Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 169 (2006) 1176–1184
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

A numerical study of some modified differential evolution algorithms

P. Kaelo, M.M. Ali*

School of Computational and Applied Mathematics, Witwatersrand University, Private-Bag-3, Wits-2050, Johannesburg, South Africa

Received 1 September 2003; accepted 1 August 2004
Available online 17 May 2005

Abstract

Modifications in mutation and localization in acceptance rule are suggested to the differential evolution algorithm for global optimization. Numerical experiments indicate that the resulting algorithms are considerably better than the original differential evolution algorithm. Therefore, they offer a reasonable alternative to many currently available stochastic algorithms, especially for problems requiring _direct search type_ methods. Numerical study is carried out using a set of 50 test problems many of which are inspired by practical applications.

Keywords: Global optimization; Population set based method; Modified differential evolution; Continuous variable

1. Introduction

The global optimization problem in this paper follows the form:

minimize $f(x)$ subject to $x \in \Omega$,

where x is a continuous variable vector with domain $\Omega \subset \mathbb{R}^n$, and $f(x): \Omega \mapsto \mathbb{R}$ is a continuous real-valued function. The domain Ω is defined by specifying upper (u^j) and lower (l^j) limits of each component j. We denote the global optimal solution by x^*, with its corresponding global optimal function value $f(x^*)$ or f^* for a short hand notation. Differential evolution (DE) [1] is a population set based global optimization algorithm [2] and purely heuristic. All population set based direct search methods use a population set S. The initial set

$$S = \{x_1, x_2, \ldots, x_N\}$$

consists of N random points in Ω. A contraction process is then used to drive these points to the vicinity of the global minimizer. The contraction process involves replacing bad point(s) in S with better point(s), per iteration. In particular, DE attempts to replace all points in S by new points

* Corresponding author.
E-mail address: mali@cam.wits.ac.za (M.M. Ali).

at each iteration. DE, therefore, progresses in an epoch or era base. During each epoch k, N new function values are evaluated on N trial points. Trial points are found using mutation and crossover. For the completeness of this paper a brief description of the DE algorithm is given below.

2. A brief description of DE

DE attempts to replace each point in S with a new better point. Therefore, in each iteration, N competitions are held to determine the members of S for the next iteration. The ith ($i = 1, 2, \ldots, N$) competition is held to replace x_i in S. Considering x_i as the target point a trial point y_i is found from two points (parents), the point x_i, i.e., the target point and the point $\hat{x}_i$ determined by the mutation operation. In its mutation phase DE randomly selects three distinct points $x_{p(1)}$, $x_{p(2)}$ and $x_{p(3)}$ from the current set S. None of these points should coincide with the current target point x_i. The weighted difference of any two points is then added to the third point which can be mathematically described as:

$$\hat{x}_i = x_{p(1)} + F(x_{p(2)} - x_{p(3)}), \tag{1}$$

where $F > 0$ is a scaling factor, and $x_{p(1)}$ is known as the base vector. If the point $\hat{x}_i \notin \Omega$ then the mutation operation is repeated. The trial point y_i is found from its parents x_i and $\hat{x}_i$ using the following crossover rule:

$$y_i^j = \begin{cases} \hat{x}_i^j & \text{if } R^j \leqslant C_R \text{ or } j = I_i, \\ x_i^j & \text{if } R^j > C_R \text{ and } j \neq I_i, \end{cases} \tag{2}$$

where I_i is a randomly chosen integer in the set I, i.e., $I_i \in I = \{1, 2, \ldots, n\}$; the superscript j represents the jth component of respective vectors; $R^j \in (0, 1)$, drawn uniformly for each j. The ultimate aim of the crossover rule (2) is to obtain the trial vector y_i with components coming from the components of target vector x_i and mutated vector $\hat{x}_i$. And this is ensured by introducing C_R and the set I. Notice that for $C_R = 1$ the trial vector y_i is the replica of the mutated vector $\hat{x}_i$. The effect of C_R has been studied in [2,3] and it was found that $C_R = 0.5$ is a good choice. The targeting process continues until all members of S are considered. After all N trial points y_i have been generated, acceptance is applied. In the acceptance phase the function value at the trial point, $f(y_i)$, is compared to $f(x_i)$, the value at the target point. If $f(y_i) < f(x_i)$ then y_i replaces x_i in S, otherwise, S retains the original x_i. Reproduction (mutation and crossover) and acceptance continues until some stopping conditions are met.

