An Improved Particle Swarm Optimization with Feasibility-based Rules for Constrained Optimization Problems*

Chao-li Sun[1], Jian-chao Zeng[1], Jeng-shyang Pan[2]

[1] Complex System and Computational Intelligence Laboratory, Taiyuan University of Science and Technology, Taiyuan, Shanxi, P.R. China, 030024
clsun1225@163.com
[2] Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung, 807, Taiwan
jspan@cc.kuas.edu.tw

Abstract. This paper presents an improved particle swarm optimization (IPSO) to solve constrained optimization problems, which handles constraints based on certain feasibility-based rules. A turbulence operator is incorporated into IPSO algorithm to overcome the premature convergence. At the same time, a set called FPS is proposed to save those P_{best} locating in the feasible region. Different from the standard PSO, g_{best} in IPSO is chosen from the FPS instead of the swarm. Furthermore, the mutation operation is applied to the P_{best} with the maximal constraint violation value in the swarm, which can guide particles to close the feasible region quickly. The performance of IPSO algorithm is tested on a well-known benchmark suite and the experimental results show that the proposed approach is highly competitive, effective and efficient.

Keywords: Particle swarm optimization; Feasibility-based rules; constrained optimization problems.

1 Introduction

Evolutionary algorithms such as Genetic Algorithm, Evolutionary Strategies, Evolutionary Programming, etc. have been proposed to handle optimization problems [1-5]. Besides, many of them have been successfully applied for tackling constrained optimization problems during the past few years. Particle Swarm Optimization (PSO) is a new global evolutionary algorithm proposed by Kennedy and Eberhart in 1995 [6,7], its idea was based on the simulation of simplified social models such as bird flocking and fish schooling. PSO has been successfully applied in a variety of fields mainly for unconstrained continuous optimization problems [22-25]. Yet many real-world applications involve difficult constrained optimization problems that must be

* This work was supported in part by National Science Foundation of China under Grant NO.60674104 and by Natural Science Foundation of Shanxi Province of China under Grant NO. 20081030

solved efficiently and effectively, such as engineering design, VLSI design, structural optimization, economics, locations and allocation problems [8]. Disliking those deterministic optimization approaches, such as Feasible Direction and Generalized Gradient Descent [8,9], PSO algorithm is generally independent of the mathematic characteristics of the objective problem, and has been considered a valid technique to solve constrained optimization problems with a simple concept, easy implementation, quick convergence. However, like other aforementioned stochastic evolutionary algorithms, PSO also need an explicit constraint-handling mechanism. Generally, three main constraint-handling mechanisms can be incorporated into PSO for solving constrained optimization problems.

The penalty function method has been the most popular constraint-handling technique due to its simple principle. It converts a constrained optimization problem to an unconstrained optimization one through adding a penalty item to the objective function [10]. This method may work quite well for some problems, but it requires a careful tuning of the penalty parameters, and which turns out to be a difficult optimization problem itself [12], since both under_ and over_ penalizations may result in an unsuccessful optimization.

The feasibility-based method introduces feasibility-based rules which give an instruction on the determination of the best solution of the population (g_{best}) and the best historical solution of every particle (P_{best}) into PSO [13-14,20]. There is no need to design additional parameters, but according to feasibility-based rules, feasible solutions are always considered better than infeasible ones, which may cause the overpressure of selecting feasible solutions so as to result in premature convergence.

The constrained-preserving method (the feasible solutions method) reduces the search space by ensuring that all the candidate solutions satisfy the constraints at all times [11, 15-16]. Solutions are initialized within the feasible space, and transformations of candidate solutions are such that the resulting solutions still lie within the feasible region. This method requires an initialization of particle inside the feasible region, which may need a long time initialization process and may be hard to achieve for some problems.

This paper proposes an improved particle swarm optimization (IPSO) with feasibility-based rules to solve constrained optimization problems. A turbulence operator is introduced into IPSO algorithm to overcome the premature convergence. Otherwise, a set, called FPS, is specially introduced in IPSO algorithm to keep those P_{best} that is in the feasible region at current generation. Different to the standard PSO, g_{best} of IPSO is the best P_{best} from FPS, but the swarm. When the total number of P_{best} in FPS is less than the predefined constant, the mutation operation is manipulated to P_{best} that has the maximal constraint violation value in the swarm, which can guide particles to close the feasible region quickly.

The paper is organized as follows: In section 2, the problem of interest and the particle swarm optimization algorithm are stated briefly. Our proposed approach IPSO is provided in section 3. In section 4, the experimental setup and the results obtained are presented. Finally, section 6 presented the conclusions and the proposal for future research.

