298** $\pm$ **166**	H_{SAR2DE}
f_{15} $[D=2]$	[2800] 521668 $\pm$ 14428	[2400] **376656** $\pm$ **12265**	H_{SAR2DE}
f_{15} $[D=3]$	[12000] 22996300 $\pm$ 31749400	[35000] 21623600 $\pm$ 13975300	(All)
f_{16} $[D=4]$	[30] 18023 $\pm$ 4109	[15] **3673** $\pm$ **753**	H_{SAR2DE}
f_{16} $[D=5]$	[30] 22335 $\pm$ 3979	[20] **6560** $\pm$ **1441**	H_{SAR2DE}
f_{16} $[D=6]$	[30] 24949 $\pm$ 3650	[20] **7306** $\pm$ **1488**	H_{SAR2DE}
f_{17} $[D=28]$	[50] 50514 $\pm$ 1284	[70] **43220** $\pm$ **1001**	H_{SAR2DE}
f_{17} $[D=29]$	[50] 52078 $\pm$ 1273	[70] **44447** $\pm$ **991**	H_{SAR2DE}
f_{17} $[D=30]$	[55] 59704 $\pm$ 1377	[80] **52485** $\pm$ **1101**	H_{SAR2DE}
f_{18} $[D=9]$	[8] 6171 $\pm$ 2275	[11] **5370** $\pm$ **1154**	H_{SAR2DE}
f_{18} $[D=10]$	[9] 8591 $\pm$ 2707	[11] **6217** $\pm$ **1291**	H_{SAR2DE}
f_{18} $[D=11]$	[9] 9182 $\pm$ 3144	[12] **7498** $\pm$ **1627**	H_{SAR2DE}
f_{19} $[D=6]$	[60] 41481 $\pm$ 5316	[60] **34530** $\pm$ **7641**	H_{SAR2DE}
f_{19} $[D=8]$	[60] 69522 $\pm$ 9020	[80] 72263 $\pm$ 17367	(All)
f_{19} $[D=10]$	[60] **99340** $\pm$ **13948**	[100] 130137 $\pm$ 24795	H_{NSDE}

Note that the bracketed numbers in the second, third and fourth columns denote the population sizes. The smallest MFE values for each problem and setting are printed in *boldface*

On the other hand, DE outperforms R2DE on Alpine (f_1), Periodic (f_8), Schaffer2 (f_{14}) and Rosenbrock (f_{19}) functions. The Alpine function approximately satisfies the condition of regularly distributed local optima (3), (4), whereas the Periodic function (almost) exactly satisfies it. These results support the regularity condition assumptions.

One test function where R2DE yields particularly good results is Zeldasine (f_{18}), which comprises several global optima exactly satisfying the regularity condition. Due to the fixed mutation scale factor $F = 0.5$, DE explores several modes (global optima) and is not able to quickly switch to 'local convergence', i.e., the average difference vectors remain large for a long period of iterations. In contrast, R2DE is able to quickly 'pick' a mode and switch to local convergence due to its stochastic mutation scale. This behavior can also be verified from the convergence plot of Zeldasine in Fig. 8.

On functions which do not fit in one of the mentioned categories, R2DE outperforms all other DE-variants in 6 out of 7 cases. This observation supports the assumption that a stochastic mutation scale factor in DE's update formula can lead to increased efficiency of global convergence.

The classification of the results obtained from the self-adaptive methods NSDE and SAR2DE yields similar conclusions. SAR2DE is to be preferred on problems comprising a 'rough sphere' property, on problems having multiple global optima and on problems having the property 'regular local optima'. Also, on problems which fall into the 'others' category, SAR2DE clearly performs better than NSDE.

However, the underlying principles to exactly explain the results are rather complex. Unfortunately, there is no algebraic analysis available of DE's global search behavior which

