Using Opposition-based Learning with Particle Swarm Optimization and Barebones Differential Evolution

Mahamed G.H. Omran
Gulf University for Science and Technology
Kuwait

1. Introduction

Particle swarm optimization (PSO) (Kennedy & Eberhart, 1995) and differential evolution (DE) (Storn & Price, 1995) are two stochastic, population-based optimization methods, which have been applied successfully to a wide range of problems as summarized in Engelbrecht (2005) and Price et al. (2005).

A number of variations of both PSO and DE have been developed in the past decade to improve the performance of these algorithms (Engelbrecht, 2005; Price et al. 2005). One class of variations includes hybrids between PSO and DE, where the advantages of the two approaches are combined. The barebones DE (BBDE) is a PSO-DE hybrid algorithm proposed by Omran et al. (2007) which combines concepts from the barebones PSO (Kennedy 2003) and the recombination operator of DE. The resulting algorithm eliminates the control parameters of PSO and replaces the static DE control parameters with dynamically changing parameters to produce an almost parameter-free, self-adaptive, optimization algorithm.

Recently, opposition-based learning (OBL) was proposed by Tizhoosh (2005) and was successfully applied to several problems (Rahnamayan et al., 2008). The basic concept of OBL is the consideration of an estimate and its corresponding opposite estimate simultaneously to approximate the current candidate solution. Opposite numbers were used by Rahnamayan et al. (2008) to enhance the performance of Differential Evolution. In addition, Han and He (2007) and Wang et al. (2007) used OBL to improve the performance of PSO. However, in both cases, several parameters were added to the PSO that are difficult to tune. Wang et al. (2007) used OBL during swarm initialization and in each iteration with a user-specified probability. In addition, Cauchy mutation is applied to the best particle to avoid being trapping in local optima. Similarly, Han and He (2007) used OBL in the initialization phase and also during each iteration. However, a constriction factor is used to enhance the convergence speed.

In this chapter, OBL is used to improve the performance of PSO and BBDE without adding any extra parameter. The performance of the proposed methods is investigated when applied to several benchmark functions. The experiments conducted show that OBL improves the performance of both PSO and BBDE.

The remainder of the chapter is organized as follows: PSO is summarized in Section 2. DE is presented in Section 3. Section 4 provides an overview of BBDE. OBL is briefly reviewed in

Section 5. The proposed methods are presented in Section 6. Section 7 presents and discusses the results of the experiments. Finally, Section 8 concludes the chaper.

2. Particle Swarm Optimization

Particle swarm optimization (PSO) is a stochastic, population-based optimization algorithm modeled after the simulation of social behavior of bird flocks. In a PSO system, a swarm of individuals (called *particles*) fly through the search space. Each particle represents a candidate solution to the optimization problem. The position of a particle is influenced by the best position visited by itself (i.e. its own experience) and the position of the best particle in its neighborhood (i.e. the experience of neighboring particles). Particle position, x_i, are adjusted using

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1) \tag{1}$$

where the velocity component, v_i, represents the step size. For the basic PSO,

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_{1,j}(t)(y_{i,j}(t) - x_{i,j}(t)) + c_2 r_{2,j}(t)(\hat{y}_j(t) - x_{i,j}(t)) \tag{2}$$

where w is the inertia weight (Shi & Eberhart, 1998), c_1 and c_2 are the acceleration coefficients, $r_{1,j}$, $r_{2,j} \sim U(0,1)$, $\boldsymbol{y}_i$ is the personal best position of particle i, and $\hat{\boldsymbol{y}}_i$ is the neighborhood best position of particle i.

The neighborhood best position $\hat{\boldsymbol{y}}_i$, of particle i depends on the neighborhood topology used (Kennedy, 1999; Kenedy & Mendes, 2002). If a fully-connected topology is used, then $\hat{\boldsymbol{y}}_i$ refers to the best position found by the entire swarm. That is,

$$\hat{y}_i(t) \in \{y_0(t), y_1(t), \ldots, y_s(t)\} = min\{f(y_0(t)), f(y_1(t)), \ldots, f(y_s(t))\} \tag{3}$$

where s is the swarm size.

