http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 10 (3): 573-578, 2011
ISSN 1812-5638 / DOI: 10.3923/itj.2011.573.578

A New Differential Evolutionary Algorithm with Neighborhood Search

[1,2]Yuzhen Liu and [1]Shoufu Li
[1]School of Mathematics and Computational Science,
Xiangtan University, Xiangtan, People's Republic of China
[2]College of Information Engineering, Xiangtan University, Xiangtan, People's Republic of China

Abstract: For Global Optimization Problems (GOPs), we propose a new variant of DE with linear neighborhood search, called LiNDE which employs a linear combination of triple vectors taken randomly from evolutionary population, in order to improve the ability of neighborhood search of Differential Evolutionary (DE) algorithm. The main characteristics of LiNDE are less parameters and powerful neighborhood search ability. Experimental studies are carried out on a benchmark set and the results show that LiNDE significantly improves the performance of DE.

Key words: Evolutionary computation, differential evolutionary algorithm, neighborhood search, global optimization problems, evaluation criterion, benchmarks

INTRODUCTION

Recently, many complicate scientific and engineering problems are difficult to be solved using traditional optimization methods and researchers begin to get help to intelligent optimization algorithms, such as DE algorithm (Storn and Price, 1997), particle swarm optimization (Eberhart and Kennedy, 1995) Among these intelligent optimization algorithms, DE algorithm has showed superior power in dealing with benchmark problems and real-world applications (Ilonen *et al*., 2003; Storn, 1996; Rogalsky *et al*., 1999; Joshi and Sanderson, 1999).

The classical DE algorithm (Storn and Price, 1997) contains five candidate mutation operators and triple control parameters: population size NP, scale factor F and crossover rate CR. The characteristic of multiple mutation schemes and control parameters allows DE adapting to various environments. However, how to choose suitable mutation scheme (s) and set parameter value are not easy tasks (Zaharie, 2002).

Control parameters setting is frequently problem-independent task and required previous experience knowledge and environmental information. Despite of its crucial importance, there is no common methodology to determine the values of all parameters and in most time, they are set with an arbitrary manner within some given ranges (Maruo *et al*., 2005). In general, the setting approaches of parameter values are divided into two major forms (Brest *et al*., 2006): parameter tuning and parameter self adaptation. The former approach tries to find better parameter values before running the algorithm. However, such a trial-and-error searching process required high computational cost (Qin *et al*., 2009). Moreover, a set of fix parameter values is not all efficient during the whole evolutionary process. The self-adaptive method of parameter setting is employed in some variants of DE algorithm. Qin *et al*. (2009) proposed a self-adaptive DE (SaDE), where crossover rate CR and mutation strategies are self-adapted according to their associated-success rate. Another self-adaptive strategy was proposed by Yang *et al*. (2008b). In the strategy, CR is allocated to each individual according to its success rate and the Gaussian distribution with standard deviation 0.1. CR is set to 0.5 initially and not updated within 5 generations. After a specified number of generations, the memory base, saving the history information of CR, is updated. And then CR is calculated.

Another issue about DE algorithm is choosing suitable mutation strategies in different evolutionary phases. Qin *et al*. (2009) employed a selection probability p to determine which strategy is used in different evolutionary phases. The probability p is gradually self-adapted according to the previous performance of the strategy. Therefore, those mutation strategies who improve the solution quality have large probability to be used. But some variants of DE algorithm employ one mutation strategy. Brest *et al*. (2006) proposed a self-adaptive control parameter DE algorithm, where only rand/1 mutation strategy is used and the algorithm, noted as jDE, shows good performance when optimizing GOPs.

In the study, we present a novel variant of DE algorithm, shorted for LiNDE, to dealing with GOPs.

Corresponding Author: Yuzhen Liu, College of Information Engineering, Xiangtan University, Xiangtan, Hunan, People's Republic of China Tel: +86-15898512860 Fax: +86-731-58292201

Proposed LiNDE uses random strategy when setting scale factor F and crossover rate CR. Meanwhile, we adapt rand/1 mutation strategy and introduce a neighborhood searching operator.