Table 6 Assignment of all test functions to one or more attributes

Rotation symmetry	Multiple global optima	Rough sphere	Regular local optima	Others
f_{12} (DE-λ)	f_1 (DE-λ)	f_2 (DE-α)	f_1 (DE-λ)	f_3 (DE-λ)
f_{13} (DE-λ)	f_{16} (DE-λ)	f_5 (DE-α)	f_2 (DE-α)	f_4 (DE-λ,DE-α)
f_{14} (DE-λ)	f_{18} (DE-λ)	f_{11} (DE-λ,DE-α)	f_5 (DE-α)	f_6 (DE-α)
f_{15} (DE-λ)			f_8 (DE-α)	f_7 (DE-λ,DE-α)
			f_{11} (DE-λ,DE-α)	f_9 (DE-λ)
			f_{18} (DE-λ)	f_{10} (DE-λ)
				f_{17} (DE-α)
				f_{19} (DE-λ)
f_{12} (ODE)	f_1 (DE)	f_2 (R2DE)	f_1 (DE)	f_3 (R2DE)
f_{13} (ODE)	f_{16} (R2DE)	f_5 (R2DE)	f_2 (R2DE)	f_4 (R2DE)
f_{14} (DERSF)	f_{18} (R2DE)	f_{11} (R2DE)	f_5 (R2DE)	f_6 (ODE)
f_{15} (R2DE)			f_8 (ODE)	f_7 (R2DE)
			f_{11} (R2DE)	f_9 (R2DE)
			f_{18} (R2DE)	f_{10} (R2DE)
				f_{17} (R2DE)
				f_{19} (DE)
f_{12} (NSDE,SAR2DE)	f_1 (NSDE)	f_2 (SAR2DE)	f_1 (NSDE)	f_3 (SAR2DE)
f_{13} (NSDE,SAR2DE)	f_{16} (SAR2DE)	f_5 (SAR2DE)	f_2 (SAR2DE)	f_4 (SAR2DE)
f_{14} (NSDE)	f_{18} (SAR2DE)	f_{11} (NSDE,SAR2DE)	f_5 (SAR2DE)	f_6 (SAR2DE)
f_{15} (NSDE,SAR2DE)			f_8 (NSDE)	f_7 (SAR2DE)
			f_{11} (NSDE, SAR2DE)	f_9 (SAR2DE)
			f_{18} (SAR2DE)	f_{10} (NSDE, SAR2DE)
				f_{17} (SAR2DE)
				f_{19} (NSDE)

Each function is marked with the method which performs best on the function at its highest considered complexity setting. In the upper part, DE-λ and DE-α are compared. In the middle, DESRF, ODE, DE and R2DE are compared. In the lower part, only NSDE and SAR2DE are compared

would help to interpret the results analytically. Empirically, the overall results indicate that randomization and the utilization of the rank information seems to generally improve DE's performance on multimodal problems.

5.4 Sensitivity to parameters F and C_r

Depending on the cost function, R2DE can be sensitive to its parameters F and C_r. Figure 10 shows some examples of the dependency of R2DE's performance on these parameters by measuring the MFE at a robustness of $\rho \approx 0.99$. As in previous experiments, for each setting, we manually adapt the optimal population size to reach the robustness condition and to minimize the MFE number at the same time. On Rastrigin and Zeldasine, there is a strong sensitivity on C_r, where small values for C_r tend to improve the convergence speed, although it is generally advised to set $C_r = 0.9$ [1,5,15,26,35,42]. This is because both functions are

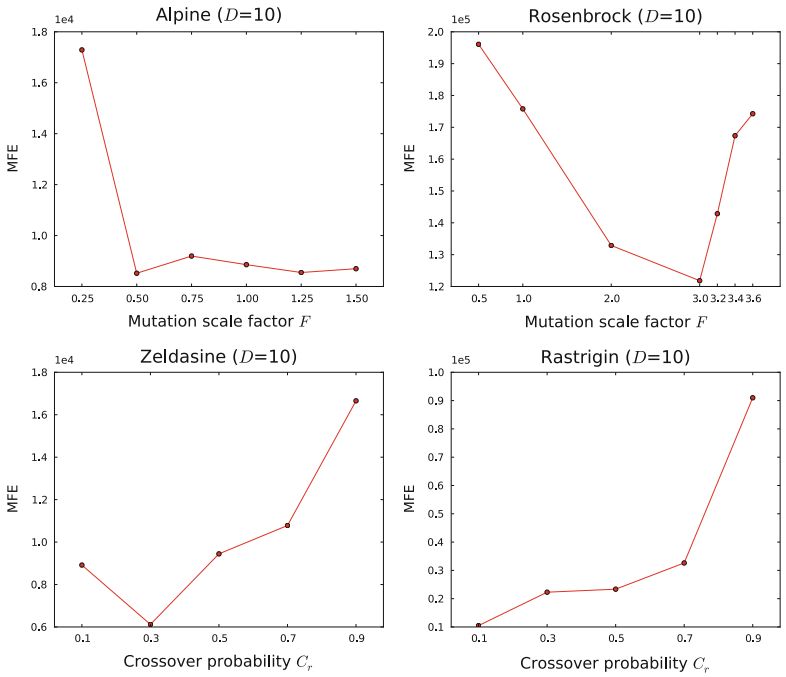

Fig. 10 Required mean function evaluations (MFE's) to find the global optimum with a robustness of $\rho \approx 0.99$ on different settings of the parameters F and C_r

