2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Effect of Pseudo Gradient on Differential Evolutionary for Global Numerical Optimization

Jinliang Ding, Lipeng Chen, Qingguang Xie, Tianyou Chai and Xiuping Zheng
the State Key Laboratory of Synthetical Automation for Process Industry
Northeastern University
Shenyang, China 110004

***Abstract*—In this paper, a novel pseudo gradient based DE approach is proposed, which takes advantage of both the differential evolutionary (DE) and the gradient-based algorithm. The gradient information, which is called pseudo gradient, is generated through randomly selected two vectors and their fitness function values. This work is to investigate the effect of proposed pseudo gradient on differential evolutionary algorithm. The simulation results show that DE with pseudo gradient can obtain better performance overall in comparison with classical DE variants. The pseudo gradient based DE with adaptive parameter section is compared with the existing adaptive DE algorithms. Also, the control parameter, step size are investigated to understand the mechanism of pseudo gradient in detail.**

I. Introduction

Differential algorithm has been shown to be an efficient evolutionary algorithm (EA) for many optimization problems [1], [2], [3]. It is easier to be implemented and has good convergence properties. The performance of DE mainly depends on mutation, crossover strategies and control parameters, including population size NP, scaling factor F, and crossover rate CR. Up to now, there is no fixed rule for choosing mutation strategies and control parameters to solve practical problems. To deal with this, adaptive or self-adaptive strategies on mutation strategies and control parameters have been utilised for years and showed excellent performance [3], [4], [5], [6], [7], [8], [9]. However, it is still necessary to improve DE to face challenges from various complex application areas.

Gradient based methods are widely used for continuously differentiable unconstrained optimization problems. These methods apply derivatives of functions to determine the search direction in order to find the local optima iteratively [10]. However, difficulties will arise when gradient methods encounter discontinuous problems in the real world. Meanwhile, fast descent often results in obtaining local optima and premature convergence easily.

In this paper, a novel approach to getting gradient information in DE is discussed. Without additional operation, such as function derivation or imposing perturbation, we select randomly two target vectors and their fitness function values to capture corresponding gradient information, which is called pseudo gradient. The ability of pseudo gradient to reflect right searching direction is investigated. Moreover, self-adaption is applied on the control parameters (F, CR, T), and compared with previous study.

The remainder of this paper is organized as follows. In section II, the basic operation of differential evolution algorithm is provided and the related work is presented in Section III. In Section IV the proposed pseudo gradient is demonstrated in detail. Experimental results on benchmark problems are showed in Section V, which shows the effectiveness of the proposed approach. Finally the paper is concluded in Section VI.

II. Basic Operations of DE

Differential Evolution (DE) searches for global optimum based on NP D-dimensional parameter vectors, i.e., $x_{i,G;i=1,2,...,NP}$, called target vector. To cover the entire parameter space, the initial vectors $x_{i,0} = (x_{1i,0}, x_{2i,0}, ..., x_{Di,0})$, are generated randomly by

$$x_{ji,0} = x_{min} + rand(0,1) \cdot (x_{max} - x_{min}) \tag{1}$$

where $j = 1, 2, ..., D$ and $i = 1, 2, ..., NP$. x_{min} is the lower bound and x_{max} is the upper bound for the parameter space. $rand(0,1)$ is an uniformly distributed random number in range [0,1].

A. Mutation

DE generates a mutant vector $v_{i,G+1}$ for each target vector $x_{i,G;i=1,2,3,...,NP}$, based on the current parent populations. The classical mutation strategies are

1) "DE/rand/1"

$$v_{i,G+1} = x_{r1,G} + F \cdot (x_{r2,G} - x_{r3,G}) \tag{2}$$

2) "DE/best/1"

$$v_{i,G+1} = x_{best,G} + F \cdot (x_{r1,G} - x_{r2,G}) \tag{3}$$

3) "DE/current-best/1"

$$v_{i,G+1} = x_{i,G} + F \cdot (x_{best,G} - x_{i,G}) + F \cdot (x_{r1,G} - x_{r2,G}) \tag{4}$$

4) "DE/best/2"

$$v_{i,G+1} = x_{best,G} + F \cdot (x_{r1,G} - x_{r2,G}) + F \cdot (x_{r3,G} - x_{r4,G}) \tag{5}$$

5) "DE/rand/2"

$$v_{i,G+1} = x_{r1,G} + F \cdot (x_{r2,G} - x_{r3,G}) + F \cdot (x_{r4,G} - x_{r5,G}) \tag{6}$$

with $F > 0$ and random integer indexes $r1, r2, r3, r4, r5 \in 1, 2, ..., NP$, which are also different from the index i.

B. Crossover

A binomial crossover is defined as follows to get the trial vector $u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, ..., u_{Di,G+1})$, where

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & \text{if } rand_j(0,1) < CR \text{ or } j = j_{rand} \\ x_{ji,G} & \text{otherwise} \end{cases} \quad (7)$$

in which $j = 1, 2, ..., D$ and $rand_j$ is a uniform random number $\in [0, 1]$. $CR \in [0, 1]$ is the crossover parameter. $j_{rand} \in 1, 2, ..., D$ is a randomly chosen integer to ensure that at least one parameter is transfered from $v_{i,G+1}$ to $u_{i,G+1}$.

C. Selection

From trial vector and target vector, the better vector producing less fitness function value is chosen as new target vector in next generation, expressed as follow:

$$x_{i,G+1} = \begin{cases} u_{i,G+1} & \text{if } f(u_{i,G+1}) < f(x_{i,G}) \\ x_{i,G} & \text{otherwise} \end{cases} \quad (8)$$

where f is the fitness function to be minimized.

III. Related Works

The work of DE optimization mostly focuses on mutation strategies, crossover schemes and control parameters.

A. Adaptive DE Algorithms

Liu and Lampinen [3] proposed the fuzzy adaptive differential evolution (FADE) algorithm, in which fuzzy logic is used to adapt the control parameter for mutation strategy and crossover scheme. The paper reports that FADE performs better than traditional one on 6 benchmark functions overall.

Another self-adaptive DE algorithm (jDE) has been implemented by stochastically refreshing the control factor F and CR for each individual. The formular for generating new control parameters are showed as follows:

$$F_{i,G+1} = \begin{cases} F_l + rand_1 * F_u & \text{if } rand_2 < \tau_1 \\ F_{i,G} & \text{otherwise} \end{cases} \quad (9)$$

$$CR_{i,G+1} = \begin{cases} rand_3 & \text{if } rand_4 < \tau_2 \\ CR_{i,G} & \text{otherwise} \end{cases} \quad (10)$$

in which F_l, F_u, τ_1, τ_2 are fixed values. The better values of control parameters tend to generate better offsprings which are more likely to survive. Hence, these values should be propagated to next generation. It is showed that jDE does not need to guess good value for control parameters and can achieve better performance [4].