PRELIMARY OF DE

A set of D-dimensional vectors is called evolutionary population P containing NP vectors, where NP is the number of vectors. The vectors with D dimension is call individual and it can be noted as x_i, $x_i = (x_{i1}, x_{i2} \ldots, x_{iD})$. The classic DE algorithm can be summarized as follows:

Mutation operator:

$$v_i = x_{r_1} + F*(x_{r_2} - x_{r_3}), r_1 \neq r_2 \neq r_3 \quad (1)$$

where, $v_i = (v_{i1}, v_{i1},\ldots,v_{iD})$ is a trial vector generated by mutation operator and x_{ri} , i = 1, 2, 3 are individuals taken from P randomly. F is a scale factor controlling the step size of the difference vector $(x_{r2} - x_{r3})$.

If a component of the trial vector v_i is beyond the given bound predefined by problem, then it is reinitialized randomly.

In classic DE algorithm, there are for different mutation strategies. For detailed description and analysis behind them (Storn and Price, 1997).

Crossover operator: The operator combines v_i and x_{r1}, then gets the target vector u_i, $u_i = (u_{i1}, u_{i2},\ldots u_{iD})$.
where,

$$u_{ij} = \begin{cases} v_{ij} & \text{if } U(0,1) < CR \text{ or } j = j\_rand \\ x_{ij} & \text{otherwise} \end{cases} \quad (2)$$

In Eq. 2, U (0, 1) stand for random distribution, j-rand is a random number chosen from {1, 2,…NP} to ensure target vector u_i getting at least one component from x_{r1}.

Selection operator: A greedy selection scheme is used in DE algorithm

$$x_i' = \begin{cases} u_i & \text{if } u_i \text{ is superior } x_i \\ x_i & \text{otherwise} \end{cases} \quad (3)$$

where x'_i enters the next generation.

According to Storn and Price (1997), DE is much more sensitive to the choice of F and CR. However, various conflicting conclusions have been drawn with regard to the rules for manually choosing the strategy and control parameters (Qin *et al*., 2009). Thus, in order to decrease the impact of control parameters, we set F and CR to a random number within (0, 1) and (0.5, 1), respectively. Another parameter is the size of evolutionary population NP. Although various setting method are given Storn and Price (1997) and Price (1999), there is no a universal method.

Consequently, we study the control parameter through a series of experiments to select a suitable value.

NEIGHBORHOOD SEARCH

Rand/1 mutation operator is used. Meanwhile, we propose a new mutation strategy searching the neighborhood of an individual. The proposed operator is given as follows:

$$v_i = \varpi * x_{r_1} + (1 - \varpi) * (x_{r_2} - x_{r_3}) \quad (4)$$

where ϖ is a scale factor generated through U (0,1) x_{ri}, I – 1, 2, 3 are individuals taken from P randomly and $r_i \neq r_2 \neq r_3$.

Figure 1a and b show the distribution of offspring individuals generated by neighborhood searching