An important issue that needs to be addressed is the value of the scaling factor F in (1). To the best of our knowledge, no optimal choice of the scaling factor F has been suggested in the literature of DE. For instance, in [4] F is a value in [0.4, 0.8], in [5] a parameter dependent an-isotropic value and in [6] dynamically calculated values are suggested. All of these are derived empirically, and in most cases choice of F varies from 0.4 to 1. However, in original DE [1] F was chosen to lie in [0, 2]. It appears that there is no coherent and systemic study using a large set of problems in seeking the optimal choice of F and the suggested values of the scaling factor have been largely dependent on small test problem sets used. In this paper, we introduce F as a uniform random variable in $[-1, -0.4] \cup [0.4, 1]$. Instead of keeping F constant throughout the course of the DE algorithm, we make F random for each trial point.

The main difficulty with the DE technique, however, appears to lie in the slowing down of convergence as the region of global minimum is approached and we have attempted to remedy this defect by incorporating two new ideas into the algorithm. Firstly, we use the tournament selection to obtain the base vector in the mutation rule (1). The process of randomly selecting three vectors and obtaining the best within these three is referred to as the tournament selection. We randomly select three vectors from S and the best vector, x_{tb}, within these three is used as the base vector for mutation. This change has a local effect when the points in S form a cluster around the global minimizer. Our first modification, therefore, is based on the base vector in the mutation. Secondly, after calculating each trial point y_i ($i = 1, 2, \ldots, N$) using the mutation (1) and crossover (2) of DE, the regions around the trial vector

y_i and the best vector, x_b, in S are then explored using some reflection and contraction process. These modifications result in two new versions of DE which will be described in the next section. Section 4 cites the source of 50 test problems used for the numerical study and also presents numerical results. Finally, Section 5 contains the concluding remarks.

3. Modified differential evolution algorithms

In this section, we propose two new versions of DE. The first version modifies the mutation rule. It randomly selects three points in S and explores the region around the tournament best x_{tb} for each mutated point. It also implements the variable F_i. This random localization feature gradually transforms itself into the search intensification feature for rapid convergence when the points in S form a cluster around the global minimizer. This version of DE is referred to as the differential evolution algorithm with random localization (DERL). The second new version extends the original DE by a single step in that for each trial point y_i generated by DE using (1) and (2), the new algorithm explores the regions in and around y_i and x_b, the current best vector in S, using some contraction and reflection rule before the acceptance rule can be applied. This feature of localization of searches around the best vector has a global exploration effect at the early stages when the points in S are scattered and a local effect in terms of rapid convergence at the later stages. This version of DE is referred to as the differential evolution algorithm with localization using the best vector (DELB).

3.1. DERL, a new differential evolution algorithm with random localizations

In the original DE three vectors are chosen at random for mutation and the base vector is then chosen at random within the three. This has an exploratory effect but it slows down the convergence of DE. Also the original DE uses a fixed positive value for the scaling parameter F in mutation. This has an effect of restricting the exploration. Our first modification to DE is to replace the random base vector $x_{p(1)}$ in the mutation rule (1) with the tournament best x_{tb}. From the three random vectors the best is used as the base vector and the remaining two are used to find the differential vector in (1). This process explores the region around each x_{tb} for each mutated point. This maintains the exploratory feature and at the same time expedites the convergence. Also, instead of using a fixed F throughout a run of DE, we use a random F in $[-1,-0.4]\cup[0.4,1]$ for each mutated point. This also improves the exploration. Although these modifications are very simple and naive, numerical results using 50 test problems have shown that the effects of these changes are considerable. We present the new DE algorithm based on these changes.