Qin and Saganthan [11] developed self-adaptive DE algorithm (SaDE) with adaption both on mutation strategy and control parameters. In their version, strategy candicate pool includes two mutation strategies $DE/rand/1/bin$ and $DE/rand - to - best/2/bin$. The trial vector generation strategy is selected according to the probability according to its success rate in generating better fitness function values within previous 50 generations. SaDE is further improved by incorporating two additional mutation strategy $DE/rand/2/bin$, $DE/current - to - rand/1$ and by adapting CRm for each trial vector generation strategy [9].

Adaptive DE with optional external archive (JADE) is presented in [8]. JADE applies new mutation strategy $DE/current - to - pbest$ with optional external archive as follows:

$$v_{i,G} = x_{i,G} + F_i \cdot (x^p_{best,g} - x_{i,g}) + F_i \cdot (x_{r1,g} - \bar{x}_{r2,g}) \quad (11)$$

where x^p_{best} is one of the top 100p% target vectors in the current population, $\bar{x}_{r2,g}$ is selected from the union of current population and the archive. The author indicates that this mutation strategy can improve the diversity of the population and avoid premature convergence. Experimental results suggest that JADE is better than or at least comparable to classical DE and other adaptive DE variants on benchmark functions. Especially on high dimensional problems, JADE with archive shows promising performance.

Recently three algorithmic components have been embedded in classical DE to improve its search performance [12]. The authors proposed the $DE/current - to - gr\_best/1$ mutation strategy,

$$v_{i,G} = x_{i,G} + F_i \cdot (x_{gr\_best,g} - x_{i,g} + x_{r1,g} - x_{r2,g}) \quad (12)$$

where $x_{gr\_best,g}$ is the best of the $q\%$ randomly selected vectors from the current population. A so called $p-$best crossover operation is employed in MDE_pBX. The new trial vector is constructed from randomly selected $p-$best vector and donor vector based on normal binomial crossover type. To update F and CR, Chauchy distribution and Gaussian distribution are adopted, respectively.

B. Improving EA with gradient approach

Hybrid EA including DE algorithm based on gradient is developed for various problems, such as real-valued unimodal, multimodel, multiobjective, constrained and dynamic optimization.

Salomon discussed the obvious similarities and differences between classical gradient methods and evolutionary algorithms in [13]. It is argued that evolutionary algorithm has a high probability to jump out of the local optima.

Chiou and Wang [14] introduced acceleration phase based on gradient in DE algorithm to solve static and dynamic problems. The corresponding gradient in the acceleration phase is obtained from finite difference and the step size $\alpha \in [0, 1]$. To overcome the drawback of premature convergence in acceleration phase, a migration phase is performed to regenerate a new population through adding a vector of independent random Gaussian numbers.

Pseudo-gradient in evolutionary algorithm is proposed in [15]. It chooses a test candicate x_l and compares the objective function value of this test vector with another point x_k. If $f(x_l) < f(x_k)$, the gradient is defined as $g(x_l) = dir(x_l)$, where

$$dir(x_{l,i}) = \begin{cases} 1 & \text{if } x_{l,i} > x_{k,i} \\ 0 & \text{if } x_{l,i} = x_{k,i} \\ -1 & \text{if } x_{l,i} < x_{k,i} \end{cases} \quad (13)$$

if $f(x_l) > f(x_k)$, $g(x_l) = 0$. Hence, it is not necessary for the objective functions to be differentiable. In addition, this approach makes use of the directions which lead the individual to better fitness and neglects the direction which is wrong.

The ε constrained DE method with gradient-based mutation is proposed in [16], which constructs gradients from both equality and non-zero equality constraints. It is showed this algorithm is efficient to solve 23 problems out of 24 benchmark problems.

For multiobjective problems, hybrid multiobjective evolutionary algorithms with a gradient-based operator is presented to solve the production planning optimization problem for mineral processing [17]. Gradients are obtained from differential of continuous objective functions.

IV. Pseudo gradient

In this section, we propose a new DE algorithm, PGDE, which implements gradient approach in mutation strategy. In mutation part of DE, we choose two target vectors and their corresponding fitness function values to generate, so called pseuso gradient, to distinguish from general gradient. The formular is as follows:

$$\nabla f = \frac{f(x_{r1}) - f(x_{r2})}{x_{r1} - x_{r2}} \tag{14}$$

in which x_{r1}, x_{r2} are randomly selected different vectors, $f(x_{r1}), f(x_{r2})$ are the values of fitness function corresponding to x_{r1}, x_{r2} respectively. The difference vectors in classical mutation strategy are random and store no information about searching direction. But pseudo gradient with fitness values store corresponding searching information. We replace the difference vector in classical mutation strategy by pseudo gradient. For example, we change $DE/rand/1/bin$ to below formular, naming $PGDE/rand/1/bin$.

$$v_{i,G+1} = x_{r1,G} + T \times \nabla f \tag{15}$$

where T is step size which behaves similarly with scale factor F in classical mutation strategy. One decaying formular for T is employed as

$$T = DT \times \frac{G_{max} - G_{current} + 1}{G_{max}} \tag{16}$$

where G_{max} is the maximal number of iterations in simulation allowed, $G_{current}$ is the current number of iterations in running and DT is a control parameter.

It is obvious that the ∇f may be zero when $f(x_{r1})$ equals to $f(x_{r2})$. Another problem arises when one or more parameters in x_1, x_2 are equal. The gradient will be not a number, called "NAN". So we set one switch parameter in mutation. In running if parameter in pseudo gradient is zero or NAN, then mutation strategy will be switched to corresponding classical mutation strategy, which is $DE/rand/1/bin$ in this case. Such switch has another advantage that algorithm based on gradient approach becomes easier to escape the local optima area. Meanwhle, a factor $F_{ration} \in (0, 1)$ is set to decide choosing pseudo gradient mutation strategy or classical one to improve the exploration of gradient approach. The pseudo code of $PGDE/rand/1/bin$ is showed as Table I.

Then $DE/best/1/bin$ is employed into the mutation strategy in PGDE, so the corresponding $PGDE/best/1/bin$ mutation strategy is shown as Table II.