The resulting algorithm is referred to as the global best (*gbest*) PSO. A pseudo-code for PSO is shown in Alg. 1.

```
for each particle $i \in 1,...,s$ do
  Randomly initialize $\boldsymbol{x}_i$
  Set $\boldsymbol{v}_i$ to zero
  Set $\boldsymbol{y}_i = \boldsymbol{x}_i$
endfor
Repeat
  for each particle $i \in 1,...,s$ do
    Evaluate the fitness of particle $i$, $f(\boldsymbol{x}_i)$
    Update $\boldsymbol{y}_i$
    Update $\hat{\boldsymbol{y}}$ using equation (3)
    for each dimension $j \in 1,...,N_d$ do
      Apply velocity update using equation (2)
    endloop
    Apply position update using equation (1)
  endloop
Until some convergence criteria is satisfied
```

Algorithm 1. General pseudo-code for PSO

Van den Bergh and Engelbrecht (2006) and Clerc and Kennedy (2002) proved that each particle converges to a weighted average of its personal best and neighborhood best position, that is,

$$\lim_{t\to+\infty} x_{i,j}(t) = \frac{c_1 y_{i,j} + c_2 \hat{y}_{i,j}}{c_1 + c_2}$$

This theoretically derived behavior provides support for the barebones PSO developed by Kennedy (2003). It replaces Eqs. 1 and 2 with

$$x_{i,j}(t+1) = N\left(\frac{y_{i,j}(t) + \hat{y}_j(t)}{2}, \left|y_{i,j}(t) - \hat{y}_j(t)\right|\right)$$

Particle positions are therefore randomly selected from a Gaussian distribution with the mean given as the weighted average of the personal best and global best positions, i.e. the swarm attractor. Note that exploration is facilitated via the deviation, $y_{i,j}(t) - \hat{y}_j(t)$, which approaches zero as *t* increases. In the limit, all particles will converge on the attractor point.
Kennedy (2003) also proposed an alternative version of the barebones PSO where Eqs. 1 and 2 are replaced with

$$x_{i,j}(t+1) = \begin{cases} N\left(\frac{y_{i,j}(t) + \hat{y}_j(t)}{2}, \left|y_{i,j}(t) - \hat{y}_j(t)\right|\right) & \text{if } U(0,1) > 0.5 \\ y_{i,j}(t) & \text{otherwise} \end{cases}$$

Based on the above equation, there is a 50% chance that the *j*-th dimension of the particle dimension changes to the corresponding personal best position. This version of PSO biases towards exploiting personal best positions.

3. Differential Evolution

Differential evolution (DE) is an evolutionary algorithm proposed by Storn and Price (1995). While DE shares similarities with other evolutionary algorithms (EA), it differs significantly in the sense that distance and direction information from the current population is used to guide the search process. DE uses the differences between randomly selected vectors (individuals) as the source of random variations for a third vector (individual), referred to as the *target* vector. Trial solutions are generated by adding weighted difference vectors to the target vector. This process is referred to as the mutation operator where the target vector is mutated. A recombination, or crossover step is then applied to produce an offspring which is only accepted if it improves on the fitness of the parent individual.
The basic DE algorithm is described in more detail below with reference to the three evolution operators: mutation, crossover, and selection.
Mutation: For each parent, $\boldsymbol{x}_i(t)$, of generation *t*, a trial vector, $\boldsymbol{v}_i(t)$, is created by mutating a target vector. The target vector, $\boldsymbol{x}_{i_3}(t)$, is randomly selected, with $i \neq i_3$. Then, two

individuals $\boldsymbol{x}_{i_1}(t)$, and $\boldsymbol{x}_{i_2}(t)$ are randomly selected with $i_1 \neq i_2 \neq i_3 \neq i$, and the difference vector, $\boldsymbol{x}_{i_1}$ - $\boldsymbol{x}_{i_2}$, is calculated. The trial vector is then calculated as

$$\boldsymbol{v}_i(t) = \boldsymbol{x}_{i_3}(t) + F(\boldsymbol{x}_{i_1}(t) - \boldsymbol{x}_{i_2}(t)) \tag{4}$$