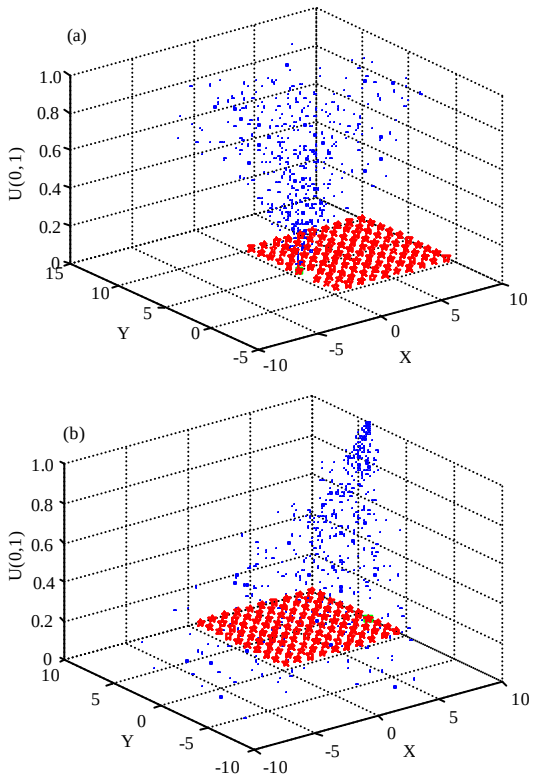

Fig. 1: The distributing illustrations of offspring

Table 1: Framework of linde

Algorithm LiNDE	
1.	Initializing evolutionary population P randomly within the given space predefined by the problem.
2.	Evaluating fitness values of every individual from P according to problem.
3.	For every individual of P, do
(a)	Generating trial vector v according to rand/1 and neighborhood searching operators with equal probability. And the scale factor F used in rand/1 operator is set to a random number.
(b)	Generating target vector u using parent vector x and trial vector v according to CR. CR = 0.5*U (0,1) +0.5.
(c)	Checking whether u is beyond bounds, If a component is beyond, then reinitialize it.
(d)	Evaluate the fitness value of target vector u.
(e)	Select a superior vector entering next generation between u and x according to their fitness values.
4.	Repeating step 3 until the terminal conditions are satisfied.
5.	Outputting the experimental results.

operator and rand/1 operator, respectively. Meanwhile, we calculate the deviation of offspring individuals generated by two operators, and their deviations are 2.04 and 3.26, respectively. Therefore we conclude the proposed neighborhood searching operator is suitable to exploit the neighborhood of any individuals.

FRAMEWORK OF LiNDE

It can be concluded that how to set control parameters and how to select strategies have been pay special attention in DE algorithm. However, the ability of neighborhood search has been neglected to some extent. Motivated by the above observation, we propose a novel DE algorithm with neighborhood searching operator. The framework of LiNDE is given as Table 1.

PERFORMANCE EVALUATION

The performance evaluation is important when comparing various population-based algorithms. However, how to evaluate the performance has no specific method, therefore we propose a novel performance evaluation criterion.

Suppose there are k test functions and their best optima are $t_1, t_2, \ldots t_k$, so the vector of best optima can be noted as $T = (t_1, t_2, \ldots t_k)$. For a given algorithm A employed to optimize the above k test functions, after a certain independent runs, the algorithm outputs the best mean vector M and standard deviation vector S for k test functions and $M = (m_1, m_2, \ldots m_k)$, $S = (s_1, s_2, \ldots s_k)$. Therefore, the performance evaluation criterion is as the following form Eq. 5

$$PEv(A) = \varpi\sqrt{\sum_{i=1}^{k}\frac{(t_i - m_i)^2}{k}} + (1-\varpi)\sqrt{\sum_{i}^{k}\frac{s_i^2}{k}} \quad (5)$$

where, ϖ is a weight factor.

The performance evaluation criterion gives the overall performance in term of solution quality and stability for evaluated algorithm, Fig. 2 show the procedure of evaluation.

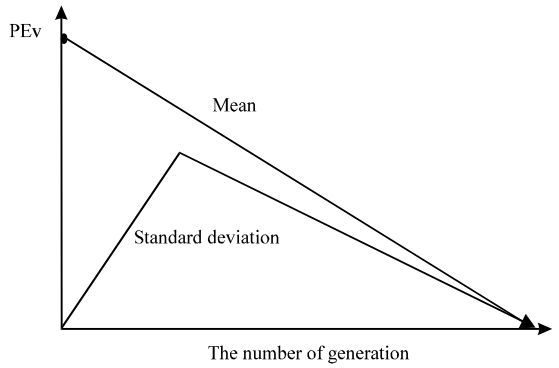

Fig. 2: The convergence of mean vector and standard deviation vector in the process of optimization

Theorem 1 If a algorithm named A obtains all best optima when optimizing k test functions in a certain independent runs, in addition, all standard deviations for all functions are zeros, then Pev (A) = 0.