TABLE I: Pseudo Code of PGDE

	Procedure of PGDE
1	Initialization
	$x_{i,0} = x_{min} + rand_i \times (x_{max} - x_{min})$
2	Set control parameter
	F, CR, DT, F_{ration}
3	Decide switch value and calculate pseudo gradient when necessary
	Select randomly vector x_{r1}, x_{r2}, x_{r3}
	if $rand(1) < F_{ration}$ set $I_{switch} = 0$
	else $I_{switch} = 1$
	if $I_{switch} = 0$,Calculate corresponding gradient
	$\nabla f = \frac{f(x_{r1}) - f(x_{r2})}{x_{r1} - x_{r2}}$
	if ∇f is NAN or 0, $I_{switch} = 1$
4	Mutation
	if $I_{switch} = 1$, then classical mutation strategy
	$v_{i,G+1} = x_{r1,G} + F \times (x_{r2,G} - x_{r3,G})$
	else
	$v_{i,G+1} = x_{r1,G} - DT \times \frac{G_{max} - G_{current} + 1}{G_{max}} \times \nabla f$
6	Crossover
	$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & \text{if } rand_j(0,1) < CR \text{ or } j = j_{rand} \\ x_{ji,G} & \text{otherwise} \end{cases}$
7	Selection
	$x_{i,G+1} = \begin{cases} u_{i,G+1} & \text{if } f(u_{i,G+1}) < f(x_{i,G}) \\ x_{i,G} & \text{otherwise} \end{cases}$

TABLE II: Mutation strategy of PGDE/best

if $I_{switch} = 1$ then classical mutation strategy
$v_{i,G+1} = x_{best,G} + F \times (x_{r2,G} - x_{r3,G})$
else
$v_{i,G+1} = x_{best,G} - DT \times \frac{G_{max} - G_{current} + 1}{G_{max}} \times \nabla f$

Also we employ the adaption on F, CR, DT in PGDE. The formulars of adaption for these control parameters are similar with those in jDE as follows,

$$F_{i,G+1} = \begin{cases} F_l + rand_1 * F_u & \text{if } rand_2 < \tau_1 \\ F_{i,G} & \text{otherwise} \end{cases} \tag{17}$$

$$CR_{i,G+1} = \begin{cases} rand_3 & \text{if } rand_4 < \tau_2 \\ CR_{i,G} & \text{otherwise} \end{cases} \tag{18}$$

$$DT_{i,G+1} = \begin{cases} DT_l + rand_1 * DT_u & \text{if } rand_2 < \tau_3 \\ DT_{i,G} & \text{otherwise} \end{cases} \tag{19}$$

in which $F_l, F_u, DT_l, DT_u, \tau_1, \tau_2, \tau_3$ are fixed values. We call it PGjDE.

V. Results

We tested PGDE on 25 benchmark functions defined in the CEC 2005 special session and real-parameter optimization[18]. These functoins cover a large range of real world problems, such as multimodality, ill-conditioning, inseparability and rotation. These 25 test functions can be divided into four groups:

1) unimodal functions $f1 - f5$
2) basic multimodal functions $f6 - f12$
3) expand multimodal functions $f13 - f14$
4) hybrid composition functions $f15 - f25$

PGDE was compared with classical DE variants and adaptive algorithms. For all algorithms, the dimension D was set

to 30 and the population size was kept equal to 50. All experiments were run 25 times independently. The same initial population were used to make sure the performance was only based on the algorithm itself only. The mean and standard deviation of the function error value $f(x)-f(x^*)$ were stored for measuring the performance of every algorithm, where x was the best individual found by the algorithm in one run and x^* is the global optimum of the test functions. The control parameters for each algorithm were chosen as in their original publications and listed as follows:

1) • DE/rand/1/bin with $F = 0.9$, $CR = 0.9$;
   • PGDE/rand/1/bin with $F = 0.9$, $CR = 0.9$, $DT = 0.05$.
2) • DE/best/1/bin with $F = 0.9$, $CR = 0.9$, $DT = 0.05$;
   • PGDE/best/1/bin with $F = 0.9$, $CR = 0.9$, $DT = 0.05$.
3) • JADE with $c = 1, p = 0.05$, and optional external archive[8];
   • jDE with $F_l = 0.1$, $F_u = 0.9$, $\tau_1 = \tau_2 = 0.1$[4];
   • SaDE [9];
   • MDE_pBX [12];
   • PGjDE with $F_l = 0.1$, $F_u = 0.9$, $DT_l = 0.001$, $DT_u = 0.999$, and $\tau_1 = \tau_2 = \tau_3 = 0.1$.

A. Comparision with classical DE algorithms

Table III shows the mean and standard deviation for test functions obtained by $DE/rand/1/bin$ and $PGDE/rand/1/bin$. The better results are typed in bold.

It is obvious that $PGDE/rand/1/bin$ achieves better performance for 22 functions out of 25 functions. For unimodal functions $f1 - f5$, PGDE has faster convergence speed, which represents that pseudo gradient has the ability to lead individuals to right direction towards gloabal optimum. Hybrid composition functions have a huge number of local optima, so it's difficult for DE operator to find the global optima. Here we observe that for most hybrid compositon functions, PGDE achieves lower fitness function values overall, which suggests that the pseudo gradient migrated with normal mutation operator enhances the diversity of the population.

Mean and standard deviation from $DE/best/1/bin$ and $PGDE/best/1/bin$ are compared in Table IV. The $PGDE/best/1/bin$ is slightly better than $DE/best/1/bin$, with 16 and 9 better results, respectively. For unimodal function $f1-f5$, $PGDE/best/1/bin$ achieves better performance. Especially for $f1, f2$, $PGDE$ improves the solution by more than 20 magnitude in comparision with classical DE variant.

For these four group test functions, the performance of $PGDE/best$ is similar with $DE/best$ to some extent. But compared with $DE/rand$, $PGDE/rand$ outperforms almost in all test functions. $DE/rand$ usually demonstrates stronger exploration capacity but with very slow convergence speed. The $DE/best$ strategy has fast onvergence speed but is more likely to drop in local optimum, as well as the pseudo gradient. So compared with $DE/best$, the drawback at exploration in $PGDE/best$ is not so well improved. This suggests us that a diverse operator, such as $DE/rand$, is more suitable for PGDE. A greedy mutation strategy can not help PGDE to find global optimum so much.