where the last term represents the mutation step size. In the above, F is a scale factor used to control the amplification of the differential variation. Note that $F \in (0, \infty)$.
Crossover: DE follows a discrete recombination approach where elements from the parent vector, $\boldsymbol{x}_i(t)$, are combined with elements from the trial vector, $\boldsymbol{v}_i(t)$, to produce the offspring, $\boldsymbol{\mu}_i(t)$. Using the binomial crossover,

$$\mu_{ij}(t) = \begin{cases} v_{ij}(t) & \text{if } U(0,1) < P_r \quad \text{or} \quad j = r \\ x_{ij}(t) & \text{otherwise} \end{cases}$$

where $j = 1, ..., N_d$ refers to a specific dimension, N_d is the number of genes (parameters) of a single chromosome, and $r \sim U(1,\ldots, N_d)$. In the above, p_r is the probability of reproduction (with $p_r \in [0, 1]$).
Thus, each offspring is a stochastic linear combination of three randomly chosen individuals when $U(0, 1) < p_r$; otherwise the offspring inherits directly from the parent. Even when $p_r = 0$, at least one of the parameters of the offspring will differ from the parent (forced by the condition $j = r$).
Selection: DE evolution implements a very simple selection procedure. The generated offspring, $\boldsymbol{\mu}_i(t)$, replaces the parent, $\boldsymbol{x}_i(t)$, only if the fitness of the offspring is better than that of the parent.

4. Barebones Differential Evolution

Both PSO and DE have their strengths and weaknesses. PSO has the advantage that formal proofs exist to show that particles will converge to a single attractor. The barebones PSO utilizes this information by sampling candidate solutions, normally distributed around the formally derived attractor point. Additionally, the barebones PSO has no parameters to be tuned. On the other hand, DE has the advantage of not being biased towards any prior defined distribution for sampling mutational step sizes and its selection operator follows a hill-climbing process. Mutational step sizes are determined as differences between individuals in the current population. One of the problems which both PSO and DE share is that control parameters need to be optimized for each new problem.

The barebones DE combines the strengths of both the barebones PSO and DE to form a new, efficient hybrid optimization algorithm. For the barebones DE, position updates are done as follows:

$$x_{i,j}(t) = \begin{cases} p_{i,j}(t) + r_{2,j} \times (x_{i_1,j}(t) - x_{i_2,j}(t)) & \text{if } U(0,1) > p_r \\ y_{i_3,j}(t) & \text{otherwise} \end{cases} \tag{5}$$

where

$$p_{i,j}(t) = r_{1,j}(t)\, y_{i,j}(t) + (1 - r_{1,j}(t))\, \hat{y}_{i,j}(t) \tag{6}$$

with i_1, i_2, $i_3 \sim U(1,\ldots,s)$, $i_1 \neq i_2 \neq i$, $r_{1,j}$, $r_{2,j} \sim U(0,1)$ and p_r is the probability of recombination.
Referring to Eq. 6, $\boldsymbol{p}_i(t)$ represents the particle attractor as a stochastic weighted average of personal best and global best positions, borrowing from the barebones PSO (Kennedy 2003). Referring to Eq. 5, the mutation operator of DE is used to explore around the current attractor, $\boldsymbol{p}_i(t)$, by adding a difference vector to the attractor. Crossover is done with a randomly selected personal best, $\boldsymbol{y}_{i_3}$, as these personal bests represent a memory of best solutions found by individuals since the start of the search process. Also note that the scale factor is a random variable. Using the position update in Eq. 6, for a proportion of (1- p_r) of the updates, information from a randomly selected personal best, $\boldsymbol{y}_{i_3}$, is used (facilitating exploitation), while for a proportion of p_r of the updates step sizes are mutations of the attractor point, $\boldsymbol{p}_\mathrm{i}$ (facilitating exploration). Mutation step sizes are based on the difference vector between randomly selected particles, $\boldsymbol{x}_{i_1}$ and $\boldsymbol{x}_{i_2}$. Using the above, the BBDE achieves a good balance between exploration and exploitation. It should also be noted that the exploitation of personal best positions does not focus on a specific position. The personal best position, $\boldsymbol{y}_{i_3}$, is randomly selected for each application of the position update.