Algorithm DERL

Step 1: **Determine the initial set**

$$S=\{x_1,x_2,\ldots,x_N\},$$

where the points x_i, $i=1,2,\ldots,N$ are sampled randomly in Ω; evaluate $f(x)$ at each x_i, $i=1,2,\ldots,N$. Take $N\gg n$, n being the dimension of the function $f(x)$. Set iteration counter $k=0$.

Step 2: **Determine best, worst point in** S. Determine the points x_{max} and x_{min}. If the stopping condition such as $|f_{max}-f_{min}|\leqslant\epsilon$ is satisfied, then stop.

Step 3: **Generate points to replace points in** S for the next population (or iteration). For each $x_i\in S$ $(i=1,2,\ldots,N)$, determine y_i by the following two operations:

- **Mutation**:

$$\hat{x}_i=x_{tb}+F_i(x_{p(2)}-x_{p(3)}), \tag{3}$$

where x_{tb} is the tournament best and $x_{p(2)}$ and $x_{p(3)}$ are the remaining two out of the three random vectors from S and $F_i\in[-1,-0.4]\cup[0.4,1]$, chosen randomly. The tournament selection is applied for each i. If $\hat{x}_i\notin\Omega$ then we select another F_i.

- **Crossover**: Calculate the trial vector y_i corresponding to the target x_i from x_i and $\hat{x}_i$ using the crossover rule (2).

Step 4: **Acceptance rule to replace points in** S. Select each trial vector y_i for the $k+1$ iteration using the acceptance criterion: replace $x_i \in S$ with y_i if $f(y_i) < f(x_i)$ otherwise retain x_i. Set $k := k+1$ and go to Step 2.

Remarks

1. All the steps in the DERL algorithm are similar to the DE algorithm, except that the mutation rule (1) is replaced by (3). Unlike DE, the scale F_i is different for each $\hat{x}_i$.
2. A suggested value for N is $10n$, where n is the dimension (see [3], for a full discussion on the choice of N).
3. The points $x_{\max}$ and $x_{\min}$ and their function values $f_{\max}, f_{\min}$ are such that

$$f_{\max} = \max_{x \in S} f(x) \quad \text{and} \quad f_{\min} = \min_{x \in S} f(x).$$

4. The mutated point $\hat{x}_i$ generated by (3) is not necessarily local to x_{tb}. It is only local when the size of the differential vector $F_i(x_{p(2)} - x_{p(3)})$ is small which is less likely when the points in S are scattered around the search space Ω and more likely when the points form a cluster around the vicinity of a minimizer. Therefore, at the early stage the search will be exploratory and at the later stage it is localized.

3.2. DELB, a new differential evolution algorithm with localizations around the best vector

The use of x_{tb} in the mutation rule (3) of DERL is a salient feature whereby it explores the region Ω when N points are sparse and expedite its convergence as soon as they form a dense cluster. This random localization is used in mutation prior to the calculation of y_i. While this localization in DERL has a role to play in producing successful trial points y_i, in certain situations it may be preferable to adopt a localization after the trial point y_i has been calculated. When the trial points y_i are favourable we explore the regions around them further. This tends to speed up the convergence. Consequently, we have devised an algorithm that explores the regions around y_i and the current best x_{b} in S. In particular, each time a trial point y_i is found by DE which is better than x_i but worse than x_{b}, two new points, r_i and c_i, are found, with some probability, using the following rules:

$$r_i = x_{\text{b}} - (y_i - x_{\text{b}}), \tag{4}$$

and

$$c_i = x_{\text{b}} + 0.5(y_i - x_{\text{b}}). \tag{5}$$

We also implemented the following scheme for calculating r_i and c_i, where the points r_i and c_i are again obtained from $x_{\text{b}} = (x_{\text{b}}^1, x_{\text{b}}^2, \ldots, x_{\text{b}}^n)$ and $y_i = (y_i^1, y_i^2, \ldots, y_i^n)$ using

$$r_i^j = \alpha^j y_i^j + (1 - \alpha^j) x_{\text{b}}^j, \tag{6}$$

and

$$c_i^j = \alpha^j x_{\text{b}}^j + (1 - \alpha^j) y_i^j, \quad j = 1, 2, \ldots, n, \tag{7}$$