2 Basic Concepts

2.1 Problem statement

Generally, a constrained problem can be described as follows:

$$\begin{aligned} &min \quad f(\vec{x}) \\ &\text{s.t.} \quad g_i(\vec{x}) \le 0 \qquad i = 1, 2, \cdots, m \\ &\qquad\quad h_j(\vec{x}) \le 0 \qquad j = 1, 2, \cdots, l \\ &x_{dmin} \le x_d \le x_{dmax} \qquad d = 1, 2, \cdots, D \end{aligned} \tag{1}$$

Where $\vec{x} = (x_1, x_2, \cdots, D)$ is the vector of solutions such that $\vec{x} \in S \subseteq R^D$, S is defined as an D-dimensional space composed by lower and upper bounds $[x_{dmin}, x_{dmax}, d = 1, 2, \cdots, D$. m is the number of inequality constraints, l is the number of equality constraints, and the feasible region $F \subset S$ is the region of S that all constraints are satisfied. Commonly, an equality constraint is transformed into two inequality constraints $h_j(\vec{x}) \le \delta$ and $h_j(\vec{x}) \ge -\delta$, where δ is the tolerance allowed(a very small positive value). We call $\vec{x}$ a feasible solution when it satisfies all the constraints.

2.2 Standard Particle Swarm Optimization

Particle Swarm Optimization (PSO) is proposed as a global evolutionary algorithm by Kennedy and Eberhart in 1995[6,7], its idea was based on the simulation of simplified social models such as bird flocking and fish schooling. In PSO, it assumes that in a D-dimensional search space $S \subseteq R^D$, the swarm consists of N particles each has no volume and no weight, holds its own velocity and denotes a solution. The trajectory of each particle in the search space is dynamically adjusted by updating the velocity of each particle, according to its own flying experience as well as the experience of neighbor particles(built through tracking and memorizing the best position encountered). Particle i is in effect a D-dimensional vector $\vec{x}_i = (x_{i1}, x_{i2}, \cdots, x_{iD}) \in S$. Its velocity is also a D-dimensional vector $\vec{v}_i = (v_{i1}, v_{i2}, \cdots, v_{iD}) \in S$. The best historical position visited by particle i is a point in S, denoted as $P_i = (P_{i1}, P_{i2}, \cdots, P_{iD})$ or P_{best}, and the best historical position that the entire swarm has passed is denoted as $P_g = (P_{g1}, P_{g2}, \cdots, P_{gD})$ or g_{best}. To particle i, the new velocity and the new position of the d-th dimension ($1 \le d \le D$) are updated as follows[17]:

$$v_{id}(t+1) = \omega v_{id}(t) + c_1 r_1 (P_{id}(t) - x_{id}(t)) + c_2 r_2 (P_{gd}(t) - x_{id}(t))) \tag{2}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \tag{3}$$

Where ω is a parameter called the inertia weight, c_1 and c_2 are positive constants respectively referred as cognitive and social parameters, r_1 and r_2 are random numbers uniformly distributed in [0,1].

The process is repeated until a user-defined stopping criterion is reached. For more detail the reader is referred to [18].

2.3 The feasibility-based rule

Referring to [13], feasibility-based rules employed in this paper are described as follows:

(1) Any feasible solution is preferred to any infeasible solution.

(2) Between two feasible solutions, the one having better objective function value is preferred.

(3) Between two infeasible solutions, the one having smaller constraint violation is preferred.

Based on the above criteria, in the first and the third cases the search tends to the feasible region rather than infeasible one, and in the second case the search tends to the feasible region with good solutions. In brief, such a simple rule aims at obtaining good feasible solutions.

3 Our Proposed Approach

Figure 1 shows the algorithm of our IPSO algorithm.

Comparing to the standard particle swarm optimization, our algorithm is different from the standard PSO in following two aspects.

3.1 Updating P_{best} and g_{best}

In this paper, the constraint violation value of an infeasible solution is calculated as follows:

$$viol(x) = \sum_{i=0}^{m} max(0, g_i(\vec{x})) + \sum_{j=0}^{l} max(0, abs(h_j(\vec{x}))) \tag{4}$$

Suppose that $P_i(t)$ represents P_{best} of particle i at generation t and $\vec{x}_i(t+1)$ represents the newly generated position of particle i at generation $t+1$. In the standard PSO, $\vec{P}_i(t+1) = \vec{x}_i(t+1)$ only if $f(\vec{P}_i(t)) > f(\vec{x}_i(t+1))$. While in IPSO algorithm, the feasibility-based rule is employed. That is, $P_i(t)$ will replaced by $\vec{x}_i(t+1)$ at any of the following scenarios:

(1) $\vec{P}_i(t)$ is infeasible, but $\vec{x}_i(t+1)$ is feasible.