5. Opposition-based Learning

Opposition-based learning (OBL) was first proposed by Tizhoosh (2005) and was successfully applied to several problems (Rahnamayan et al., 2008). Opposite numbers are defined as follows:
Let $x \in [a,b]$, then the opposite number x' is defined as

$$x' = a + b - x$$

The above definition can be extended to higher dimensions as follows:
Let $P\left(x_1, x_2, \cdots, x_n\right)$ be an n-dimensional vector, where $x_i \in [a_i,b_i]$ and i=1, 2, …, n. The opposite vector of P is defined by $P'\left(x_1', x_2', \cdots, x_n'\right)$ where

$$x_i' = a_i + b_i - x_i$$

6. Proposed Methods

In this chapter, OBL is used to enhance the performance of PSO and BBDE without adding any extra parameter. Two variants are proposed as follows:

6.1 Improved PSO (iPSO)

An improved version of PSO is proposed such that in each iteration the particle with the lowest fitness, $\boldsymbol{x}_b$, is replaced by its opposite (the *anti-particle*) as follows,

$$x_{b,j} = LB_j + UB_j - x_{b,j}$$

where $x_{b,j} \in [LB_j, UB_j]$, $j=1,2,\ldots,N_d$ and N_d is the dimension of the problem.
The velocity and personal experience of the anti-particle are reset. The global best solution is also updated. A pseudo-code for *i*PSO is shown in Alg. 2.
The rationale behind this approach is the basic idea of opposition-based learning: if we begin with a random guess, which is very far away from the existing solution, let say in worst case it is in the *opposite location*, then we should look in the *opposite direction*. In our case, the guess that is "very far away from the existing solution" is the particle with the lowest fitness.
The main difference between *i*PSO on one side and the approaches proposed by Han and He (2007) and Wang et al. (2007) on the other side, is that we did not introduce any extra parameter to the original PSO. In addition, *i*PSO uses only OBL to enhance the performance of PSO while (Han & He, 2007; Wang et al. 2007) use OBL combined with other techniques (e.g. Cauchy mutation).

```
for each particle $i \in 1,\ldots,s$ do
  for each dimension $j \in 1,\ldots,N_d$ do
    $x_{i,j} = LB_j + r_j \times (UB_j - LB_j)$
  endloop
endfor
for each particle $i \in 1,\ldots,s$ do
  Set $\boldsymbol{v}_i$ to zero
  Set $\boldsymbol{y}_i = \boldsymbol{x}_i$
endfor
Repeat
  for each particle $i \in 1,\ldots,s$ do
    Evaluate the fitness of particle $i$, $f(\boldsymbol{x}_i)$
    Update $\boldsymbol{y}_i$
    Update $\hat{\boldsymbol{y}}$ using equation (3)
    for each dimension $j \in 1,\ldots,N_d$ do
      Apply velocity update using Eq. (2)
    endloop
    Apply position update using equation (1)
  endloop
  Let $\boldsymbol{x}_b$ be the particle with the lowest fitness
  for each dimension $j \in 1,\ldots,N_d$ do
    $x_{b,j} = LB_j + UB_j - x_{b,j}$
  endloop
  $\boldsymbol{v}_b = \boldsymbol{0}$
  $\boldsymbol{y}_b = \boldsymbol{x}_b$
  if $f(\boldsymbol{x}_b) < f(\hat{\boldsymbol{y}})$
      $\hat{\boldsymbol{y}} = \boldsymbol{x}_b$
  endif
Until some convergence criteria is satisfied
```

Algorithm 2. General pseudo-code for *i*PSO

6.2 Improved BBDE (iBBDE)

Similar to *i*PSO, BBDE is modified such that in each iteration the particle with the lowest fitness, $\boldsymbol{x}_b$, is replaced by its opposite. The personal experience of the anti-particle is also reset. The global best solution is updated.

