Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 761708, 12 pages
doi:10.1155/2012/761708

Research Article

Opposition-Based Barebones Particle Swarm for Constrained Nonlinear Optimization Problems

Hui Wang

School of Information Engineering, Nanchang Institute of Technology, Nanchang 330099, China

Correspondence should be addressed to Hui Wang, wanghui_cug@yahoo.com.cn

Received 29 December 2011; Revised 9 April 2012; Accepted 10 May 2012

Academic Editor: Jianming Shi

This paper presents a modified barebones particle swarm optimization (OBPSO) to solve constrained nonlinear optimization problems. The proposed approach OBPSO combines barebones particle swarm optimization (BPSO) and opposition-based learning (OBL) to improve the quality of solutions. A novel boundary search strategy is used to approach the boundary between the feasible and infeasible search region. Moreover, an adaptive penalty method is employed to handle constraints. To verify the performance of OBPSO, a set of well-known constrained benchmark functions is used in the experiments. Simulation results show that our approach achieves a promising performance.

1. Introduction

Many engineering problems can be converted to constrained optimization problems. The aim of constrained optimization is to find a feasible solution with minimized cost (In this paper, we only consider minimization problems). A general constrained minimization optimization problem can be defined as follows.

$$\text{Minimize } f(x) \tag{1.1}$$

subject to

$$\begin{aligned} g_j(x) &\le 0, \quad j = 1, 2, \ldots, q, \\ h_j(x) &= 0, \quad j = q+1, 2, \ldots, m, \end{aligned} \tag{1.2}$$

where $g(x)$ is inequality constraint, $h(x)$ is the equality constraint, m is the number of constraints, q is the number of inequality constraints, and $m - q$ is the number of equality constraints.

Particle swarm optimization (PSO) is a population-based stochastic search algorithm developed by Kennedy and Eberhart [1]. Although PSO shares many similarities with evolutionary algorithms (EAs), the standard PSO does not use evolution operators such as crossover and mutation. For PSO's simple concept, easy implementation, and effectiveness, it has been widely applied to many optimization areas.

Although PSO has shown a good performance over many optimization problems, it does not work well when solving complex problems. Especially for constrained optimization problems, the standard PSO could hardly search promising solutions. The possible reason is that constrained optimization problems are usually multimodal and having some constraints. PSO could easily fall into local minima and hardly search feasible solutions. To enhance the performance of PSO on constrained optimization problems, many improved PSO variants have been proposed in the past several years.

Meng et al. [2] used a quantum-inspired PSO (QPSO) to solve constrained economic load dispatch. The QPSO shows stronger search ability and quicker convergence speed, not only because of the introduction of quantum computing theory, but also due to two-novel strategies: self-adaptive probability selection and chaotic sequences mutation. Coelho [3] presented a novel quantum-behaved PSO (QPSO) to solve constrained engineering design problems. The proposed approach embedded a Gaussian mutation into QPSO to prevent premature convergence to local minima. Sun et al. [4] proposed an improved vector PSO (IVPSO) based on multidimensional search, in which a simple constraint-preserving method is used to handle constraints. Liu et al. [5] used a hybrid PSO called PSO-DE to solve constrained numerical and engineering optimization problems. The PSO-DE integrates PSO with differential evolution (DE) to obtain a good performance. Venter and Haftka [6] proposed a new method to solve constrained optimization problems. The constrained, single objective optimization problem is converted into an unconstrained, biobjective optimization problem that is solved using a multiobjective PSO algorithm. Lu and Chen [7] presented an enhanced PSO by employing a dynamic inertia weight to avoid premature convergence. The inertia weight of every individual is dynamically controlled by the Euclidean distance between individual and the global best individual. Daneshyari and Yen [8] proposed a cultural-based constrained PSO to incorporate the information of the objective function and constraint violation. The archived information facilitates communication among swarms in the population space and assists in selecting the leading particles in three different levels: personal, swarm, and global levels.