Proof: Suppose the vector of best mean obtained by A for k test functions is $M = (m_1, m_2, \ldots m_k)$ and the corresponding vector of standard deviation is $S = (s_1, s_2, \ldots s_k)$; The vector of best optima for k test functions is $T = (t_1, t_2, \ldots t_k)$. Consequently, the value of performance evaluation can be computed through the above form Eq. 6. The algorithm obtains all best optima in a certain independent runs, therefore M = T, namely, mi = ti, i = 1, 2, .., k. Then the item

$$\sqrt{\sum_{i=1}^{k}\frac{(t_i - m_i)^2}{k}} = 0 \quad (6)$$

Following the same idea, the next item

$$\sqrt{\sum_{i}^{k}\frac{s_i^2}{k}} = 0 \quad (7)$$

Consequently,

$$PEv(A) = \varpi\sqrt{\sum_{i=1}^{k}\frac{(t_i - m_i)^2}{k}} + (1-\varpi)\sqrt{\sum_{i}^{k}\frac{s_i^2}{k}} = 0 \quad (8)$$

Deduction: Suppose two algorithms A and B are employed to optimize k test functions. The overall optimization process is independently done for a given times. If A performs better than B overall, then Pev (A)<PEv (B).

EXPERIMENTAL STUDIES

Experimental studies includes three parts: (1) A brief introduction to benchmarks; (2) A serial experiments on how to tuning parameters used in LiNDE; (3) A comparison studies with SaNSDE (Yang *et al*., 2008a), SaDE (Brest *et al*., 2006) and NSDE (Yang *et al*., 2008a).

Benchmarks: Experimental validation for the proposed LiNDE is conducted on 13 classical benchmarks (Yao *et al*., 1999), which can be divided into two classes: (1) F1-F7 are high dimensional unimodal problems employed to validate the convergence speed of algorithm; and (2) F8-F13 are high dimensional multimodal problems where the number of local optima increases exponentially with the problem dimension and they are utilized to demonstrate the searching ability of escaping local optima and tracing the global optima.

Parameter tuning: Two parameters population size NP and selection probability p are employed in LiNDE. In order to obtain the best performance of LiNDE, it is necessary to tune control parameters.

Firstly, we tune NP by optimizing 13 benchmarks when set p = 0.5 and Fes = 1e+5. The experimental results are listed in Table 2. For unimodal functions F1-F7 LiNDE achieved much better results when using small population size (10, 20 and 30) than that of large population size (50 and 100) except on the simple step function F6. The explanation behind the results is that: LiNDE using a small population could evaluate more iterations than that of large population. Therefore, the fitness values of unimodal functions could reach higher precision. However, the performance of LiNDE with large population size is better than that of LiNDE with small of population size when optimizing multi-modals functions F8-F13. The explanation behind the results is also the difference of iteration number.

According to the proposed evaluation criterion, LiNDE obtained the best performance when NP = 30 than that of other population size. Thus we set NP = 30 in later experiments.

Secondly, we conducted a series of experiments under various selection probability p when NP = 30 and Fes = 1e+5. Optimization results are given in Table 3 where the last row lists scores obtained by evaluation criterion. We can observe that: scores are almost identical when p = 0.25, 0.5 and 0.75 which means LiNDE is not sensitive to the parameter. Therefore, we set p = 0.5 in the subsequent experiments.

Performance of LiNDE: In order to obtain the performance of LiNDE optimizing benchmarks, we run the algorithm 25 times independently and experimental results are listed in Table 4. For unimodal functions F1-F7, LiNDE obtained the optima of F1, F2 and F6, meanwhile, the optimizing precision of F4 reached e-45. For multi-modal functions F8-F13, LiNDE algorithm performed better than that of unimodal functions and obtained higher precision for all benchmarks except for F8.

Comparison studies: In the experimental studies, we compare LiNDE with self-adaptive differential evolution (SaDE), differential evolution with neighborhood algorithm (NSDE) and self adaptive differential evolution with neighborhood search (SANSDE). In order to comparing the performance of different algorithms fairly, we adopted experimental data from reference. Then we

Table 2: Optimizing results with different population size NP when p = 0.5 and Fes = 1e+5