where α^j are uniform random numbers in $[-1, 1]$ for each j. The best of the function values $f(y_i)$ and $f(r_i)$ or $f(c_i)$ is then compared with $f(x_i)$ during the acceptance rule. Eqs. (4) and (5) are special cases of Eqs. (6) and (7) respectively. For $\alpha^j = -1$ Eq. (6) reduces to (4), and for $\alpha^j = 0.5$ (7) reduces to (5). If the second scheme is used for obtaining r_i and c_i, Eq. (4) in sub Step 4b and Eq. (5) in sub Step 4c are replaced respectively by (6) and (7). This modification results in the following algorithm.

Algorithm DELB

Step 1: **Determine the initial set** S: same as for DERL or DE.

Step 2: **Determine best, worst point in** S: same as for DERL or DE.

Step 3: **Generate points to replace points in** S for the next population. For each $x_i \in S$ $(i = 1, 2, \ldots, N)$, determine y_i by the following two operations:

- **Mutation**:

$$\hat{x}_i = x_{p(1)} + F_i(x_{p(2)} - x_{p(3)}), \tag{8}$$

where $x_{p(1)}$, $x_{p(2)}$ and $x_{p(3)}$ are random vectors from S and $F_i \in [-1, -0.4] \cup [0.4, 1]$.

- **Crossover**: same as for DERL or DE, i.e., using (2).

Step 4: **Localization and acceptance rule to replace points in** S. For $i = 1, 2, \ldots, N$, do the following steps.

Step 4a: if $f(y_i) \geqslant f(x_i)$ then retain x_i, otherwise if $R^i < w$ and $f(y_i) > f(x_b)$ then go to Step 4b else go to Step 4d.

Step 4b: **Reflection**: Find r_i using (4) and calculate $f(r_i)$. If $f(r_i) < f(y_i)$ then replace x_i by r_i and go to Step 4a for the next index. Otherwise go to Step 4c.

Step 4c: **Contraction**: Find c_i using (5) and calculate $f(c_i)$. If $f(c_i) < f(y_i)$ then replace x_i by c_i and go to Step 4a for the next index. Otherwise go to Step 4d.

Step 4d: Replace x_i by y_i and go to Step 4a for the next index.

Step 5: Set $k = k + 1$ and go to Step 2.

Remarks

1. All the steps in the DELB algorithm are similar to the DE algorithm, except the variable F_i in mutation rule (8), and the extended form of the Step 4, in particular the sub-steps 4b and 4c in Step 4.
2. R^i in Step 4a is a random number in $(0, 1)$, and w is a user provided value within $(0, 1)$. w controls the frequency of local exploration around y_i and x_b.
3. Unlike DE and DERL, where the trial point y_i competes with the target x_i, for DELB the trial point y_i and either r_i or c_i compete with the target x_i.
4. If the mutation or the localization procedure used by (6) and (7) finds a point outside of Ω that process is repeated in order to obtain a point inside the search region. In case of reflection (4) going outside of Ω then the contraction (5) is carried out.

4. Numerical results

Performance of the new algorithms described thus far was judged using a collection of 50 test problems. These problems range from 2 to 20 in dimension and have a variety of inherent difficulty. All the problems have continuous variables. A detailed description of each test problem (P) in the collection can be found in [7].

In this section we compare the DERL and DELB with the original DE algorithm. The algorithms were run 100 times on each of the test problems to determine the percentages of success (ps). There were 5000 runs in total. We calculated the average number of function evaluations (fe) and average cpu time (cpu) for those runs for which the global minima were found. The ps, fe and cpu were taken as criteria for comparison. We note that except for the 9 dimensional Price (PTM) and 10 dimensional Odd Square (OSP), Salomon (SAL) and Shekel Foxhole (FX) functions [7], all algorithms have positive success rate on the remaining 46 problems. Therefore, the results for these problems are not reflected in averages. We also note that accuracy of DE on 9 dimensional Storn function (ST) [7] is much less than the accuracies obtained by DERL and DELB [3]. However, the results for ST are reflected in all averages.