(2) Both $\vec{P}_i(t)$ and $\vec{x}_i(t+1)$ are feasible, but $f(\vec{P}_i(t)) > f(\vec{x}_i(t+1))$.

(3) Both $\vec{P}_i(t)$ and $\vec{x}_i(t+1)$ are infeasible, but $viol(\vec{P}_i(t)) > viol(\vec{x}_i(t+1))$

According to feasibility-based rules, there will be many infeasible particles if the feasible region is a highly constrained search space, which will lower the search efficiency. Thus, we define a feasible particle set as following:

$$FPS = \{\vec{P}_i(t) | \vec{P}_i(t) \quad is \quad feasible\}$$

If the particle number in FPS is more than rN (where r is a constant number that less than 1; N is the particle number), update velocity and position of each particle; otherwise, select one P_{best} in FPS to substitute P_{best} that the violation value is maximal. There are many methods to implement the selection, such as stochastic method, sequence method. In IPSO algorithm, sequence method is chosen to do the selection. Figure 2 shows the mutative procedure.

In standard PSO, g_{best} is the best historical position that the entire swarm has passed. In IPSO algorithm, g_{best} is the best P_{best} in FPS that has the best fitness value.

```
Function IPSO Algorithm
Begin
  For each particle
    Initialize position and velocity;
    Compute violation value;
    Compute fitness value;
    P_best = the particle's best history position;
    Put P_best in FPS if P_best is in feasible region;
  EndFor
  Do
    g_best = P_best that has the best fitness value in the feasible region;
    If the total number of P_best in FPS is less than or equal to rN then
        Mutate P_best of particle that the violation value is maximal with one P_best in FPS;
    EndIf
    Compute the average velocity;
    For each particle
        Calculate the new velocity and position with formula (5) and (6);
        Calculate the fitness value;
        If the fitness value is better than P_best according to feasibility-based rules then
            P_best =the particle's position;
        EndIf
    EndFor
  While stopping condition not satisfied
End.
```

Fig. 1. Pseudocode of the IPSO algorithm

```
Mutative procedure:
Begin
    If the number in FPS more than rN then
        Update the velocity and position;
    Else
        Calculate violation of each particle that inFPS=0;
        If (P_i is the position that has the maximum violation)
            and (P_j is in FPS and Particle_i.isselected = 0) then
                Particle_i.isselected = 1;
                Particle_j.inFPS = 1;
                Particle_j.isselected = 0;
        End If
    End If
End
```

Fig. 2. The mutative procedure

3.2 Updating velocity and position

According to feasibility-based rules, feasible solutions are always considered better than infeasible solutions. That may cause the overpressure of selecting feasible solutions so as to result in premature convergence. So to improve the exploration of the particle, in our algorithm, the swarm is manipulated according to the following update equations:

$$\begin{aligned} v_{id}(t+1) = \omega v_{id}(t) + c_1 r_1 (P_{id}(t) - x_{id}(t)) + \\ c_2 r_2 (P_{gd}(t) - x_{id}(t))) - \omega' \bar{v}_{id}(t) \end{aligned} \quad (5)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (6)$$

Where ω is a parameter called the inertia weight, c_1 and c_2 are positive constants respectively referred as cognitive and social parameters, r_1 and r_2 are random numbers uniformly distributed in [0,1], ω' is a parameter that we call the turbulence parameter ,$\bar{v}_{id}(t)$ is average velocity of the swarm at generation t.

4 Experiment and Discussions

To evaluate the performance of the proposed algorithm, we conducted a series of experiments on the well known Michalewicz' benchmark functions[19] extended by Runarsson and Yao[12]. All the testing problems are described in detail in [12].These test functions selected include characteristics that are representative of what can be considered "difficult" global optimization problems for an evolutionary algorithm.

Problems g02, g03, g08 and g12 are maximization problems and they were converted into minimization problems using $-f(\vec{x})$. Problems g03, g05, g11 and g13 involve equality constraints. All equality constraints $h_j(\vec{x})=0, j=1,2,\cdots,l$ have been transformed into inequality constraints $|h_j(\vec{x})|\le\delta$, using the degree of violation $\delta=10^{-4}$ when the particles were initialed. A total of 40 particles were employed, the maximum number of iterations were set to 8500 per run, and 30 independent runs of our algorithm were executed for each problem. The parameters in update equation are set as follows: ω linearly decreases from 0.9 to 0.4; to improve the diversity of the particle, c_1 is set to decrease from 3.6 to 2 and c_2 is set to increase from 0.2 to 2 respectively; and turbulence parameter $\omega^{'}$ linearly decreases from 0.6 to 0.