separable, but this property is a special case and in general functions are not expected to be separable.

We assume that the sensitivity of R2DE on C_r is comparable to DE, since the application of the crossover operator is identical in both methods.

The dependence on F can be different compared to DE, as shown in the case of the Rosenbrock cost function, where an optimum is found for $F \approx 3$. In contrast, values of $F > 1$ do generally not yield better results in DE. The cost function Alpine represents a case where values $F \in [0.5, 1.5]$ do not significantly influence the MFE.

6 Conclusions

A novel Evolutionary Algorithm, Randomized and Rank-based Differential Evolution (R2DE) and a self-adaptive version, SAR2DE, are presented as a modification to the well known Differential Evolution (DE) method. The application domain of R2DE contains highly complex, multimodal functions. In the presented experiments, R2DE is compared to DE, DE with Random Scale Factor (DERSF) and Opposition Based Differential Evolution (ODE) techniques, respectively. Each problem is evaluated at several complexity settings, such as the dimension, to determine the tendency of global search efficiency for each method.

Regarding the required mean function evaluations (MFE's), the empirical results indicate that R2DE outperforms DE and DERSF in 15 and ODE in 13 out of 19 bench-

mark problems. On problems with exactly-regular distribution of local optima with a unique global optimum or unshifted rotation symmetric problems, other DE-variants outperform R2DE. On the other hand, R2DE is superior on problems with a large number of global optima, problems with approximately-regular distribution of local optima, rough sphere type of problems or problems having a more complex pattern of local optima distributions.

According to the presented experimental results, R2DE generally requires a greater population of individuals to achieve the same robustness. On the other hand, it requires a much smaller number of iterations for global convergence, and outperforms DE, DERSF and ODE on the majority of common global optimization problems. Furthermore, the MFE-differences increase with the complexity of the problem.

The self-adaptive version of R2DE (SAR2DE) is compared to NSDE, since both methods use Cauchy distributed scale factors. NSDE outperforms SAR2DE on 4 out of 19 test functions, whereas SAR2DE outperforms NSDE on 10 functions. On the remaining 5 functions, there was no statistically significant difference.

Experiments on robust estimation of artificial neural network (ANN) based problems show that R2DE clearly outperforms DE. Furthermore, the performance improvement of R2DE over DE increases with increasing number of outliers.

Generally, the stochastic mutation scale factor can improve global convergence in the majority of the cost functions considered in this paper. Furthermore, according to presented experiments, R2DE yields particularly good results on functions having a large number of global optima, such as the Zeldasine problem and the robust estimation of ANN.

Acknowledgments We would like to thank all the reviewers, especially Reviewer 3 for the valuable comments and inspirations. Thanks to Reviewer 3, this paper was enriched by a self-adaptive version of R2DE (SAR2DE), and the rank-based heuristic could be explained and motivated with much more clarity.

Appendix

A parameters of the foxholes function

In the Foxholes cost function (f_4), c_k and a_{jk} are defined by:

$$c = \begin{pmatrix} 8.147 \\ 1.355 \\ 9.058 \\ 8.35 \\ 1.27 \\ 9.689 \\ 9.134 \\ 2.21 \\ 6.324 \\ 3.082 \\ 0.9754 \\ 5.472 \\ 2.785 \\ 1.884 \\ 5.469 \\ 9.929 \\ 9.575 \\ 9.965 \\ 9.649 \\ 9.677 \\ 1.576 \\ 7.258 \\ 9.706 \\ 9.811 \\ 9.572 \\ 1.099 \\ 4.854 \\ 7.981 \\ 8.003 \\ 2.97 \\ 1.419 \\ 0.04783 \\ 4.218 \\ 1.125 \\ 9.157 \\ 6.398 \\ 7.922 \\ 8.784 \\ 9.595 \\ 5.037 \\ 6.557 \\ 7.979 \\ 0.3571 \\ 3.613 \\ 8.491 \\ 2.119 \\ 9.34 \\ 6.814 \\ 6.787 \\ 3.987 \end{pmatrix}, \quad a = \begin{pmatrix} 7.577 & 7.406 & 7.431 & 4.748 & 3.922 \\ 4.221 & 6.555 & 1.739 & 1.712 & 3.019 \\ 7.06 & 7.973 & 0.3183 & 3.166 & 2.769 \\ 8.724 & 0.4617 & 1.491 & 0.9713 & 9.941 \\ 8.235 & 8.219 & 6.948 & 1.252 & 3.171 \\ 7.638 & 9.502 & 4.906 & 0.3445 & 6.636 \\ 4.387 & 1.259 & 3.816 & 2.102 & 7.655 \\ 0.5122 & 7.952 & 0.3644 & 1.869 & 4.087 \\ 4.898 & 4.58 & 4.456 & 4.876 & 6.463 \\ 7.94 & 7.094 & 9.209 & 7.547 & 8.075 \\ 2.76 & 7.058 & 6.797 & 0.02818 & 6.551 \\ 7.107 & 1.626 & 6.44 & 1.19 & 4.56 \\ 4.984 & 7.739 & 9.597 & 5.738 & 3.404 \\ 8.768 & 5.853 & 8.082 & 2.238 & 0.1777 \\ 7.513 & 8.212 & 2.551 & 8.208 & 5.06 \\ 9.401 & 6.991 & 4.127 & 8.909 & 4.232 \\ 9.593 & 5.81 & 5.472 & 1.581 & 1.386 \\ 7.617 & 1.493 & 2.302 & 2.575 & 8.097 \\ 8.407 & 9.885 & 2.543 & 3.324 & 8.143 \\ 2.998 & 2.435 & 0.1354 & 9.293 & 2.172 \\ 3.5 & 9.074 & 1.966 & 8.485 & 2.511 \\ 9.55 & 6.16 & 7.789 & 4.733 & 9.875 \\ 3.517 & 0.676 & 8.308 & 7.936 & 5.853 \\ 5.945 & 5.497 & 7.328 & 9.172 & 6.952 \\ 2.858 & 6.798 & 7.572 & 3.923 & 7.537 \\ 5.616 & 3.804 & 2.081 & 5.678 & 5.274 \\ 0.7585 & 4.042 & 0.5395 & 3.528 & 5.308 \\ 5.928 & 7.792 & 3.563 & 9.34 & 9.65 \\ 1.299 & 1.544 & 5.688 & 3.949 & 4.694 \\ 3.873 & 0.119 & 7.27 & 3.371 & 3.886 \\ 1.622 & 9.275 & 7.943 & 4.361 & 3.112 \\ 8.627 & 5.285 & 6.204 & 1.656 & 1.195 \\ 6.02 & 4.72 & 2.63 & 3.402 & 6.541 \\ 5.298 & 6.892 & 7.161 & 7.482 & 9.884 \\ 4.505 & 7.205 & 0.8382 & 9.126 & 2.29 \\ 5.055 & 9.133 & 5.583 & 1.524 & 5.032 \\ 8.258 & 4.625 & 5.383 & 5.466 & 9.961 \\ 4.476 & 0.7818 & 8.545 & 4.427 & 6.042 \\ 1.067 & 4.985 & 9.619 & 9.799 & 0.04634 \\ 0.3432 & 7.749 & 9.77 & 8.173 & 3.632 \\ 8.687 & 6.795 & 0.8444 & 3.462 & 3.998 \\ 8.559 & 2.599 & 0.4506 & 8.001 & 6.601 \\ 4.314 & 7.499 & 9.106 & 1.33 & 1.818 \\ 9.824 & 2.638 & 0.9536 & 1.455 & 2.827 \\ 1.361 & 8.021 & 8.693 & 0.7756 & 5.797 \\ 6.274 & 5.499 & 0.08094 & 1.45 & 6.803 \\ 8.53 & 5.339 & 6.221 & 4.387 & 3.51 \\ 1.996 & 5.132 & 1.38 & 4.018 & 3.823 \\ 0.7597 & 7.624 & 2.399 & 0.4047 & 1.233 \\ 2.53 & 1.839 & 5.048 & 2.4 & 8.226 \end{pmatrix}$$