There have been many modifications to the original PSO algorithm to improve the efficiency and robustness of the search. Although these modified PSO variants have shown good search abilities, their performance greatly depends on the control parameters in the velocity updating model, such as inertia weight (w) and acceleration coefficients (c_1 and c_2). Recently, a parameter-free PSO, known barebones PSO (BPSO) [9], used Gaussian normal distribution to update the particles in the population. It does not involve inertia weight, acceleration coefficients, and velocity. Its performance has been found to be competitive, and a number of BPSO algorithms have been proposed in the past several years. Omran et al. [10] incorporate the idea of BPSO into DE. Krohling and Mendel [11] employed a jump strategy in BPSO to avoid premature convergence. Motivated by the idea of BPSO, this paper presents an improved BPSO, namely OBPSO, to solve constrained nonlinear optimization problems. In OBPSO, opposition-based learning (OBL) concept [12] is used for population

initialization and generation jumping. To verify the performance of OBPSO, a set of well-known constrained benchmark problems are used in the experiments. Results obtained by the proposed OBPSO are compared with those in the literature and discussed.

The rest of the paper is organized as follows. In Section 2, the standard PSO and barebones PSO are briefly introduced. Section 3 describes our proposed approach. Section 4 presents experimental simulations, results, and discussions. Finally, the work is concluded in Section 5.

2. Belief Descriptions of PSO and Barebones PSO

In traditional PSO, a member in the swarm, called a particle, represents a potential solution in the D-dimensional search space. Each particle has two vectors: velocity and position. It is attracted by its previous best particle ($pbest$) and the global best particle ($gbest$) during the evolutionary process. The velocity v_{ij} and position x_{ij} of the jth dimension of the ith particle are updated according to (2.1) [13]:

$$\begin{aligned} v_{ij}(t+1) &= w \cdot v_{ij}(t) + c_1 \cdot \text{rand1}_{ij} \cdot \Big(pbest_{ij}(t) - x_{ij}(t)\Big) \\ &\quad + c_2 \cdot \text{rand2}_{ij} \cdot \Big(gbest_j(t) - x_{ij}(t)\Big), \\ x_{ij}(t+1) &= x_{ij}(t) + v_{ij}(t+1), \end{aligned} \tag{2.1}$$

where $i = 1, 2, \ldots, N$ is the particle's index, N is the population size, $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ is the position of the ith particle; $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$ represents the velocity of the ith particle; the $pbest_i = (pbest_{i1}, pbest_{i2}, \ldots, pbest_{iD})$ is the best previous position yielding the best fitness value for the ith particle; $gbest = (gbest_1, gbest_2, \ldots, gbest_D)$ is the global best particle found by all particles so far. The parameter w, called inertia factor, which is used to balance the global and local search abilities of particles [13], rand1_{ij} and rand2_{ij} are two random numbers generated independently within the range of $[0, 1]$, c_1 and c_2 are two learning factors which control the influence of the social and cognitive components, and $t = 1, 2, \ldots$ indicates the iteration number.

Recently, Kennedy [9] developed the barebones PSO (BPSO). This new version of PSO eliminates the velocity term, and the position is updated as follows.

$$x_{ij}(t+1) = G\left(\frac{gbest_j(t) + pbest_{ij}(t)}{2}, \left|gbest_j(t) - pbest_{ij}(t)\right|\right), \tag{2.2}$$

where $x_{ij}(t+1)$ is the position of the ith particle in the population, and G represents a Gaussian distribution with mean $(gbest_j(t) + pbest_{ij}(t))/2$ and standard deviation $|gbest_j(t) - pbest_{ij}(t)|$.

Note that the particle positions are sampled by the above Gaussian distribution. The BPSO facilitates initial exploration, due to large deviation (initially, $pbest$ will be far from the $gbest$). As the number of generation increases, the deviation approaches to zero, by focussing on exploitation of the $pbest$ and $gbest$ [14].

3. Opposition-Based Barebones PSO (OBPSO)

3.1. Opposition-Based Learning

Opposition-based learning (OBL) developed by Tizhoosh [12] is a new concept in computational intelligence. It has been proven to be an effective concept to enhance various optimization approaches [15–17]. When evaluating a solution x to a given problem, simultaneously computing its opposite solution will provide another chance for finding a candidate solution which is closer to the global optimum.