7. Experimental Results

This section compares the performance of the proposed methods with that of *gbest* PSO and BBDE discussed in Section 2 and 4, respectively. For the PSO algorithms, w = 0.72 and $c_1 = c_2$ = 1.49. These values have been shown to provide very good results (van den Berg, 2002). In addition s = 50 for all methods. All functions were implemented in 30 dimensions.
The following functions have been used to compare the performance of the different approaches. These benchmark functions provide a balance of unimodal, multimodal, separable and non-separable functions.
For each of these functions, the goal is to find the global minimizer, formally defined as

$$\text{Given } f\text{: } \Re^{N_d} \rightarrow \Re$$

$$\text{find } \boldsymbol{x}^* \in \Re^{N_d} \text{ such that } f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in \Re^{N_d}$$

The following functions were used:
A. *Sphere* function, defined as

$$f(\boldsymbol{x}) = \sum_{i=1}^{N_d} x_i^2$$

where $\boldsymbol{x}^* = \boldsymbol{0}$ and $f(\boldsymbol{x}^*) = 0$ for $-100 \leq x_i \leq 100$.
B. *Rosenbrock* function, defined as

$$f(x) = \sum_{i=1}^{N_d - 1} \left(100\left(x_i - x_{i-1}^2\right)^2 + \left(x_{i-1} - 1\right)^2\right)$$

where $\boldsymbol{x}^* = (1,1,\ldots,1)$ and $f(\boldsymbol{x}^*) = 0$ for $-2 \leq x_i \leq 2$.
C. *Rotated hyper-ellipsoid* function, defined as

$$f(\boldsymbol{x}) = \sum_{i=1}^{N_d} \left(\sum_{j=1}^{i} x_j \right)^2$$

where $\boldsymbol{x}^* = \boldsymbol{0}$ and $f(\boldsymbol{x}^*) = 0$ for $-100 \leq x_i \leq 100$.
D. *Rastrigin* function, defined as

$$f(\boldsymbol{x}) = \sum_{i=1}^{N_d} \left(x_i^2 - 10\cos(2\pi x_i) + 10 \right)$$

where $\boldsymbol{x}^* = \mathbf{0}$ and $f(\boldsymbol{x}^*) = 0$ for $-5.12 \le x_i \le 5.12$.

E. *Ackle*y's function, defined as

$$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{N_d} x_i^2} \right) - \exp\left(\frac{1}{30}\sum_{i=1}^{N_d} \cos(2\pi x_i) \right) + 20 + e$$

where $\boldsymbol{x}^* = \mathbf{0}$ and $f(\boldsymbol{x}^*) = 0$ for $-32 \le x_i \le 32$.

F. *Griewank* function, defined as

$$f(\boldsymbol{x}) = \frac{1}{4000}\sum_{i=1}^{N_d} x_i^2 - \prod_{i=1}^{N_d} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$$

where $\boldsymbol{x}^* = \mathbf{0}$ and $f(\boldsymbol{x}^*) = 0$ for $-600 \le x_i \le 600$.

G. *Salomon* function, defined as

$$f(x) = -\cos\left(2\pi \sum_{i=1}^{N_d} x_i^2 \right) + 0.1\sqrt{\sum_{i=1}^{N_d} x_i^2} + 1$$

where $\boldsymbol{x}^* = \mathbf{0}$ and $f(\boldsymbol{x}^*) = 0$ for $-100 \le x_i \le 100$.

Sphere, Rosenbrock and Rotated hyper-ellipsoid are unimodal, while Rastrigin, Ackley, Griewank and Salomon are difficult multimodal functions where the number of local optima increases exponentially with the problem dimension.

The results reported in this section are averages and standard deviations over 30 simulations. In order to have a fair comparison, each simulation was allowed to run for 50,000 evaluations of the objective function.

Table 1 summarizes the results obtained by applying the two PSO approaches to the benchmark functions. In general, the results show that *i*PSO performed better than (or equal to) *gbest* PSO. Figure 1 illustrates results for selected functions. The figure shows that *i*PSO generally reached good solutions faster than PSO. Similarly, Table 2 shows that *i*BBDE generally outperformed BBDE. Figure 2 illustrates results for selected functions. Thus, Tables 1 and 2 suggest that using the simple idea of replacing the worst particle is the main reason for improving the performance of PSO and BBDE. In additon, we can conclude that opposition-based learning improved the performance of both PSO and BBDE without requiring any extra parameter.