Table 1 summarizes the experimental results obtained using our algorithm with the above experimental settings, where "Opt" represents the known "optimal" solution for each problem, "Std" stands for "Standard deviation" of the obtained statistics for the 30 independent runs. The statistical results of our algorithm show that IPSO algorithm was able to find the global optimum in almost all test functions except for g10, and the standard deviation is small.

Moreover, we compared our results with respect to three algorithms representative of the state-of-the-art in the area: Stochastic Ranking (SR) [5], the Constraint-Handling Mechanism for PSO (CHMPSO) [13] and the Simple Multimembered Evolution Strategy (SMES) [21]. Our comparison of results with respect to the three previously described is presented in Table 2, Table 3 and Table 4.

Table 1. Experimental results on 13 benchmark functions using IPSO with $I_{max}=8500$

Pro	Opt	Best	Mean	Worst	Std
g01	-15	-15	-15	-15	0.000000
g02	0.803619	0.803603	0.663532	0.488776	0.015865
g03	1	1.004987	1.004860	1.004309	0.000024
g04	-30665.539	-30665.538672	-30665.538672	-30665.538672	0
g05	5126.4981	5126.498110	5126.500182	5126.507464	0.000460
g06	-6961.81388	-6961.813875	-6961.813856	-6961.813794	0.000003
g07	24.306	24.334560	24.961397	26.011018	0.081870
g08	0.095825	0.095825	0.095825	0.095825	0
g09	680.630	680.630765	680.668078	680.986895	0.0011918
g10	7049.3307	7251.206603	7377.286579	7582.967352	18.443359
g11	0.75	0.749000	0.749001	0.749003	0.000000
g12	1	1.000000	1.000000	1.000000	0.000000
g13	0.0539498	0.060055	0.060056	0.060065	0.000000

When comparing IPSO with respect to SR, we can see that IPSO found better solutions for g03, g06, g11 and similar result in g01, g04, g08 and g12. For problem g02, IPSO found better solution in terms of better solution. For problems g05 and g10, though the results of IPSO in terms of best solution are worse than SR, it does better in terms of mean solutions and worst solutions, which indicates that the capability to find optimal solution of IPSO algorithm is better than SR's for these two problems.

Compared with respect to CHMPSO, IPSO algorithm found better or similar solutions for all problems except g10. For problem g10, IPSO algorithm got better solutions in terms of mean solution and worst solution.

When comparing against SMES, IPSO found better solutions for g03, g05, g06, g11 and g13, and the same or similar results in g04, g08 and g12. For problems g02, g07 and g09, IPSO algorithm found better solutions in terms of best result.

Moreover, the computational cost measured in the number of evaluations of the objective function (FFE) performed by IPSO algorithm is 340,000 FFE, lower than the Stochastic Ranking (SR) which performed 350,000 FFE, and the same as CHMPSO and SMES.

5 Conclusions and Future Work

This paper presents an improved particle swarm optimization (IPSO) to solve constrained optimization problems, which handles constraints based on certain feasibility-based rules. IPSO introduces the turbulence operator to improve the swarm diversity and to overcome the premature convergence validly. The record about the feasible P_{best} in FPS and the mutation operator on the P_{best} with a maximal violation value can lead the swarm flying to the feasible region quickly. Obviously, the principle of IPSO algorithm is simple and its implement is easy, the experimental results also show the technique is highly competitive.

The future work is to study alternative mechanisms to accelerate convergence while keeping the same quality of the results achieved in this paper.

Table 2. Comparison of our IPSO with respect to SR[21]

Problem	Optimal	Best Result		Mean Result		Worst Result	
		IPSO	SR	IPSO	SR	IPSO	SR
g01	-15	-15	-15	-15	-15	-15	-15
g02	0.803619	0.803603	0.803515	0.663532	0.781975	0.488776	0.726288
g03	1	1.004987	1.000000	1.004860	1.000000	1.004309	1.000000
g04	-30665.539	-30665.538672	-30665.539	-30665.538672	-30665.539	-30665.538672	-30665.539
g05	5126.4981	5126.498110	5126.497	5126.500182	5128.881	5126.507464	5142.472
g06	-6961.81388	-6961.813875	-6961.814	-6961.813856	-6875.94	-6961.813794	-6350.262
g07	24.306	24.334560	24.07	24.961397	24.374	26.011018	24.642
g08	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825
g09	680.630	680.630765	680.630	680.668078	680.656	680.986895	680.763
g10	7049.3307	7251.206603	7054.316	7377.286579	7559.192	7582.967352	8835.655
g11	0.75	0.749000	0.75	0.749001	0.75	0.749003	0.75
g12	1	1.000000	1	1.000000	1	1.000000	1
g13	0.0539498	0.060055	0.053957	0.060056	0.057006	0.060065	0.216915