References

1. Ali, M.M., Törn, A.: Population set-based global optimization algorithms: some modifications and numerical studies. Comput. Oper. Res. **31**(10), 1703–1725 (2004)
2. Ali, M.M., Khompatraporn, C., Zabinsky, Z.B.: A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. J. Global Optim. **31**(4), 635–672 (2005)
3. Bersini, H., Dorigo, M., Langerman, S., Seront, G., Gambardella, L.M.: Results of the first international contest on evolutionary optimisation (1st iceo). In: International Conference on Evolutionary Computation, pp. 611–615 (1996)
4. Breiman, L., Cutler, A.: A deterministic algorithm for global optimization. Math. Program. **58**(2), 179–199 (1993)
5. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. Evol. Comput. IEEE Transact. **10**(6), 646–657 (2006)
6. Das, S., Konar, A., Chakraborty, U.k.: Improved differential evolution algorithms for handling noisy optimization problems, vol 2, pp. 1691–1698 (2005a)
7. Das, S., Konar, A., Chakraborty, U.K.: Two improved differential evolution schemes for faster global search. In: GECCO '05: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 991–998, New York, NY, USA, ACM (2005b)
8. Das, S., Abraham, A., Chakraborty, U.K., Konar, A.: Differential evolution using a neighborhood-based mutation operator. IEEE Transact. Evol. Comput. **13**(3), 526–553 (2009)
9. Fan, H.-Y., Lampinen, J.: A trigonometric mutation operation to differential evolution. J. Global Optim. **27**(1), 105–129 (2003)
10. Haflidason, S., Neville, R.: On the significance of the permutation problem in neuroevolution. In: GECCO '09: Proceedings of the 11th Annual conference on Genetic and Evolutionary Computation, pp. 787–794, New York, NY, USA, ACM (2009)
11. Hansen, N., Ros, R., Mauny, N., Schoenauer, M., Auger, A.: Pso facing non-separable and ill-conditioned problems. Research Report 6447, INRIA, February 2008. http://hal.inria.fr/inria-00250078/en
12. Kaelo, P., Ali, M.M.: Probabilistic adaptation of point generation schemes in some global optimization algorithms. J. Optim. Methods Softw. **21**(3), 343–357 (2006)
13. Koumousis, V.K., Katsaras, C.P.: A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. IEEE Trans. Evol. Comput. **10**(1), 19–28 (2006)
14. Levy, A.V., Montalvo, A.: The tunneling algorithm for the global minimization of functions. SIAM J. Sci. Stat. Comput. **6**(1), 15–29 (1985)
15. Liu, J., Lampinen, J.: A fuzzy adaptive differential evolution algorithm. Soft Comput. **9**(6), 448–462 (2005)
16. Locatelli, M.: A note on the griewank test function. J. Global Optim. **25**(2), 169–174 (2003)
17. Michalewicz, Z.: Genetic algorithms + data structures = evolution programs (2nd, extended ed.). Springer-Verlag New York, Inc., New York, NY, USA (1994)
18. Moré, J.J., Garbow, B.S., Hillstrom, K.E.: Testing unconstrained optimization software. ACM Trans. Math. Softw. **7**(1), 17–41 (1981)
19. Noman, N., Iba, H.: Enhancing differential evolution performance with local search for high dimensional function optimization. In: GECCO '05: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 967–974, New York, NY, USA, ACM (2005)
20. Omran, M., Salman, A., Engelbrecht, A.: Self-adaptive differential evolution. In Hao, Y., Liu, J., Wang, Y.-P., Cheung, Y.-M., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (ed.) Computational Intelligence and Security, vol. 3801 of Lecture Notes in Computer Science, pp. 192–199. Springer Berlin /Heidelberg (2005)
21. Onwubolu, G.C., Babu, B.: New Optimization Techniques in Engineering. Springer, ISBN-13: 978-3540201670 (2004)
22. Price, K.V.: Differential evolution: a fast and simple numerical optimizer. In: Biennial Conference of the North American Fuzzy Information Processing Society, NAFIPS, pp. 524–527. IEEE Press, New York. ISBN:0-7803-3225-3, June (1996)
23. Price, W.L.: A controlled random search procedure for global optimisation (1977)
24. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution A Practical Approach to Global Optimization. Natural Computing Series. Springer-Verlag, Berlin, Germany (2005)
25. Qin, A., Huang, V., Suganthan, P.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput. **13**(2), 398–417 (2009)
26. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A.: Opposition-based differential evolution. IEEE Trans. Evol. Comput. **12**(1), 64–79 (2008)