	Mean best (SD)				
Funs	NP = 10	NP = 20	NP = 30	NP = 50	NP = 100
F_1	8.69e-282 (0.00)	3.45e-155 (5.07e-155)	4.18e-95 (6.40e-95)	1.60e-52 (1.42e-52)	4.24e-23 (2.18e-23)
F_2	7.11e-153 (2.63e-152)	2.88e-85 (5.42e-85)	1.33e-52 (1.36e-52)	3.14e-29 (1.41e-29)	2.51e-13 (5.95e-14)
F_3	2.94e-2 (3.29e-2)	5.35e-1 (6.39e-1)	9.69e+1 (9.67e+1)	1.09e+3 (5.05e+2)	4.95e+3 (1.20e+3)
F_4	2.60e-50 (5.02e-50)	8.43e-28 (5.72e-28)	1.13e-16 (6.01e-17)	8.33e-9 (2.37e-9)	1.61e-3 (3.37e-4)
F_5	2.76e+1 (5.28e-1)	2.60e+1 (3.07e-1)	2.58e+1 (2.49e-1)	2.57e+1 (4.29e-1)	2.62e+1 (2.16e-1)
F_6	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
F_7	9.46e-4 (4.26e-4)	1.21e-3 (3.37e-4)	1.85e-3 (6.87e-4)	2.94e-3 (9.73e-4)	6.26e-3 (1.35e-3)
F_8	1.15e+3 (3.38e+2)	3.61e+2 (2.40e+2)	7.60e+1 (9.33e+1)	5.92 (2.65e+1)	7.47e-2 (4.97e-2)
F_9	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	1.42e-12 (2.93e-12)
F_{10}	4.44e-15 (0.00)	4.09e-15 (1.09e-15)	4.44e-15 (0.00)	4.44e-15 (0.00)	1.57e-12 (3.41e-13)
F_{11}	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
F_{12}	5.35e-4 (1.01e-3)	1.85e-32 (6.37e-33)	1.94e-29 (2.14e-29)	9.63e-18 (5.29e-18)	3.78e-9 (1.18e-9)
F_{13}	2.23e-2 (3.03e-2)	5.42e-3 (2.18e-2)	2.85e-28 (2.10e-28)	1.47e-16 (1.18e-16)	4.16e-8 (1.08e-8)
PEv	3.19e+2	1.00e+2	3.49e+1	3.03e+2	1.37e+3

Table 3: Optimizing results with different slection probability p when NP = 30 and Fes = 1e+5

	Mean best (SD)				
Funs	p = 0.0	p = 0.25	p = 0.5	p = 0.75	p = 1.0
F_1	1.05e-52 (0.00)	2.43e-75 (0.00)	4.08e-95 (0.00)	4.79e-114 (0.00)	2.85e-132 (0.00)
F_2	4.98e-32 (0.00)	7.22e-43 (0.00)	7.21e-53 (0.00)	3.78e-62 (0.00)	6.68e-71 (0.00)
F_3	4.95e+3 (1746.34)	7.57e+2 (347.29)	8.93e+1 (81.28)	6.55 (4.97)	1.11 (1.69)
F_4	3.26 (2.19)	7.94e-12 (0.00)	1.20e-16 (0.00)	4.78e-21 (0.00)	1.77e-25 (0.00)
F_5	2.35e+1 (1.12)	2.49e+1 (0.67)	2.57e+1 (0.34)	2.66e+1 (0.26)	2.83e+1 (0.20)
F_6	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
F_7	6.85e-3 (0.00)	2.57e-3 (0.00)	1.61e-3 (0.00)	1.44e-3 (0.00)	9.53e-4 (0.00)
F_8	9.48e+1 (98.72)	1.13e+2 (97.78)	1.01e+2 (79.45)	1.72e+2 (105.06)	7.00e+2 (225.13)
F_9	2.19e (1.05)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
F_{10}	7.11e-15 (0.00)	4.44e-15 (0.00)	3.91e-15 (0.00)	4.26e-15 (0.00)	4.26e-15 (0.00)
F_{11}	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
F_{12}	1.57e-32 (0.00)	1.57e-32 (0.00)	1.14e-29 (0.00)	2.68e-16 (0.00)	6.79e-3 (0.00)
F_{13}	1.36e-032 (0.00)	1.36e-032 (0.00)	5.49e-004 (0.00)	3.36e-015 (0.00)	1.73e-001 (0.08)
PEv	3.77e+003	3.52e+003	3.51e+003	3.53e+003	3.68e+003