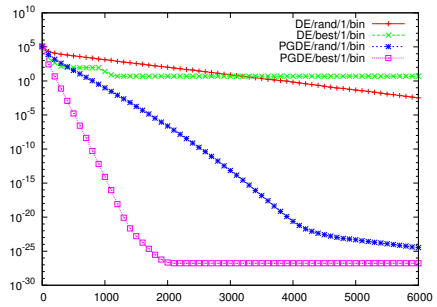
TABLE III: Results of DE/rand/1/bin and PGDE/rand/1/bin

	DE/rand/1 Mean(Std)	PGDE Mean(Std)
f1	3.1504e-03 (2.5905e-03)	**3.3695e-25 (1.4917e-25)**
f2	3.8247e+02 (3.4840e+02)	**3.2253e-05 (2.7187e-05)**
f3	6.0577e+06 (2.7770e+06)	**5.1552e+05 (2.9022e+05)**
f4	1.9425e+03 (1.1291e+03)	**1.5046e-01 (1.8837e-01)**
f5	8.7800e+02 (3.1192e+02)	**4.3144e+02 (2.4811e+02)**
f6	**6.1159e+01 (4.3896e+01)**	1.2193e+02 (1.0457e+02)
f7	9.8685e-01 (5.7216e-02)	**2.4491e-02 (2.3381e-02)**
f8	2.0954e+01 (3.8102e-02)	**2.0190e+01 (5.3467e-02)**
f9	4.0538e+01 (1.9449e+01)	**1.4574e+01 (3.7175e+00)**
f10	2.2325e+02 (1.9636e+01)	**3.7059e+01 (1.3151e+01)**
f11	3.9667e+01 (1.3115e+00)	**1.8575e+01 (2.8601e+00)**
f12	**4.8309e+03 (4.4040e+03)**	8.0804e+03 (1.1108e+04)
f13	1.1431e+01 (3.1793e+00)	**3.5494e+00 (6.6556e-01)**
f14	1.3351e+01 (1.6539e-01)	**1.3237e+01 (3.3412e-01)**
f15	**3.1378e+02 (1.2206e+02)**	3.7022e+02 (1.1111e+02)
f16	2.7104e+02 (5.4446e+01)	**7.0941e+01 (2.4790e+01)**
f17	3.0175e+02 (4.8966e+01)	**1.0933e+02 (3.3507e+01)**
f18	9.0513e+02 (1.0461e+00)	**9.0408e+02 (3.0238e-01)**
f19	9.0564e+02 (9.4354e-01)	**9.0415e+02 (8.4920e-01)**
f20	9.0544e+02 (1.0128e+00)	**9.0410e+02 (6.6735e-01)**
f21	5.0000e+02 (2.7211e-04)	**5.0000e+02 (1.6583e-05)**
f22	8.9640e+02 (1.2548e+01)	**8.9410e+02 (1.4225e+01)**
f23	5.7880e+02 (1.5448e+02)	**5.5028e+02 (8.0567e+01)**
f24	3.2279e+02 (2.8714e+02)	**2.0000e+02 (1.5264e-04)**
f25	2.1245e+02 (1.2176e+00)	**2.1122e+02 (6.2671e-01)**
+	3	22

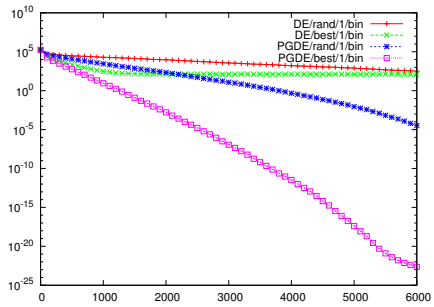
TABLE IV: Results of DE/best/1/bin and PGDE/best/1/bin

	DE/best/1 Mean(Std)	PGDE Mean(Std)
f1	4.9159e+00 (2.4579e+01)	**1.6841e-27 (1.2569e-27)**
f2	1.3190e+02 (6.4589e+02)	**2.3112e-23 (2.8864e-23)**
f3	**2.9761e+05 (2.6411e+05)**	6.2329e+05 (6.5475e+05)
f4	5.3745e+02 (2.0747e+03)	**2.2998e-04 (3.2503e-04)**
f5	7.9126e+02 (1.3524e+03)	**7.1605e+02 (1.0707e+03)**
f6	3.8230e+06 (1.9115e+07)	**1.0050e+05 (5.0246e+05)**
f7	**1.5023e-02 (2.0639e-02)**	1.9971e-02 (1.5017e-02)
f8	2.0944e+01 (6.3513e-02)	**2.0009e+01 (6.9385e-03)**
f9	5.5903e+01 (1.7997e+01)	**1.5918e+01 (5.4719e+00)**
f10	**6.0692e+01 (1.8240e+01)**	8.4452e+01 (2.8520e+01)
f11	**1.3856e+01 (3.0597e+00)**	1.6173e+01 (2.6337e+00)
f12	2.4010e+04 (2.6936e+04)	**1.3794e+04 (1.7756e+04)**
f13	4.1760e+00 (1.5109e+00)	**3.7896e+00 (1.2315e+00)**
f14	**1.2469e+01 (6.6538e-01)**	1.3788e+01 (3.0120e-01)
f15	3.5877e+02 (7.9901e+01)	**3.5060e+02 (6.4145e+01)**
f16	**3.7854e+02 (1.3547e+02)**	4.4602e+02 (1.1074e+02)
f17	3.2496e+02 (1.7156e+02)	**3.1735e+02 (1.6146e+02)**
f18	**9.0673e+02 (1.9906e+00)**	9.0807e+02 (4.7449e+00)
f19	9.0660e+02 (1.8944e+00)	**9.0624e+02 (3.1766e+00)**
f20	**9.0734e+02 (4.1633e+00)**	9.0768e+02 (3.8427e+00)
f21	6.6686e+02 (2.3391e+02)	**5.7195e+02 (1.7188e+02)**
f22	8.7985e+02 (3.4838e+01)	**8.7955e+02 (2.2637e+01)**
f23	6.8477e+02 (2.0698e+02)	**5.8417e+02 (1.3609e+02)**
f24	6.5038e+02 (3.7010e+02)	**2.3049e+02 (1.5244e+02)**
f25	**2.1146e+02 (1.0515e+00)**	2.1515e+02 (1.6653e+00)
+	9	16

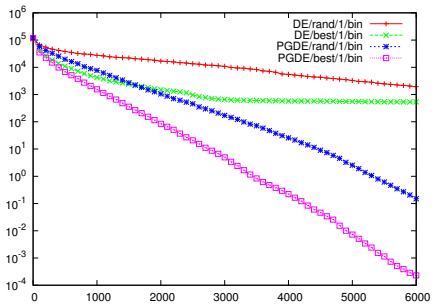
Figure 1 shows the median convergence characteristics of above 4 DE variants. It shows that classical DE operator $DE/rand/1/bin$ converges too slowly for unimodal functions