The optimal parameter values for all algorithms were found empirically using 50 test problems. We do not claim these values to be the optimal for any problem in general but they will be good values to choose. The parameters of DE are the scaling parameter F in its mutation rule (1) and controlling parameter C_R in its crossover (2). We have conducted a series of runs of DE using values (F varying from 0.3 to 1.25; C_R from 0.25 to 0.9) for each of these parameters. The best results obtained using 50 problems were for $F = 0.5$ and $C_R = 0.5$. We will present the best results here, although the full set of results for all runs can be found in [3]. It was found that a good choice for C_R is 0.5 for all algorithms (see, [3]). The value of ϵ in the stopping rule $|f_{\max} - f_{\min}| \leqslant \epsilon$ was chosen to be 10^{-4} for all algorithms: DE, DERL and DELB.

To motivate our first modification, DERL, we present the results showing the effect of x_{tb} and F_i separately. The combined effect of these two changes results in DERL. Therefore, we compare the results obtained by original DE($F = 0.5$, $C_R = 0.5$) with those obtained by DE^1(F_i random,

Table 1
Effect of random F_i and x_{tb} in DE

	DE			DE^1			DE^2		
	fe	cpu	ps	fe	cpu	ps	fe	cpu	ps
tr	2,131,089	73.85	4298	2,113,963	72.84	4474	1,345,425	39.20	4319
rpf	46,328	1.61	93.43	45,956	1.58	97.26	29,248	0.85	93.89

$C_R = 0.5$) and DE^2($F = 0.5$, $C_R = 0.5$, x_{tb} in (1)). Parameter values are given in parenthesis. DE^1 is the original DE algorithm with F_i random in mutation, and DE^2 is the original DE algorithm with x_{tb} as the base vector. Results of these algorithms are presented in Table 1. For the compactness of presentation, we have totalled all averages fe and cpu, and ps for all problems. In Table 1, tr represents the total result and rpf the result per function. We calculated rpf by dividing the total result by 46. It is clear from Table 1 that the incorporation of F_i enhances the robustness of DE considerably. In particular, in terms of ps DE^1 is superior to DE by 176 successes, although it is slightly inferior to DE in terms of fe and cpu. However, the incorporation of x_{tb} in (1) makes DE much more superior in all respects. For instance, DE^2 is superior to DE by 35% and 45% respectively in terms of fe and cpu, while achieving 21 more successes in ps than DE. The results of DERL will be presented along with the results of DELB shortly.

Unlike DERL, DELB does not use the x_{tb} as the base vector in the mutation rule (8), rather it uses random $x_{p(1)}$ as in the case of DE. This will also be justified later, but first we study the effect of the parameter w which controls the frequency of localization around x_b. As in the previous case, we study the effect of localization in DE using the best vector instead. We obtained about the similar results for DELB using both localization schemes given by (4)–(7). These results are fully reported and discussed in [3]. However, the results of DELB presented in this paper were obtained by implementing (4) and (5). We compare the results of original DE($F = 0.5$, $C_R = 0.5$) with those of DE^3($F = 0.5$, $C_R = 0.5$, localization using (4) and (5) after each y_i). Clearly, for $w = 0$ DE^3 becomes original DE. We present this comparison in Table 2 to see the effect of w. Again we use tr and rpf for various values of w. From Table 2, it is clear that fe decreases with the increase of w. This trend is not true for ps. Overall best results were obtained when $w = 0.1$. For the values after this value ps worsens rapidly and for values before this both fe and ps worsen. Therefore, we implemented DELB with $w = 0.1$. Notice that DELB is different from DE^3 in that it implements random F_i in (1). To see the improved results obtained by the new algorithms on each problem we present results on each problem for DE, DERL and DELB in Table 3. From the total results in the last row in Table 3, it is clear that both implementations of our new algorithm perform much superior to DE in terms of fe, ps and cpu. For example, DERL is superior to DE by 30% and 41% in terms of fe and cpu, and DELB by 15% and 28%. Although DERL is the best algorithm in terms of fe and cpu, in terms of ps it is runner up. In total ps DELB is superior to DERL by 113 successes and superior to DE by 147. Moreover, it is also clear from this table that the total results are dominated by the results of four difficult problems, namely EM, ML, RB and ST. The total results on these