	PSO	*iPSO*
Sphere	0(0)	0(0)
Rosenbrock	22.191441 (1.741527)	20.645323 (0.426212)
Rotated hyper-ellipsoid	2.021006 (1.675313)	0.355572 (0.890755)
Rastrigin	48.487584 (14.599249)	27.460845 (11.966896)
Ackley	1.096863 (0.953266)	0(0)
Griewank	0.015806 (0.022757)	0.006163 (0.009966)
Salomon	0.446540 (0.122428)	0.113207 (0.034575)

Table 1. Mean and standard deviation (±SD) of the function optimization results

	BBDE	*iBBDE*
Sphere	0(0)	0(0)
Rosenbrock	25.826400 (0.216660)	25.942146 (0.209437)
Rotated hyper-ellipsoid	15.409460 (20.873456)	0.905987 (1.199178)
Rastrigin	34.761833 (28.598884)	0(0)
Ackley	0(0)	0(0)
Griewank	0.000329 (0.001800)	0(0)
Salomon	0.166540 (0.047946)	0.149917 (0.050826)

Table 2. Mean and standard deviation (±SD) of the function optimization results

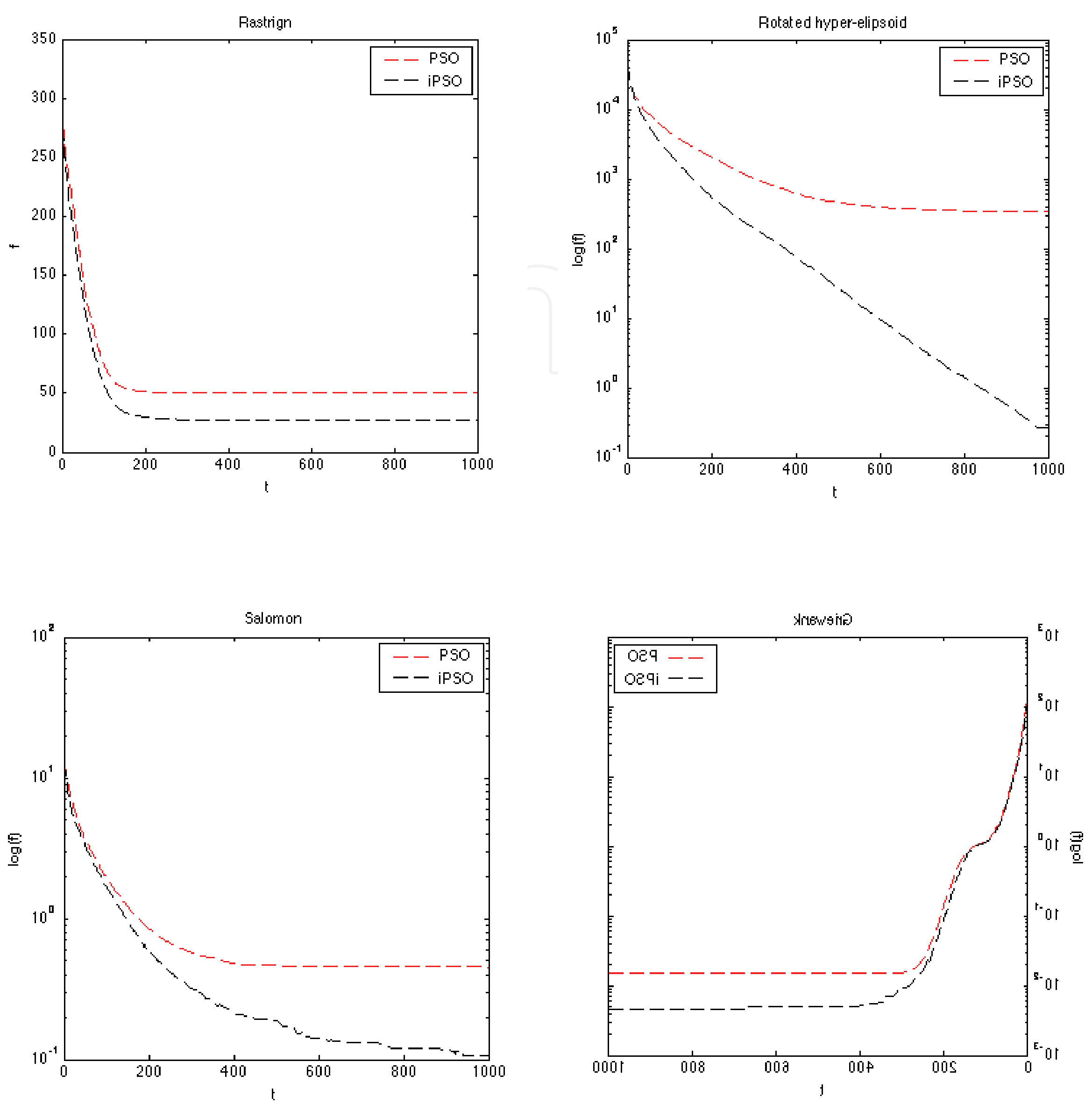

Figure 1. Performance Comparison of PSO and *i*PSO when applied to selected functions

7. Conclusion

Opposition-based learning was used in this chapter to improve the performance of PSO and BBDE. Two opposition-based variants were proposed (namely, *i*PSO and *i*BBDE). The *i*PSO and *i*BBDE algorithms replace the least-fit particle with its anti-particle. The results show that, in general, *i*PSO and *i*BBDE outperformed PSO and BBDE, respectively. In addition, the results show that using OBL enhances the performance of PSO and BBDE without requiring additional parameters. The ideas introduced in this chapter could also be used with any PSO/BBDE variant.
Future research will investigate the effect of noise on the performance of the proposed approaches. Furthermore, a scalability study will be conducted. Finally, applying the proposed approaches to real-world problem will be investigated.