Table 4: Performance of linde when setting NP = 30, p = 0.5, Fes = 3e+5

	Mean best (SD)						
Funs	Minimum	Median	Maximum	Mean	Standard dev.	Precision	Success rate
F_1	2.24e-265	1.01e-262	8.24e-261	9.19e-262	0.00e+000	1e-100	100%
F_2	7.31e-144	1.80e-142	6.85e-142	2.38e-142	2.60e-142	1e-100	100%
F_3	6.80e-001	2.43e+000	5.33e+000	2.47e+000	1.70e+000	1e-100	0.00%
F_4	3.05e-046	2.38e-045	1.32e-044	3.78e-045	4.03e-045	1e-100	0.00%
F_5	2.20e+001	2.30e+001	2.35e+001	2.28e+001	5.63e-001	1e-100	0.00%
F_6	0.00e+000	0.00e+000	0.00e+000	0.00e+000	0.00e+000	1e-100	100%
F_7	4.32e-004	7.69e-004	1.37e-003	7.96e-004	2.42e-004	1e-100	0.00
F_8	0.00e+000	5.92e+001	2.37e+002	8.29e+001	9.75e+001	1e-12	50%
F_9	0.00e+000	0.00e+000	0.00e+000	0.00e+000	0.00e+000	1e-100	100%
F_{10}	4.44e-015	4.44e-015	4.44e-015	4.44e-015	0.00e+000	1e-16	0.00%
F_{11}	0.00e+000	0.00e+000	0.00e+000	0.00e+000	0.00e+000	1e-100	100%
F_{12}	1.57e-032	1.57e-032	1.57e-032	1.57e-032	2.88e-048	1e-30	100%
F_{13}	1.35e-032	1.35e-032	1.47e-032	1.36e-032	3.90e-034	1e-30	100%
	-	-	-	-	-	-	53.84%

Table 5: Experimental results averaged over 25 independent

Funs	#FEs	SANSDE mean	SaDE mean	NSDE mean	LiNDE mean	Best optima
F_1	150 000	3.02E-23	7.49E-20	7.76E-16	5.23e-129	0.00
F_2	150 000	4.64E-11	6.22E-11	4.51E-10	1.09e-070	0.00
F_3	150 000	6.62E-22	1.12E-18	1.06E-14	1.96e+002	0.00
F_4	150 000	1.59E-03	2.96E-02	2.54E-02	5.15e-022	0.00
F_5	500 000	4.13E-30	2.10E+01	1.24E+01	2.49e+001	0.00
F_6	150 000	0.00	0.00	0.00	0.00	0.00
F_7	150 000	7.21E-003	7.58E-03	1.20E-02	1.55e-003	0.00
PEv(1-7)	-	1.54E-003	4.38	2.59	5.48e+001	0.00
F_8	150 000	0.00	0.00	0.00	6.51e+001	0.00
F_9	150 000	1.84E-05	4.00E-08	7.97E-02	0.00e+000	0.00
F_{10}	150 000	2.36E-12	9.06E-11	6.72E-09	4.44e-015	0.00
F_{11}	150 000	0.00	8.88E-18	4.68E-15	0.00	0.00
F_{12}	150 000	5.94E-23	5.21E-19	5.63E-17	1.57e-032	0.00
F_{13}	150 000	3.12E-22	1.75E-19	5.52E-16	1.36e-032	0.00
PEv(8-13)	-	3.84E-06	8.34E-09	1.66E-02	1.81e+001	0.00

also evaluate the performance of four algorithms using the proposed evaluation criterion. The average results of 25 independent runs are listed in Table 5.

High-dimensional unimodal functions (F1-F7). LiNDE algorithm outperformed SANSDE, SaDE and NSDE except functions F3, F5 and F6. The significant difference can be seen from the result on function F3. On the other hand, SaDE, NSDE and LiNDE performed almost the same for function F5 and F6, however, SANSDE obtained the highest precision. Finally, we evaluate the performance of four algorithms by proposed evaluation criterion. LiNDE obtained a higher score than that of others; the reason behind of the scores is function F3 suffering LiNDE.