Table 2
Effect of w in DE

	$DE = DE^{3,w=0}$		$DE^{3,w=0.05}$		$DE^{3,w=0.1}$		$DE^{3,w=0.25}$		$DE^{3,w=0.40}$	
	fe	ps	fe	ps	fe	ps	fe	ps	fe	ps
tr	2,131,089	4298	1,777,536	4430	1,597,841	4456	1,275,127	4408	1,127,897	4341
rpf	46,328	93.43	38,642	96.30	34736	96.86	27,720	95.82	24,433	94.36

Table 3
Comparison of DE, DERL and DELB using 50 problems

P	n	DE			DERL			DELB		
		fe	ps	cpu	fe	ps	cpu	fe	ps	cpu
ACK	10	25,810	100	0.56	21,583	100	0.43	20,395	100	0.45
AP	2	683	100	0.006	567	100	0.005	700	100	0.007
BL	2	991	100	0.01	762	100	0.007	1004	100	0.01
B1	2	977	100	0.009	815	100	0.01	995	100	0.01
B2	2	1039	100	0.009	869	100	0.008	1061	100	0.01
BR	2	1030	100	0.01	772	100	0.007	1047	100	0.01
CB3	2	717	100	0.006	590	100	0.005	730	100	0.007
CB6	2	976	100	0.009	746	100	0.008	965	100	0.01
CM	4	2300	100	0.03	1948	100	0.02	2203	100	0.03
DA	2	1277	100	0.01	1021	100	0.01	1268	100	0.01
EP	2	650	95	0.006	542	99	0.005	629	96	0.006
EM	10	249,958	85	10.35	182,696	56	7.63	248,509	87	10.18
EXP	10	9903	100	0.20	8281	100	0.16	7860	100	0.16
GP	2	937	100	0.008	763	100	0.007	948	100	0.01
GW	10	14,705	100	0.32	12,271	100	0.25	11,841	100	0.27
GRP	3	3024	98	0.25	2261	100	0.19	2858	100	0.24
H3	3	1220	100	0.01	988	100	0.01	1209	100	0.01
H6	6	6836	99	0.12	4980	99	0.10	6686	98	0.13
HV	3	3782	99	0.04	2942	100	0.03	3654	100	0.04
HSK	2	559	100	0.005	467	100	0.004	579	100	0.006
KL	4	1839	100	0.02	1546	100	0.02	1691	100	0.02
LM1	3	1465	100	0.02	1240	100	0.01	1457	100	0.02
LM2	10	11,583	100	0.25	9531	100	0.22	9222	100	0.21
MC	2	639	100	0.005	531	100	0.006	659	100	0.007
MR	3	3270	74	0.03	2538	77	0.03	3202	80	0.04
MCP	4	3672	100	0.04	2136	100	0.03	2400	100	0.03
ML	10	200,673	70	4.95	89,629	56	2.22	173,153	71	4.32
MRP	2	1485	64	0.01	1058	65	0.01	1449	63	0.01
MGP	2	1032	68	0.01	775	65	0.01	1070	70	0.01
NF2	4	80,475	100	0.93	42,282	100	0.56	94,189	100	1.24
NF3	10	84,620	100	1.67	61,310	100	1.30	79,602	100	1.65
OSP	10	0	0	0	0	0	0	0	0	0
PP	10	14,785	100	0.33	11,654	100	0.29	12,107	100	0.28
PRD	2	1862	90	0.02	1239	82	0.02	1216	98	0.01
PQ	4	5346	100	0.06	4290	100	0.05	5048	100	0.06
PTM	9	0	0	0	0	0	0	0	0	0
RG	10	98,303	100	2.05	95,991	100	2.13	114,138	100	2.50
RB	10	265,700	2	18.22	176,113	100	3.70	192,676	98	3.95
SAL	10	0	0	0	0	0	0	0	0	0
SF1	2	3079	63	0.04	3173	41	0.03	3653	85	0.03
SF2	2	1990	100	0.02	1627	100	0.01	2025	100	0.02
SBT	2	2581	100	0.03	1939	100	0.02	2704	100	0.03
SWF	10	28,157	100	0.99	21,862	100	0.91	24,792	100	1.04
S5	4	5734	95	0.07	3914	93	0.05	5097	99	0.07
S7	4	4707	100	0.05	3497	100	0.05	4156	100	0.06
S10	4	4788	100	0.06	3599	100	0.05	4343	100	0.06
FX	10	0	0	0	0	0	0	0	0	0
SIN	20	61,242	100	2.99	48,610	100	2.33	34,919	100	1.58
ST	9	900,090	100	28.84	646,380	100	21.00	700,402	100	23.47
WP	4	14,598	96	0.16	10,389	100	0.13	12,222	100	0.16
tr		2,131,089	4298	73.85	1,492,717	4332	44.08	1,802,733	4445	52.59