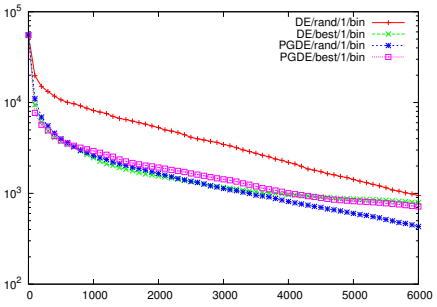
(a) f1



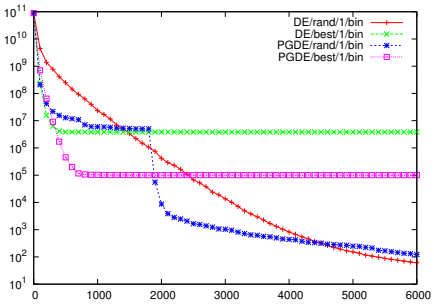
(b) f2



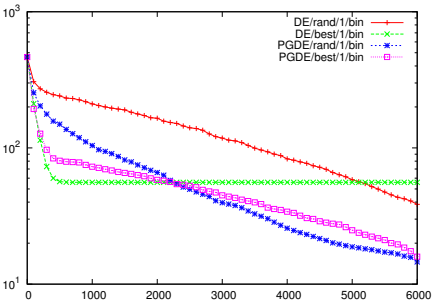
(c) f4



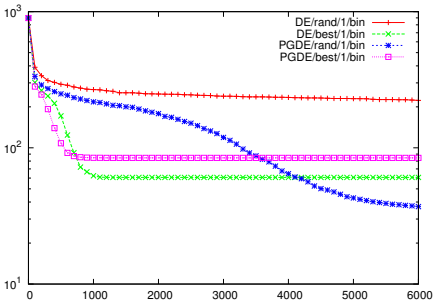
(d) f5



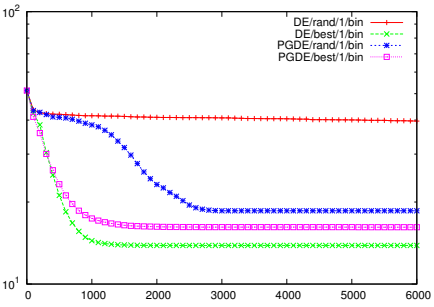
(e) f6



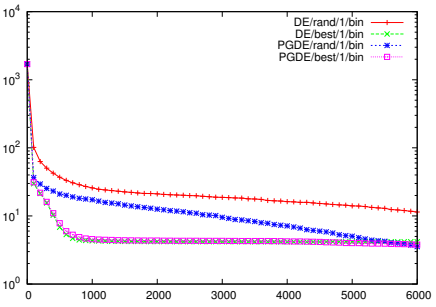
(f) f9



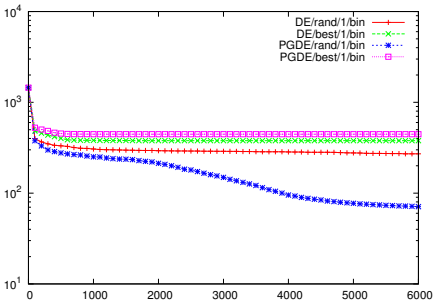
(g) f10



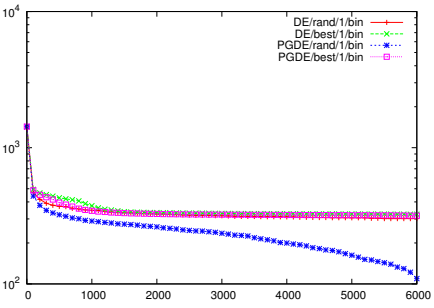
(h) f11



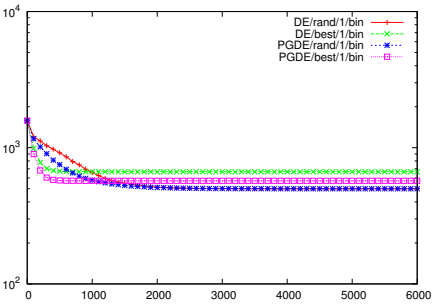
(i) f13



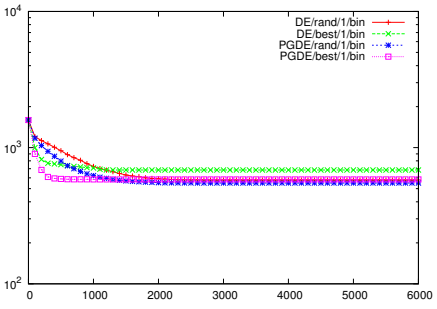
(j) f16



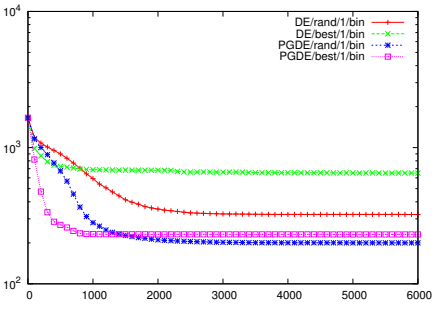
(k) f17



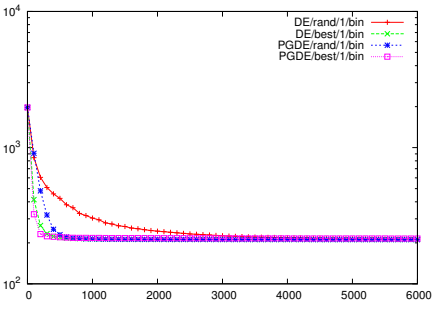
(l) f21



(m) f23



(n) f24



(o) f25

Fig. 1: Median convergence characteristics of DE/rand/1, DE/best/1, PGDE/rand/1 and PGDE/best/1

$f1, f2, f4$. Instead, $DE/best/1/bin$ converges too early. In contrast, both PGDE performs well with consideration about convergence speed and searching ability of gloabal optimum.

For multimodal functions, $PGDE/rand/1/bin$ speed up the convergence compared with $DE/rand/1/bin$. It is obvious that for $f6$, in which there is a very narrow valley from local optimum to global optimum, the PGDE operation jumps out the local optimum. This demonstrates that PGDE has the ability to jump out local optimum. Also for composition functions, which have huge number of local optimums, PGDE performs better overall. This suggests that PGDE is more suitable for optimization of complex problems with many local optimums. Moreover, for some test functions $DE/rand/1/bin$ and $DE/best/1/bin$ perform better than corresponding PGDE variants. This reflects the capacity of pseudo gradient to express the right searching direction for optimization has some limitations to some extent, as the pseudo gradient may reflect wrong searching direction if there are vallies between two random points.