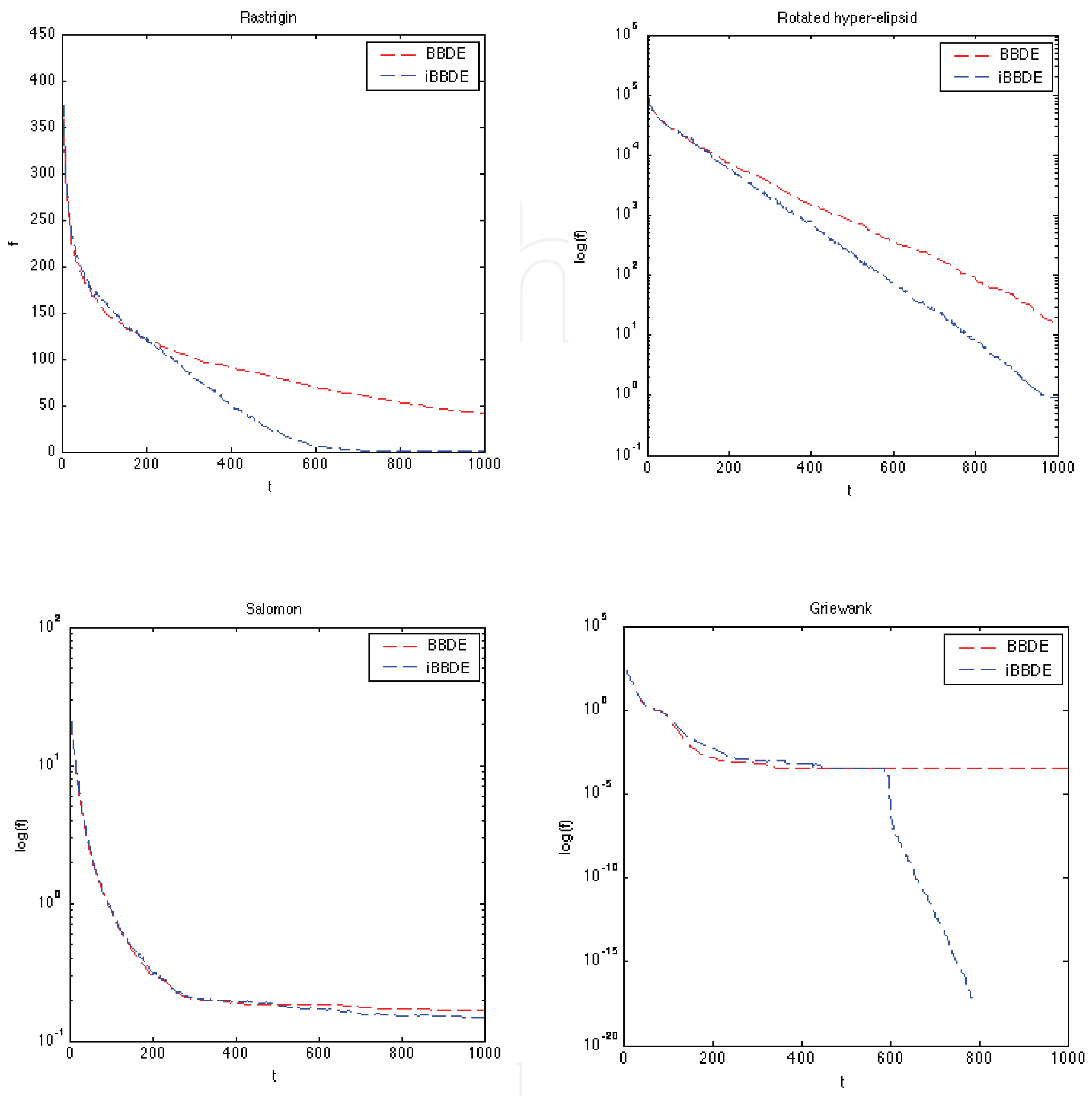

Figure 2. Performance Comparison of BBDE and *i*BBDE when applied to selected functions

8. References

Clerc, M. & Kennedy, J. (2002). The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 1, pp. 58-73.

Engelbrecht, A. (2005). *Fundamentals of Computational Swarm Intelligence*, Wiley & Sons.

Han, L. & He, X. (2007). A novel Opposition-based Particle Swarm Optimization for Noisy Problems. *Proceedings of the Third International Conference on Natural Computation*, IEEE Press, Vol. 3, pp. 624 – 629.

Kennedy, J. (1999). Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance. *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol. 3, pp. 1931-1938.

Kennedy, J. (2003). Bare Bones Particle Swarms. *Proceedings of the IEEE Swarm Intelligence Symposium*, pp. 80-87.

Kennedy, J. & Eberhart, R. (1995). Particle Swarm Optimization. *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 1942-1948.

Kennedy, J. & Mendes, R. (2002). Population Structure and Particle Performance. *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1671-1676, IEEE Press.

Omran, M., Engelbrecht, A. & Salman, A. (2007). Differential evolution based on particle swarm optimization. *Proceedings of the IEEE Swarm Intelligence Symposium*, pp. 112-119.