27. Ronkkonen, J., Lampinen, J.: On Using Normally Distributed Mutation Step Length for the Differential Evolution Algorithm, pp. 11–18. Brno, Czech Republic, June (2003)
28. Salomon, R.: Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms. Biosystems **39**(3), 263–278 (1996)
29. Shang, Y.-W., Qiu, Y.-H.: A note on the extended rosenbrock function. Evol. Comput. **14**, 119–126 (2006)
30. Shi, Y.-J., Teng, H.-F., Li, Z.-Q.: Cooperative co-evolutionary differential evolution for function optimization. In: Proceedings 1st International Conference Advances in Natural Computer, pp. 1080–1088. Changsha, China, Aug. (2005)
31. Soliman, O., Bui, L.: A Self-adaptive Strategy for Controlling Parameters in Differential Evolution, pp. 2837–2842. June (2008)
32. Soliman, O., Bui, L., Abbass, H.: The Effect of a Stochastic Step Length on the Performance of the Differential Evolution Algorithm. pp. 2850–2857, Sep. (2007)
33. Storn, R., Price, K.: Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, ICSI, March (1995)
34. Storn, R., Price, K.: Minimizing the real functions of the icec'96 contest by differential evolution. In: IEEE International Conference on Evolutionary Computation, pp. 842–844. Nagoya, May (1996)
35. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**(4), 341–359 (1997)
36. Sun, J., Zhang, Q., Tsang, E.P.: De/eda: a new evolutionary algorithm for global optimization. Inform. Sci. **169**(3–4), 249–262 (2005)
37. Tasoulis, D.K., Pavlidis, N.G., Plagianakos, V.P., Vrahatis, M.N.: Parallel differential evolution
38. Teo, J.: Exploring dynamic self-adaptive populations in differential evolution. Soft Comput. **10**(8), 673–686 (2006)
39. Torn, A., Zilinskas, A.: Global Optimization. Springer-Verlag New York, Inc., New York, NY, USA (1989)
40. Urfalioglu, O.: Robust estimation of camera rotation, translation and focal length at high outlier rates. In: Proceedings International Canadian Conference on Computer and Robot Vision, pp. 464–471. May (2004)
41. Urfalioglu, O., Mikulastik, P., Stegmann, I.: Scale invariant robust registration of 3d-point data and a triangle mesh by global optimization. In: Proceedings of Advanced Concepts for Intelligent Vision Systems, (2006)
42. Vesterstrom, J., Thomsen, R.: A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: Evolutionary Computation, 2004. CEC2004. Congress on, vol. 2, pp. 1980–1987. June (2004)
43. Weissenfeld, A., Urfalioglu, O., Liu, K., Ostermann, J.: Robust rigid head motion estimation based on differential evolution. In: Proceedings of International Conference on Multimedia and Expo (2006)
44. Yang, Z., Tang, K., Yao, X.: Self-adaptive Differential Evolution with Neighborhood Sarch, pp. 1110–1116, June (2008)
45. Yang, Z., Yao, X., He, J. : Making a difference to differential evolution. In: Siarry, P., Michalewicz, Z. (eds.) Advances in Metaheuristics for Hard Optimization, Natural Computing Series, pp. 397–414. Springer, Berlin Heidelberg (2008)
46. Zabinsky, Z.B., Graesser, D.L., Tuttle, M.E., Kim, G.-I.: Global Optimization of Composite Laminates using Improving Hit and Run, pp. 343–368 (1992)
47. Zaharie, D.: Critical Values for the Control Parameters of Differential Evolution Algorithmss, pp. 62–67, Brno, Czech Republic (2002)
48. Zhenyu, G., Bo, C., Min, Y., Binggang, C.: Self-adaptive chaos differential evolution. In: Jiao, L., Wang, L., Gao, X., Liu, J., Wu, F., (eds.) Advances in Natural Computation, vol. 4221 of Lecture Notes in Computer Science, pp. 972–975. Springer Berlin/Heidelberg (2006)