B. Comparision with adaptive DE algorithms

The performance of DE is very sensitive to the choice of the control parameters the control parameters (F, CR). In this section, an adaption operator on F,CR,DT is introduced into PGDE, denoted as PGjDE. According to last section, we know that $DE/rand/1/bin$ is more suitable for PGDE. Meanwhile, in most adaptive DE algorithms, jDE adopt $DE/rand/1/bin$ mutation strategy. So the adaption strategy of the parameter F,CR chosen in this papar is the same as in jDE. PGjDE was compared with other adaptive DE variants, including jDE [4], SaDE [9], JADE [8] and MDE_pBX [12].

Table V shows that PGjDE is better than jDE and SaDE, equal to JADE, and slightly worse than MDE_pBX according to overall performance. For the unimodal function $f1$, JADE, SaDE and PGjDE find the global optimum. For function $f2, f3, f4$, JADE is the best one. PGjDE obtains the best results for function $f5$. For the multimodal functions, only JADE and PGjDE find the global optimum for function $f9$. Meanwhile, JADE is better for almost all the multimodal functions, except for $f8$. JADE achieves better performances both in $f13, f14$, comparied with the other adaptive DE variants. In composition function group, for the function $f18, f19, f20$, MDE_pBX is the best one, followed by SaDE and PGjDE. PGjDE sits at the second place for function $f17, f25$. For function $f15, f16, f21, f23, f24$, the best results are obtained by PGjDE.

It is obvious in Table V that PGjDE is better than jDE, especially on composition functions, which suggests again that the pseudo gradient operator improves the searching capacity of DE algorithm and is more suitable for optimization of complex problems with many local optimums. Table VI gives imformation about the performances of $PGDE/rand/1/bin$ and $PGjDE/rand/1/bin$. PGjDE achieves much better performance than PGDE, which demonstrates that the adaption strategies of the control parameters enhance the exploration ability of PGDE to some extent.

TABLE VI: Results of PGDE/rand/ and PGjDE/rand/

	PGDE Mean(Std)	PGjDE Mean(Std)
f1	3.3695e-25 (1.4917e-25)	**0.0000e+00 (0.0000e+00)**
f2	3.2253e-05 (2.7187e-05)	**1.0178e-05 (7.7833e-06)**
f3	5.1552e+05 (2.9022e+05)	**4.0186e+05 (2.3273e+05)**
f4	**1.5046e-01 (1.8837e-01)**	4.0092e+05 (2.4636e+05)
f5	4.3144e+02 (2.4811e+02)	**2.2466e-01 (3.0454e-01)**
f6	1.2193e+02 (1.0457e+02)	**3.2865e+01 (2.8541e+01)**
f7	2.4491e-02 (2.3381e-02)	**1.8897e-02 (1.1757e-02)**
f8	2.0190e+01 (5.3467e-02)	**2.0026e+01 (1.5544e-02)**
f9	1.4574e+01 (3.7175e+00)	**0.0000e+00 (0.0000e+00)**
f10	**3.7059e+01 (1.3151e+01)**	4.7554e+01 (9.9144e+00)
f11	**1.8575e+01 (2.8601e+00)**	2.9247e+01 (1.6473e+00)
f12	**8.0804e+03 (1.1108e+04)**	1.7727e+04 (5.0032e+03)
f13	3.5494e+00 (6.6556e-01)	**1.3674e+00 (1.0973e-01)**
f14	1.3237e+01 (3.3412e-01)	**1.2900e+01 (4.6891e-01)**
f15	3.7022e+02 (1.1111e+02)	**2.2800e+02 (9.3629e+01)**
f16	**7.0941e+01 (2.4790e+01)**	1.0210e+02 (7.3737e+01)
f17	**1.0933e+02 (3.3507e+01)**	1.4702e+02 (7.3364e+01)
f18	9.0408e+02 (3.0238e-01)	**9.0391e+02 (1.1700e+00)**
f19	9.0415e+02 (8.4920e-01)	**9.0286e+02 (2.4005e+00)**
f20	9.0410e+02 (6.6735e-01)	**9.0308e+02 (2.6515e+00)**
f21	5.0000e+02 (1.6583e-05)	**5.0000e+02 (8.9877e-14)**
f22	8.9410e+02 (1.4225e+01)	**8.8260e+02 (1.7051e+01)**
f23	5.5028e+02 (8.0567e+01)	**5.3416e+02 (3.4676e-04)**
f24	2.0000e+02 (1.5264e-04)	**2.0000e+02 (2.9008e-14)**
f25	**2.1122e+02 (6.2671e-01)**	2.1134e+02 (9.5884e-01)
+	7	18

C. Timestep

In PGDE, the control parameters are F,CR,T. T consists of two parts: constant DT and the decreasing part as

$$\frac{G_{max} - G_{current} + 1}{G_{max}} \tag{20}$$

The effect of step size DT on the performance of PGDE is studied in this section. Three step size are chosen as $0.5, 0.05, 0.005$, repectively. 25 independent runs are made on 25 test functions in 30 dimensions.

Tabel VII reports the experimental results for different step size. For most functions, smaller step sizes achieve better results overall. Especially for functions $f1, f2, f4, f8, f11, f19, f20, f22, f24$, smaller is the step size, better is the performance. However, this is just in the range $0.005 - 0.5$, there is still lack of results on more step sizes. Due to the high computational cost, we didn't investigate it in more detail.