Let $X = (x_1, x_2, \ldots, x_D)$ be a candidate solution in a D-dimensional space, where $x_1, x_2, \ldots, x_D \in R$ and $x_j \in [a_j, b_j]$, $j \in 1, 2, \ldots, D$. The opposite solution $X^* = (x_1^*, x_2^*, ..., x_D^*)$ is defined by [15]

$$x_j^* = a_j + b_j - x_j. \tag{3.1}$$

By staying within variables' interval static boundaries, we would jump outside of the already shrunken search space and the knowledge of the current converged search space would be lost. Hence, we calculate opposite particles by using dynamically updated interval boundaries $[a_j(t), b_j(t)]$ as follows [15].

$$x_{ij}^* = a_j(t) + b_j(t) - x_{ij}, \tag{3.2}$$

$$a_j(t) = \min\left(x_{ij}(t)\right), \qquad b_j(t) = \max\left(x_{ij}(t)\right), \tag{3.3}$$

$$\begin{gathered} x_{ij}^* = \operatorname{rand}\left(a_j(t), b_j(t)\right), \quad \text{if } x_{ij}^* < x_{\min} \Big\| x_{ij}^* > x_{\max}, \\ i = 1, 2, \ldots, N, \quad j = 1, 2, \ldots, D, \end{gathered} \tag{3.4}$$

where x_{ij} is the jth position element of the ith particle in the population, x_{ij}^* is the opposite particle of x_{ij}, $a_j(t)$ and $b_j(t)$ are the minimum and maximum values of the jth dimension in current search space, respectively, $\operatorname{rand}(a_j(t), b_j(t))$ are random numbers within $[a_j(t), b_j(t)]$, $[x_{\min}, x_{\max}]$ is the box-constraint of the problem, and N is the population size, and $t = 1, 2, \ldots$, indicates the generations.

3.2. Adaptive Constraint Handling

To handle the constraints in solving constrained optimization problems, this paper employs an adaptive penalty method (APM) which was early considered in [18–20]. It aims to help users avoid manually defining the coefficients of penalty functions. In the APM, each constraint of the candidate solutions is monitored. If a constraint seems to be more difficult to satisfy, then a larger penalty coefficient is added.

For each candidate solution, its jth constraint violation $\mathrm{cv}_j(x)$ is computed as follows:

$$\mathrm{cv}_j(x) = \begin{cases} \left|h_j(x)\right| \\ \min\left\{0, g_j(x)\right\}, \end{cases} \tag{3.5}$$

where $h_j(x)$ is the jth equality constraint, and $g_j(x)$ is the jth inequality constraint.

The fitness value of candidate solution is defined by

$$F(x) = \begin{cases} f(x), & \text{if } x \text{ is feasible} \\ \widehat{f}(x) + \sum_{j=1}^{m} (k_j \cdot \mathrm{cv}_j(x)), & \text{otherwise,} \end{cases} \tag{3.6}$$

where m is the number of constraints, k_j is a penalty coefficient for each constraint, and $\widehat{f}(x)$ is defined by

$$\widehat{f}(x) = \begin{cases} f(x), & \text{if } \overline{f}(x) \text{ is better than } f(x) \\ \overline{f}(x), & \text{otherwise,} \end{cases} \tag{3.7}$$

where $\overline{f}(x)$ is the average objective function values in the current swarm, and it is computed as

$$\overline{f}(x) = \frac{1}{N} \cdot \sum_{i=1}^{N} f(x_i), \tag{3.8}$$

where N is the population size.