Price, K.; Storn, R. & Lampinen, J. (2005). *Differential Evolution: A Practical Approach to Global Optimization*, Springer.

Rahnamayan, S.; Tizhoosh, H. & Salama, M. (2008). Opposite-based Differential Evolution. *IEEE Trans. On Evolutionary Computation*, Vol. 12, No. 1, pp. 107-125.

Shi, Y. & Eberhart, R. (1998). A Modified Particle Swarm Optimizer. *Proceedings of the IEEE Congress on Evolutionary Computation,* pp. 69-73.

Storn, R. & Price, K. (1995). Differential evolution - A Simple and efficient adaptive scheme for global optimization over continuous spaces. *Technical Report TR-95-012*, International Computer Science Institute.

Tizhoosh, H. (2005). Opposition-based Learning: A New Scheme for Machine Intelligence. *Proceedings Int. Conf. Comput. Intell. Modeling Control and Autom*, Vol. I, pp. 695-701.

van den Bergh, F. (2002). An Analysis of Particle Swarm Optimizers. *PhD thesis*, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

van den Bergh, F. & Engelbrecht, A. (2006). A Study of Particle Swarm Optimization Particle Trajectories. *Information Sciences*, Vol. 176, No. 8, pp. 937-971.

Wang, H.; Liu, Y.; Zeng, S.; Li, H. & Li, C. (2007). Opposition-based Particle Swarm Algorithm with Cauchy Mutation. *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 4750-4756.