VI. Conclusion

The experimental results above demonstrate that the pseudo gradient can improve the searching capacity and convergence speed of DE algorithm. To some extent, the pseudo gradient could reflect the right searching direction to the global optima, for it takes advantage of the objective function value of target vector. In addition, it is much simpler to implement compared with other hybrid evolutionary algorithm variants based on

TABLE V: Results of Adaptive DE Variants and PGDE

	JADE Mean(Std)	jDE Mean(Std)	SaDE Mean(Std)	MDEpBX Mean(Std)	PGjDE Mean(Std)
f1	0.0000e+00 (0.0000e+00) =	2.0195e-30 (1.0097e-29) −	0.0000e+00 (0.0000e+00)=	3.8654e-14(2.6516e-14)−	0.0000e+00 (0.0000e+00)
f2	5.7032e-28 (2.5723e-28) +	1.0525e-10 (2.6553e-10) +	705042e-06 (1.3994e-05)+	2.0464e-13(7.3677e-14)+	1.0178e-05 (7.7833e-06)
f3	4.3241e+03 (2.8661e+03) +	1.2453e+05 (7.7210e+04) +	4.4039e+05 (1.5004e+05)−	2.7951e+04(1.7481e+04)+	4.0186e+05 (2.3273e+05)
f4	2.7300e-06 (6.8744e-06) +	5.4116e-01 (2.1424e+00) +	1.1672e+02 (1.3563e+02)+	2.8870e-04(5.3276e-04)+	4.0092e+05 (2.4636e+05)
f5	8.0595e+00 (2.0554e+01) −	1.1254e+03 (6.4164e+02) −	3.4119e+03 (6.5088e+02)−	9.4952e+02(5.0235e+02)−	2.2466e-01 (3.0454e-01)
f6	6.5783e+00 (2.2252e+01) +	8.9871e-01 (1.6257e+00) +	4.7039e+01 (3.1083e+01)−	1.4352e+00(1.9136e+00)+	3.2865e+01 (2.8541e+01)
f7	1.2999e-02 (1.0750e-02) +	1.9189e-02 (1.2645e-02) −	1.5549e-02 (1.3844e-02)+	1.8989e-02(1.3929e-02)−	1.8897e-02 (1.1757e-02)
f8	2.0859e+01 (2.6173e-01) −	2.0935e+01 (5.4696e-02) −	2.0946e+01 (6.1664e-02)−	2.0000e+01(1.0836e-05)+	2.0026e+01 (1.5544e-02)
f9	0.0000e+00 (0.0000e+00) =	3.9798e-02 (1.9899e-01) −	1.1940e-01 (3.2999e-01)−	2.4306e+01(6.3840e+00)−	0.0000e+00 (0.0000e+00)
f10	2.8152e+01 (7.4952e+00) +	4.2160e+01 (8.2429e+00) +	4.6524e+01 (8.9881e+00)+	4.1537e+01(1.0547e+01)+	4.7554e+01 (9.9144e+00)
f11	2.6619e+01 (1.9175e+00) +	2.3280e+01 (5.5476e+00) +	1.7267e+01 (3.2715e+00)+	2.4065e+01(5.1563e+00)+	2.9247e+01 (1.6473e+00)
f12	5.2917e+03 (5.2179e+03) +	4.2452e+03 (4.4603e+03) +	2.5364e+03 (2.1263e+03)+	2.2442e+03(2.9947e+03)+	1.7727e+04 (5.0032e+03)
f13	1.2030e+00 (1.1086e-01) +	1.2382e+00 (1.2995e-01) +	3.9040e+00 (3.2194e-01)−	4.1902e+00(1.1646e+00)−	1.3674e+00 (1.0973e-01)
f14	1.2312e+01 (3.1576e-01) +	1.2939e+01 (2.1952e-01) −	1.2711e+01 (2.3522e-01)+	1.2974e+01(5.1086e-01)−	1.2900e+01 (4.6891e-01)
f15	2.9310e+02 (5.2091e+01) −	3.0800e+02 (1.1150e+02) −	3.6423e+02 (5.7016e+01)−	2.9908e+02(7.8990e+01)−	2.2800e+02 (9.3629e+01)
f16	3.0624e+02 (1.5846e+02) −	1.3875e+02 (1.2307e+02) −	1.4766e+02 (1.2944e+02)−	2.3341e+02(1.5632e+02)−	1.0210e+02 (7.3737e+01)
f17	2.4125e+02 (1.4873e+02) −	1.6417e+02 (1.0381e+02) −	8.4245e+01 (3.5212e+01)+	2.4576e+02(1.6763e+02)−	1.4702e+02 (7.3364e+01)
f18	9.0549e+02 (1.3428e+00) −	9.0525e+02 (1.5861e+00) −	8.8442e+02 (5.9290e+01)+	8.1761e+02(2.3823e+00)+	9.0391e+02 (1.1700e+00)
f19	9.0517e+02 (1.0972e+00) −	9.0513e+02 (1.2322e+00) −	8.9198e+02 (5.3022e+01)+	8.1736e+02(1.0467e+00)+	9.0286e+02 (2.4005e+00)
f20	9.0536e+02 (1.2991e+00) −	9.0536e+02 (1.2501e+00) −	8.9301e+02 (5.3753e+01)+	8.1621e+02(1.2008e-01)+	9.0308e+02 (2.6515e+00)
f21	5.1600e+02 (8.0000e+01) −	5.0000e+02 (9.3547e-14) −	6.4714e+02 (2.7146e+02)−	6.2121e+02(2.2632e+02)−	5.0000e+02 (8.9877e-14)
f22	8.6795e+02 (1.9264e+01) +	8.7695e+02 (1.8165e+01) +	9.3634e+02 (2.0121e+01)−	5.0009e+02(2.8583e-01)+	8.8260e+02 (1.7051e+01)
f23	5.4969e+02 (7.6948e+01) −	5.3416e+02 (3.8845e-04) =	5.3417e+02 (5.4817e-03)−	5.7514e+02(1.0971e+02)−	5.3416e+02 (3.4676e-04)
f24	2.0000e+02 (2.9008e-14) =	2.0000e+02 (2.9008e-14) =	2.0000e+02 (2.9008e-14)=	2.0831e+02(4.2255e+00)−	2.0000e+02 (2.9008e-14)
f25	2.1315e+02 (2.7725e+00) −	2.1195e+02 (1.6222e+00) −	2.1252e+02 (1.3979e+00)−	2.1043e+02(7.9917e-01)+	2.1134e+02 (9.5884e-01)
+	11	9	11	13	
−	15	14	12	12	
=	3	1	2	0	

gradient. Parameter step size also plays an important role in PGDE, but to understand its mechanism still needs more work.

Based on the obtained results above, we can conclude that our pseudo gradient is effective to improve DE with stronger diversity. However, to enlarge this positive effect of pseudo gradient leaves to the future work.

REFERENCES

[1] R.Storn and K.Price. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997. 10.1023/A:1008202821328.

[2] K.Price, R.Storn, and J.Lampinen. *Differential Evolution-A Practical Approach to Global optimization.* Springer Verlag, Berlin,Germany, 2005.

[3] Junhong Liu and Jouni Lampinen. A fuzzy adaptive differential evolution algorithm. In *TENCON '02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, volume 1, pages 606 – 611 vol.1, oct. 2002.