For the penalty coefficient k_j, it determines the scaled factor of the jth constraint. Every generation, the k_j is adaptively adjusted as follows.

$$k_j = \left|\overline{f}(x)\right| \cdot \frac{\overline{\mathrm{cv}_j}(x)}{\sum_{l=1}^{m} \left(\overline{\mathrm{cv}_l}(x)\right)^2}, \tag{3.9}$$

where $\overline{\mathrm{cv}_l}(x)$ is the violation of the lth constraint averaged over the current swarm, and it is computed by

$$\overline{\mathrm{cv}_j}(x) = \frac{1}{N} \cdot \sum_{i=1}^{N} \mathrm{cv}_{ij}(x), \tag{3.10}$$

where $\mathrm{cv}_{ij}(x)$ is the violation of ith particle on the jth constraint.

3.3. Boundary Search Strategy

For constrained optimization problems, the solution search space can be divided into two parts: feasible space and infeasible space. In some cases, the global optimum is located at the boundaries of feasible space. It is difficult to find this kind of solutions. Because many algorithms can hardly judge the boundaries of feasible space. To tackle this problem, this paper employs a boundary search strategy as follows.

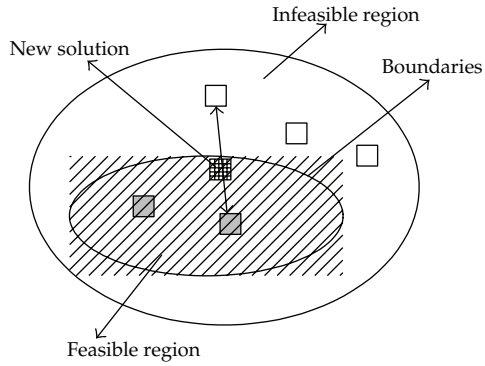

Figure 1: The boundary search strategy.

If the current population contains M feasible solutions and $N - M$ infeasible solutions ($M < N$), then we randomly select one feasible solution x and one infeasible solution y. Based on x and y, a new solution z is generated by

$$z_j = 0.5 \cdot x_j + 0.5 \cdot y_j, \tag{3.11}$$

where $j = 1, 2, \ldots, D$.

Note that the boundary search strategy works when the current population contains feasible and infeasible solutions. Figure 1 clearly illustrates the boundary search. As seen, if the new solution z is infeasible, then replace y with z. Because z is nearer to the boundary than y. If z is feasible, then replace x with z. Because z is nearer to the boundary than x.

3.4. The Proposed Approach

The proposed approach (OBPSO) uses a similar procedure to that of opposition-based differential evolution (ODE) for opposition-based population initialization and dynamic opposition [15]. To handle the constraints, we define a new fitness evaluation function as described in (3.6). Moreover, we also use a boundary search strategy to find the solutions located at the margin of the feasible region. The framework of OBPSO is shown in Algorithm 1, where P is the current population, OP is the population after using OBL, P_i is the ith particle in P, OP_i is the ith particle in GOP, p_o is the probability of opposition, N is the population size, D is the dimension size, $[a_j(t), b_j(t)]$ is the interval boundaries of current population, FEs is the number of fitness evaluations, and MAX_FEs is the maximum number of fitness evaluations.

4. Experimental Verifications

4.1. Test Problems

To verify the performance our proposed approach, we employ a set of 13 benchmark functions from the literature [21, 22]. The main characteristics of these benchmark functions are summarized in Table 1. For specific definitions of these functions, please refer to [23].

```
Uniform randomly initialize each particle in the swarm;
Initialize pbest and gbest;
While FEs ≤ MAX_FEs do
    /* Barebones PSO                                                    */
  for i = 1 to N do
    Calculate the position of the ith particle P_i according to (2.2);
    Calculate the fitness value of P_i according to (3.6);
     FEs ++;
  end
    /* Opposition-based learning                                        */
   if rand(0, 1) ≤ p_o then
     Update the dynamic interval boundaries [a_j(t), b_j(t)] in P according to (3.3);
     for i = 1 to N do
       Generate the opposite particle of P_i according to (3.2)
       Calculate the fitness value of P_i according to (3.6)
       FEs ++;
     end
     Select N fittest particles from {P, OP} as current population P;
   end
    /* Boundary search strategy                                         */
   if M < N then
     K = min{M, N − M}
     for i = 1 to K do
       Randomly select a feasible solution x and an infeasible solution y;
       Generate a new solution z according to (3.11);
       Calculate the fitness value of z according to (3.6);
       FEs ++;
       if z is feasible then
             x = z;
       end
       else
             y = z;
       end
     end
   end
   Update pbest and gbest;
end
```

Algorithm 1: The proposed OBPSO algorithm.