problems show the superiorities of the new algorithms over DE even more. In particular, the total ps on these problems obtained respectively by DE, DERL and DELB are 257, 312 and 356. We further compare the algorithms excluding these problems. The total (fe, ps, cpu) obtained respectively by DE, DERL and DELB are (514,668, 4041, 11.53), (397,899, 4020, 9.61) and (487,993, 4089, 10.63). Clearly, DERL obtained about the same level of ps as DE but it obtained much lower fe and cpu. On the other hand, DELB achieved superiority over DE in terms of fe and ps. In particular DELB has obtained much more ps than DE while their cpu are about the same level. It is therefore, clear that for the four difficult problems the new algorithms are much superior to DE in all respects. For the remaining 42 less difficult problems the superiorities of the new algorithms over DE still hold. One can notice that DERL reduces the fe and cpu while DELB increases ps considerably. We next combine these complementary strengths to see their combined effect by simply replacing the random $x_{p(1)}$ with x_{tb} in mutation rule (8) in DELB. As before, using parenthesis we can write this as $DE^4(F_i$ random, $C_R = 0.5$, x_{tb} in (8), localization around $x_b)$. DE^4 therefore, uses both localizations, i.e., random localization using the tournament best and the localization around the best vector.

4.1. Effect of global and local techniques

The essential differences between DE and DERL are that DERL uses variable F_i whereas DE uses fixed F, and that DERL uses the tournament best point x_{tb} while DE uses a random point as the base vector in their respective mutation rule. On the other hand, DELB differs from DE in that it uses random F_i and that for each trial point y_i it explores the vicinity of the best point x_b with some probability. If we fix F_i and let the probability w be zero then DE and DELB are the same. To show that the proposed modifications to the DE algorithm prevails in their superiority over the original, we have carried out a comprehensive numerical study. The results, which were found to be very encouraging are reported in full in [3] where we have also considered the accuracy of the global minimum as one of the indicators. However, we would like to summarise the results in a compact format that will also be clearly understandable. In order to facilitate the understanding and to make the difference between the methods more explicit we append to each individual algorithm name the appropriate parameter containing '(global or local effect, local effect, parameter F)'. If an algorithm does (does not) use x_{tb} as the base vector, we say it has a local (global) effect. The global and local effects are complementary to each other with respect to the parameter in the first component. However, since the local technique in DELB is not related to the base vector in the mutation rule, it is reflected in the second component. DE and DERL do not have local effect in the second component. We denote the global and local effect respectively by g and l, and the parameter F by f or v depending upon whether it is fixed or variable. Using these notation we thus write DE, DERL and DELB as $DE^{(g,\cdot,f)}$, $DE^{(l,\cdot,v)}$ and $DE^{(g,l,v)}$. We can also write DE^1, DE^2, DE^3 and DE^4 respectively as $DE^{(g,\cdot,v)}$, $DE^{(l,\cdot,f)}$, $DE^{(g,l,f)}$ and $DE^{(l,l,v)}$. We also include $DE^{(l,l,f)}$ to see the effect of the fixed scaling factor F and x_{tb} as the base vector in DELB. We compare these algorithms in Table 4 using total results to see the effects of the parameters such as the base vector, localization around x_b and F. It can be seen from Table 4 that the dominant factor in reducing the fe has been the random localization, i.e., the use of l in the first component of the parameters of the algorithms. This is true for both the new algorithms. One can see this by comparing the fe under DERL = $DE^{(l,\cdot,v)}$ and $DE^{(g,\cdot,v)}$, and DELB = $DE^{(g,l,v)}$ and