High-dimensional multimodal functions with many local minima (F8-F13). Functions F8-F13 are all multimodal functions with many local minima and the number of local minima increases exponentially as the dimension increases. According to the experimental results, LiNDE performed better than other algorithms in term of mean values except function F8. However, SANSDE, SaDE and

NSDE all found the optima of the function. Consequently, the score of LiNDE is higher than that of other algorithm.

CONCLUSION

In this study we proposed a new variant of DE algorithm, LiNDE, aiming to improve searching ability of neighborhood. In LiNDE, (1) we generalized the control parameters F and CR, to whom DE algorithm is sensitive. In our study, F and CR are random numbers generated by uniform random distribution function; (2) we introduced a new operator for neighborhood search based on rand/1 operator and two operators had same the selection probability. The corresponding experimental results showed that LiNDE is non-sensitivity to the selection probability. Meanwhile, we proposed a novel evaluation criterion to evaluate the performance of different optimization algorithm. In order to select suitable control parameters and to evaluate the performance of proposed LiNDE algorithm, the algorithm was conducted on 13 classical benchmarks and LiNDE showed superiority over SANSDE, SaDE and NSDE for most benchmarks.

ACKNOWLEDGMENTS

This work was supported by National Natural Science Foundation of China under Grant 10871164 (Jan., 2009-Jan., 2011), 10676031 (Jan., 2007-Jan., 2010), 60673193 (Jan., 2007-Jan., 2010).

REFERENCES

Brest, J., S. Greiner, B. Boskovic, M. Mernik and V. Zumer, 2006. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Trans. Evolut. Comput., 10: 646-657.

Eberhart, R.C. and J. Kennedy, 1995. A new optimizer using particle swarm theory. Proceedings of the 6th International Symposium on Micro Machine and Human Science, Oct. 4-6, Nagoya, Japan, pp: 39-43.

Ilonen, J., J.K. Kamarainen and J. Lampinen, 2003. Differential evolution training algorithm for feed-forward neural networks. Neural Process. Lett., 7: 93-105.

Joshi, R. and A.C. Sanderson, 1999. Minimal representation multisensor fusion using differential evolution. IEEE Trans. System. Man Cybernet. A Syst. Humans, 29: 63-76.

Maruo, M.H., H.S. Lopes and M.R. Delgado, 2005. Self-adapting evolutionary parameters: Encoding aspects for combinatorial optimization problems. Lecture Notes Comput. Sci., 3448: 154-165.

Price, K.V., 1999. An Introduction to Differential Evolution. In: New Ideas in Optimization, Corne, D., M. Dorigo and F. Glover (Eds.). McGraw-Hill, London, UK., pp: 79–108.

Qin, A.K., V.L. Huang and P.N. Suganthan, 2009. Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evolut. Comput., 13: 398-417.

Rogalsky, T., R.W. Derksen and S. Kocabiyik, 1999. Differential evolution in aerodynamic optimization. Proceedings of the 46th Annual Conference of the Canadian Aeronautics and Space Institute, (CASI'99), Canada, pp: 29-36.

Storn, R. and K. Price, 1997. Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. J. Global Optimiz., 11: 341-359.

Storn, R., 1996. On the usage of differential evolution for function optimization. Proceedings of Biennial Conference of the North American Fuzzy Information Processing Society, Jun. 19-22, Berkeley, CA, pp: 519-523.

Yang, Z., J. He and X. Yao, 2008a. Making a Difference to Differential Evolution. In: Advances in Metaheuristics for Hard Optimization, Michalewicz, Z. and P. Siarry (Eds.). Springer, USA., pp: 397-414.

Yang, Z., K. Tang and X. Yao, 2008b. Self-adaptive differential evolution with neighborhood search. Proceedings of the IEEE World Congress on Evolutionary Computation, (CEC'08), Hong Kong, China, pp: 1110-1116.

Yao, X., Y. Liu and G. Lin, 1999. Evolutionary programming made faster. IEEE Trans. Evol. Comput., 3: 82-102.

Zaharie, D., 2002. Critical values for the control parameters of differential evolution algorithms. Proceedings of the 8th International Conference on Soft Computing Mendel 2002, June 2002, Brno, Czech Republic, pp: 62-67.