4.2. Comparison of OBPSO with Similar PSO Algorithms

In this section, we compare the performance of OBPSO with standard PSO, barebones PSO (BPSO), and OBPSO without boundary search strategy (OBPSO-1). To have a fair comparison, the same settings are used for common parameters. The population size N is set to 40. For PSO, The inertia weight w is set to 0.72984. The acceleration coefficients c_1 and c_2 are set to 1.49618. The maximum velocity $V_{\max}$ was set to the half range of the search space for each dimension. For OBPSO-1 and OBPSO, the probability of opposition p_o is set to 0.3. For each test functions, both OBPSO and PSO stop running when the number of iterations reaches to 1,000.

Table 2 presents average results of the four PSO algorithms over 30 runs, where Mean represents the mean best function values. As seen, PSO outperforms BPSO on only one

Table 1: Summary of main characteristics of benchmark problems, where F means the feasible region, S is the whole search region, NE represents nonlinear equality, NI indicates nonlinear inequality, LI means linear inequality, and α is the number of active constraints at optimum.

Problems	D	Type	$\lvert F\rvert/\lvert S\rvert$	LI	NE	NI	α
$G01$	13	Quadratic	0.011%	9	0	0	6
$G02$	20	Nonlinear	99.990%	1	0	1	1
$G03$	10	Polynomial	0.002%	0	1	0	1
$G04$	5	Quadratic	52.123%	0	0	6	2
$G05$	4	Cubic	0.000%	2	3	0	3
$G06$	2	Cubic	0.006%	0	0	2	2
$G07$	10	Quadratic	0.000%	3	0	5	6
$G08$	2	Nonlinear	0.856%	0	0	2	0
$G09$	7	Polynomial	0.512%	0	0	4	2
$G10$	8	Linear	0.001%	3	0	3	3
$G11$	2	Quadratic	0.000%	0	1	0	1
$G12$	3	Quadratic	4.779%	0	0	9^3	0
$G13$	5	Exponential	0.000%	0	3	0	3

problem $G05$. BPSO achieves better results than PSO on 7 problems. Both of them can find the global optimum on 5 problems. It demonstrates that the barebones PSO is better than standard PSO for these problems.

For the comparison of OBPSO with BPSO, both of them obtain the same results on 6 problems. For the rest 7 problems, OBPSO performs better than BPSO. It demonstrates that the opposition-based learning is helpful to improve the quality of solutions.

To verify the effects of the boundary search strategy, we compare the performance of OBPSO with OBPSO-1. For the OBPSO-1 algorithm, it does not use the proposed boundary search strategy. As seen, OBPSO outperforms OBPSO-1 on 6 problems, while they obtain the same results for the rest 7 problems. These results demonstrate the effectiveness of the boundary search strategy.

Figure 2 shows the evolutionary processes on four representative problems. It can be seen that OBPSO converges faster than other 3 PSO algorithms. The OBPSO-1 shows faster convergence rate than PSO and BPSO. This confirms that the opposition-based learning is beneficial for accelerating the evolution [15].

4.3. Comparison of OBPSO with Other State-of-the-Art PSO Variants

In this section, we compare the performance of OBPSO with three other PSO variants on the test suite. The involved algorithms and parameter settings are listed as follows.

(i) New vector PSO (NVPSO) [4].

(ii) Dynamic-objective PSO (RVPSO) [21].

(iii) Self-adaptive velocity PSO (SAVPSO) [22].

(iv) Our approach OBPSO.

Table 2: Mean best function values achieved by PSO, BPSO, OBPSO-1, and OBPSO, and the best results among the four algorithms are shown in bold.