Table 4
Effect of local and global techniques

	$DE^{(g,\cdot,f)}$	$DE^{(l,\cdot,v)}$	$DE^{(g,l,v)}$	$DE^{(l,\cdot,f)}$	$DE^{(g,\cdot,v)}$	$DE^{(g,l,f)}$	$DE^{(l,l,v)}$	$DE^{(l,l,f)}$
fe	2,131,089	1,492,717	1,802,733	1,345,425	2,113,963	1,597,841	1,139,903	1,050,976
ps	4298	4332	4445	4319	4474	4456	4359	4325

$DE^{(l,l,o)}$. The same is true for other implementations, e.g., for $DE^{(g,l,f)}$ and $DE^{(l,l,f)}$. On the other hand, fixing the scaling factor F in the last component has an effect of reducing the success rate ps. This is clear if we compare the ps under $DE = DE^{(g,\cdot,f)}$ and $DE^{(g,\cdot,o)}$, and the ps under $DE^{(l,l,o)}$ and $DE^{(l,l,f)}$. By comparing the results under the last three columns we see that the implementation of both local techniques has a negative effect on ps. The choice of parameter values is flexible, and it is upto the users to decide which combination of parameter values to choose. This may differ from user to user depending on whether they want more ps or less fe, or both.

5. Conclusion

We introduced two new algorithms and tested them on 50 benchmark test problems with a maximum dimension of 20. From the comparison in the previous section it is quite clear that the new algorithms are superior to the original DE. We have made every effort to make the comparison as fair as possible despite the differences in the accuracy of the global minimum obtained by different algorithms. If we had used the accuracy of the solution found as a criterion for comparison, the new algorithms would have been much more superior [3]. The new algorithms could also be preferable because they are simple and easily programmable. There are clearly circumstances in which an easily implementable, direct search, global optimization algorithm will have an important role. Consequently, we feel that the introduction of our two new algorithms, with their substantial improvement over the original DE algorithm, is justified. Therefore, using these direct search methods, which have shown to be robust and efficient, could be rewarding.

A disturbing fact concerning DE algorithms is their totally heuristic nature with no theoretical convergence properties. Further research is underway in developing convergence properties and in developing an efficient hybrid DE global optimization method for large dimensional problems.

References

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

[2] M.M. Ali, A. Törn, Population set based global optimization algorithms: Some modifications and numerical studies, Computers and Operations Research 31 (10) (2004) 1703–1725.

[3] P. Kaelo, Some population set based methods for unconstrained global optimization, PhD thesis, in preparation.

[4] K. Price, An introduction to differential evolution, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, London, 1999, pp. 79–108.

[5] D. Zaharie, Critical values for the control parameters of differential evolution algorithms, in: R. Matousek, P. Osmera (Eds.), Proceedings of MENDEL 2002, 8th International Mendel Conference on Soft Computing, Bruno, Czech Republic, Bruno University of Technology, Faculty of Mechanical Engineering, Bruno, Czech Republic, 2002, pp. 62–67.

[6] M.M. Ali, A. Törn, Topographical differential evolution using pre-calculated differentials, in: G. Dzemyda, V. Saltenis, A. Zilinskas (Eds.), Stochastic and Global Optimization, Kluwer Academic Publisher, London, 2002, pp. 1–17.

[7] M.M. Ali, C. Khompatraporn, Z.B. Zabinsky, A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems, Journal of Global Optimization, in press.