[4] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *Evolutionary Computation, IEEE Transactions on*, 10(6):646 –657, dec. 2006.

[5] J. Brest, V. Zumer, and M.S. Maucec. Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 215 –222, 0-0 2006.

[6] V.L. Huang, A.K. Qin, and P.N. Suganthan. Self-adaptive differential evolution algorithm for constrained real-parameter optimization. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 17 –24, 0-0 2006.

[7] Janez Brest, Borko Bokovi, Sao Greiner, Viljem umer, and Mirjam Mauec. Performance comparison of self-adaptive and adaptive differential evolution algorithms. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 11:617–629, 2007. 10.1007/s00500-006-0124-0.

[8] Jingqiao Zhang and A.C. Sanderson. Jade: Adaptive differential evolution with optional external archive. *Evolutionary Computation, IEEE Transactions on*, 13(5):945 –958, oct. 2009.

[9] A.K. Qin, V.L. Huang, and P.N. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *Evolutionary Computation, IEEE Transactions on*, 13(2):398 –417, april 2009.

[10] W.H.Press, S.A.Teukolsky, W.T.Vetterling, and B.P.Flaanery. *Numerical Recipes in C*. Cambridge Univ.Press, Cambridge, UK, 1994.

[11] A.K. Qin and P.N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 1785 – 1791 Vol. 2, sept. 2005.

[12] S.M. Islam, S. Das, S. Ghosh, S. Roy, and P.N. Suganthan. An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 42(2):482 – 500, april 2012.

[13] R.Salomon. Evolutionary algorithms and gradient search: similarities and differences. *Evolutionary Computation, IEEE Transactions on*, 2(2):45 –55, jul 1998.

TABLE VII: Results of PGDE/rand/1/bin with Different Timesteps

	dt=0.5 Mean(Std)	dt=0.05 Mean(Std)	dt=0.005 Mean(Std)
f1	6.0415e-13 (4.9504e-13)	3.3695e-25 (1.4917e-25)	**3.1509e-27 (1.4808e-27)**
f2	5.4031e+00 (3.0213e+00)	3.2253e-05 (2.7187e-05)	**7.6497e-11 (1.3982e-10)**
f3	**5.0663e+05 (4.0193e+05)**	5.1552e+05 (2.9022e+05)	2.4943e+07 (1.5307e+07)
f4	9.8215e+01 (5.5781e+01)	1.5046e-01 (1.8837e-01)	**4.7148e-03 (9.5225e-03)**
f5	5.4628e+02 (2.5958e+02)	**4.3144e+02 (2.4811e+02)**	8.5669e+02 (4.0329e+02)
f6	4.2008e+06 (2.0839e+07)	**1.2193e+02 (1.0457e+02)**	2.1095e+02 (6.9626e+02)
f7	**2.0772e-02 (1.2253e-02)**	2.4491e-02 (2.3381e-02)	2.1453e-02 (1.3602e-02)
f8	2.0248e+01 (7.9406e-02)	2.0190e+01 (5.3467e-02)	**2.0094e+01 (5.0834e-02)**
f9	1.1948e+01 (2.9040e+00)	1.4574e+01 (3.7175e+00)	**7.5684e+00 (2.0280e+00)**
f10	5.0305e+01 (1.4404e+01)	**3.7059e+01 (1.3151e+01)**	4.7758e+01 (1.4153e+01)
f11	1.9469e+01 (2.3448e+00)	1.8575e+01 (2.8601e+00)	**9.6940e+00 (2.4074e+00)**
f12	**7.1353e+03 (7.9096e+03)**	8.0804e+03 (1.1108e+04)	3.5544e+04 (2.7264e+04)
f13	2.6639e+00 (9.0135e-01)	3.5494e+00 (6.6556e-01)	**2.5460e+00 (5.8094e-01)**
f14	**1.2994e+01 (4.4777e-01)**	1.3237e+01 (3.3412e-01)	1.3126e+01 (4.0344e-01)
f15	3.5684e+02 (9.4427e+01)	3.7022e+02 (1.1111e+02)	**3.2289e+02 (1.1268e+02)**
f16	1.5204e+02 (1.1726e+02)	**7.0941e+01 (2.4790e+01)**	9.7246e+01 (7.4342e+01)
f17	1.0685e+02 (4.3476e+01)	1.0933e+02 (3.3507e+01)	**1.0183e+02 (4.3635e+01)**
f18	9.0589e+02 (5.6861e-01)	**9.0408e+02 (3.0238e-01)**	9.0412e+02 (4.0976e+00)
f19	9.0575e+02 (4.5663e-01)	9.0415e+02 (8.4920e-01)	**9.0269e+02 (4.8259e+00)**
f20	9.0573e+02 (4.4817e-01)	9.0410e+02 (6.6735e-01)	**9.0121e+02 (4.8739e+00)**
f21	5.0000e+02 (1.0705e-04)	**5.0000e+02 (1.6583e-05)**	5.1200e+02 (6.0000e+01)
f22	8.9748e+02 (1.3235e+01)	8.9410e+02 (1.4225e+01)	**8.7038e+02 (2.9007e+01)**
f23	**5.3416e+02 (6.6218e-04)**	5.5028e+02 (8.0567e+01)	5.4964e+02 (7.7380e+01)
f24	2.3135e+02 (1.5676e+02)	2.0000e+02 (1.5264e-04)	**2.0000e+02 (2.1512e-12)**
f25	**2.1065e+02 (7.7258e-01)**	2.1122e+02 (6.2671e-01)	2.1192e+02 (8.5702e-01)
+	6	6	13

[14] Ji-Pyng Chiou and Feng-Sheng Wang. Hybrid method of evolutionary algorithms for static and dynamic optimization problems with application to a fed-batch fermentation process. *Computers and Chemical Engineering*, 23(9):1277 – 1291, 1999.

[15] J.Y. Wen, Q.H. Wu, L. Jiang, and S.J. Cheng. Pseudo-gradient based evolutionary programming. *Electronics Letters*, 39(7):631 – 632, april 2003.

[16] T. Takahama and S. Sakai. Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1 –9, july 2010.

[17] Gang Yu, Tianyou Chai, and Xiaochuan Luo. Multiobjective production planning optimization using hybrid evolutionary algorithms for mineral processing. *Evolutionary Computation, IEEE Transactions on*, 15(4):487 –514, aug. 2011.

[18] P.N. Suganthan, N.Hansen, J.J.Liang, K.Deb, Y-P.Chen, A.Auger, and S.Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. In *Nanyang Technol.Univ.,Singapore.*, May 2005.