Functions	Optimum	PSO mean	BPSO mean	OBPSO-1 mean	OBPSO mean
*G*01	−15	−13.013	−14.46	**−15**	**−15**
*G*02	−0.803619	−0.47191	−0.58944	−0.70536	**−0.79973**
*G*03	−1.0	−1.00468	−1.00339	−1.00287	**−1.00126**
*G*04	−30665.539	**−30665.5**	**−30665.5**	**−30665.5**	**−30665.5**
*G*05	5126.4981	5213.14	5228.32	5154.76	**5126.68**
*G*06	−6961.814	**−6961.81**	**−6961.81**	**−6961.81**	**−6961.81**
*G*07	24.306	25.9185	25.3492	24.8576	**24.4196**
*G*08	−0.095825	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**
*G*09	680.630	680.679	**680.630**	**680.630**	**680.630**
*G*10	7049.248	7639.4	7474.5	7292.82	**7049.2605**
*G*11	0.750	**0.749**	**0.749**	**0.749**	**0.749**
*G*12	−1.0	**−1.0**	**−1.0**	**−1.0**	**−1.0**
*G*13	0.05395	0.819146	0.61415	0.52312	**0.33837**

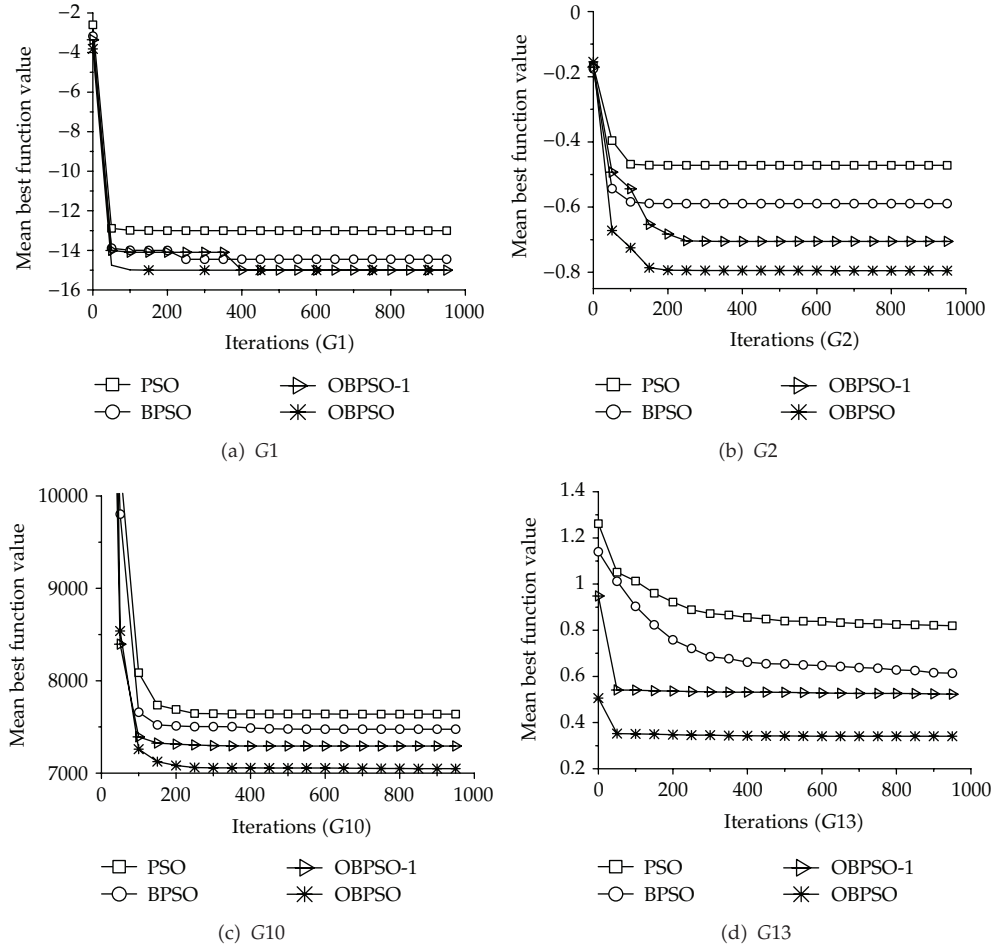

Figure 2: The evolutionary processes of PSO, BPSO, OBPSO-1, and OBPSO on four selected problems.

Table 3: Comparison results of OBPSO with other three PSO algorithms, where "$w/t/l$" means that OBPSO wins in w functions, ties in t functions, and loses in l functions, compared with its competitors. The best results among the four algorithms are shown in bold.

Functions	Optimum	NVPSO [4] mean	RVPSO [21] mean	SAVPSO [22] mean	OBPSO mean
$G01$	−15	−13.871875	−14.7151	−14.4187	**−15**
$G02$	−0.803619	−0.336263	−0.74057	−0.413257	**−0.79973**
$G03$	−1.0	−1.00484	−1.0034	−1.0025	**−1.00126**
$G04$	−30665.539	**−30665.5**	**−30665.5**	**−30665.5**	**−30665.5**
$G05$	5126.4981	**5126.4957**	5202.3627	5241.0549	5126.68
$G06$	−6961.814	**−6961.81**	**−6961.81**	**−6961.81**	**−6961.81**
$G07$	24.306	25.1301	24.989	**24.317**	24.4196
$G08$	−0.095825	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**
$G09$	680.630	680.634430	680.653	**680.630**	**680.630**
$G10$	7049.248	7409.065752	7173.2661	7049.2701	**7049.2606**
$G11$	0.750	**0.749**	**0.749**	**0.749**	**0.749**
$G12$	−1.0	**−1.0**	**−1.0**	**−1.0**	**−1.0**
$G13$	0.05395	0.465217	0.552753	0.681123	**0.33837**
$w/t/l$	—	7/5/1	8/5/0	6/6/1	—

The parameter settings of NVPSO are described in [4]. For RVPSO and SAVPSO, their parameter settings are given in [22]. For OBPSO, we use the same parameter values as described in the previous section. For each test functions, all algorithms stop running when the number of iterations reaches to the maximum value 1,000.

Table 3 presents average results of NVPSO, RVPSO, SAVPSO, and OBPSO over 30 runs, where Mean represents the mean best function values. The comparison results among OBPSO and other algorithms are summarized as $w/t/l$ in the last row of the table, which means that OBPSO wins in w functions, ties in t functions, and loses in l functions, compared with its competitors.

From the results of Table 3, OBPSO outperforms NVPSO on 7 problems, while NVPSO only achieves better results on a single problem. For the rest 5 problems, both OBPSO and NVPSO can find the global optimum. OBPSO performs better than RVPSO on 8 problems, while both of them obtain the same results for the rest 5 problems. For the comparison of SAVPSO and OBPSO, both of them achieve the same results on 6 problems. For the rest 7 problems, OBPSO wins 6, while SAVPSO wins only 1.

5. Conclusion

This paper proposes a modified barebones particle swarm optimization to solve constrained nonlinear optimization problems. The proposed approach is called OBPSO which employs two novel strategies including opposition-based learning and boundary search. Compared to other improved PSO variants, OBPSO is almost a parameter-free algorithm (except for the probability of opposition). Moreover, an adaptive penalty method is used to handle constraints. Experimental verifications on a set of constrained benchmark functions show that OBPSO achieves a promising performance compared to four other PSO variants. The parameter p_o may affect the performance of OBPSO. To determine the best choice of p_o, different values of p_o will be investigated. This will be conducted in the future work.

Acknowledgments

The authors would like to thank the editor and anonymous reviewers for their detailed and constructive comments that helped them to increase the quality of this work. This work is supported by the Science and Technology Plan Projects of Jiangxi Provincial Education Department (nos. GJJ12641, GJJ12633, and GJJ12307), and the National Natural Science Foundation of China (nos. 61070008, 61165004).

References

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks. Part 1*, pp. 1942–1948, December 1995.

[2] K. Meng, H. G. Wang, Z. Y. Dong, and K. P. Wong, "Quantum-inspired particle swarm optimization for valve-point economic load dispatch," *IEEE Transactions on Power Systems*, vol. 25, no. 1, pp. 215–222, 2010.

[3] L. D. S. Coelho, "Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1676–1683, 2010.

[4] C. L. Sun, J. C. Zeng, and J. S. Pan, "An new vector particle swarm optimization for constrained optimization problems," in *Proceedings of the International Joint Conference on Computational Sciences and Optimization (CSO '09)*, pp. 485–488, April 2009.

[5] H. Liu, Z. Cai, and Y. Wang, "Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization," *Applied Soft Computing Journal*, vol. 10, no. 2, pp. 629–640, 2010.

[6] G. Venter and R. T. Haftka, "Constrained particle swarm optimization using a bi-objective formulation," *Structural and Multidisciplinary Optimization*, vol. 40, no. 1–6, pp. 65–76, 2010.

[7] H. Lu and X. Chen, "A new particle swarm optimization with a dynamic inertia weight for solving constrained optimization problems," *Information Technology Journal*, vol. 10, no. 8, pp. 1536–1544, 2011.

[8] M. Daneshyari and G. G. Yen, "Constrained multiple-swarm particle swarm optimization within a cultural framework," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 18, pp. 1–16, 2011.

[9] J. Kennedy, "Bare bones particle swarms," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '03)*, pp. 80–87, 2003.

[10] M. G. H. Omran, A. P. Engelbrecht, and A. Salman, "Bare bones differential evolution," *European Journal of Operational Research*, vol. 196, no. 1, pp. 128–139, 2009.

[11] R. A. Krohling and E. Mendel, "Bare bones particle swarm optimization with Gaussian or cauchy jumps," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '09)*, pp. 3285–3291, May 2009.

[12] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA '05) and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (IAWTIC '05)*, pp. 695–701, November 2005.

[13] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 69–73, May 1998.

[14] A. P. Engelbrecht, "Heterogeneous particle swarm optimization," in *Proceedings of the International Conference on Swarm Intelligence*, pp. 191–202, 2010.

[15] R. S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, 2008.

[16] H. Wang, Z. Wu, S. Rahnamayan, Y. Liu, and M. Ventresca, "Enhancing particle swarm optimization using generalized opposition-based learning," *Information Sciences*, vol. 181, no. 20, pp. 4699–4714, 2011.

[17] H. Wang, Z. Wu, and S. Rahnamayan, "Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems," *Soft Computing*, vol. 15, no. 11, pp. 2127–2140, 2011.

[18] A. C. C. Lemonge and H. J. C. Barbosa, "An adaptive penalty scheme for genetic algorithms in structural optimization," *International Journal for Numerical Methods in Engineering*, vol. 59, no. 5, pp. 703–736, 2004.

[19] E. K. da Silva, H. J. C. Barbosa, and A. C. C. Lemonge, "An adaptive constraint handling technique for differential evolution with dynamic use of variants in engineering optimization," *Optimization and Engineering*, vol. 12, no. 1-2, pp. 31–54, 2011.

[20] X. Pan, Y. Cao, and Q. Pu, "Improved particle swarm optimization with adaptive constraint handling for engineering optimization," *Journal of Information and Computational Science*, vol. 8, no. 15, pp. 3507–3514, 2011.

[21] H. Y. Lu and W. Q. Chen, "Dynamic-objective particle swarm optimization for constrained optimization problems," *Journal of Combinatorial Optimization*, vol. 12, no. 4, pp. 409–419, 2006.

[22] H. Y. Lu and W. Q. Chen, "Self-adaptive velocity particle swarm optimization for solving constrained optimization problems," *Journal of Global Optimization*, vol. 41, no. 3, pp. 427–445, 2008.

[23] J. J. Liang, T. P. Runarsson, E. Mezura-Montes et al., "Problem definitions and evaluation criteria for the CEC 2006, special session on constrained real-parameter optimization," Tech. Rep., Nanyang Technological University, Singapore, 2006.