Table 3. Comparison of our IPSO with respect to CHMPSO[13]

Problem	Optimal	Best Result		Mean Result		Worst Result	
		IPSO	CHMPSO	IPSO	CHMPSO	IPSO	CHMPSO
g01	-15	-15	-15	-15	-15	-15	-15
g02	0.803619	0.803603	0.803432	0.663532	0.790406	0.488776	0.750393
g03	1	1.004987	1.004720	1.004860	1.003814	1.004309	1.002490
g04	-30665.539	-30665.538672	-30665.5	-30665.538672	-30665.5	-30665.538672	-30665.5
g05	5126.4981	5126.498110	5126.64	5126.500182	5461.081333	5126.507464	6104.75
g06	-6961.81388	-6961.813875	-6961.81	-6961.813856	-6961.81	-6961.813794	-6961.81
g07	24.306	24.334560	24.3511	24.961397	25.355771	26.011018	27.3168
g08	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825
g09	680.630	680.630765	680.638	680.668078	680.852393	680.986895	681.553
g10	7049.3307	7251.206603	7057.59	7377.286579	7560.047857	7582.967352	8104.31
g11	0.75	0.749000	0.749999	0.749001	0.750107	0.749003	0.752885
g12	1	1.000000	1	1.000000	1	1.000000	1
g13	0.0539498	0.060055	0.068665	0.060056	1.716426	0.060065	13.6695

Table 4. Comparison of our IPSO with respect to SMES[22]

Problem	Optimal	Best Result		Mean Result		Worst Result	
		IPSO	SMES	IPSO	SMES	IPSO	SMES
g01	-15	-15	-15	-15	-15	-15	-15
g02	0.803619	0.803603	0.803601	0.663532	0.785238	0.488776	0.751322
g03	1	1.004987	1.000	1.004860	1.000	1.004309	1.000
g04	-30665.539	-30665.538672	-30665.539	-30665.538672	-30665.539	-30665.538672	-30665.539
g05	5126.4981	5126.498110	5126.599	5126.500182	5174.492	5126.507464	5304.167
g06	-6961.81388	-6961.813875	-6961.81	-6961.813856	-6961.284	-6961.813794	-6952.482
g07	24.306	24.334560	24.3511	24.961397	24.4751	26.011018	24.843
g08	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825
g09	680.630	680.630765	680.638	680.668078	680.643	680.986895	680.719
g10	7049.3307	7251.206603	7057.59	7377.286579	7253.047	7582.967352	7638.366
g11	0.75	0.749000	0.749999	0.749001	0.75	0.749003	0.75
g12	1	1.000000	1	1.000000	1.000	1.000000	1
g13	0.0539498	0.060055	0.068665	0.060056	0.166385	0.060065	0.468294

References

1. Coello Coello, C.A.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. Comput. Meth. Appl. Mech. Eng. 191, 1245-1287(2002)
2. Michalewicz, Z.: A survey of constraint handling techniques in evolutionary computation methods, in: J.R. McDonnell et al.(Eds.), In: Mcdonnell J.R., Reynolds R.G. and Fogel D.B.

(eds.), Proceedings of the 4th Annual Conference on Evolutionary Programming, pp. 135-155, MIT Press, San Diego (1995)
3. Coello Coello, C.A.: Use of a self-adaptive penalty approach for engineering optimization problems, Comput. Ind. 41,113-127(2000)
4. Koziel, S., Michalewicz, Z.: Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization, Evol. Comput. Vol.7, No. 1, pp. 19-44(1999)
5. Runarsson, T.P., Yao, X.: Stochastic ranking for constrained evolutionary optimization, IEEE Trans. Evol. Comput. Vol.4, No. 3, pp. 284-294(2000)
6. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, pp.1942-1948. Piscataway, NJ (1995)
7. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of 6th International Symposium on Micro Machine and Human Science, pp. 39-43, Nagoya, Japan (1995)
8. Floudas, C.A., Pardalos, P.M.: A collection of test problems for constrained global optimization algorithms. Lext. Notes Comput. Sci. 455, Springer-Verlag(1987)
9. Himmelblau, D.M.: Applied Nonlinear Programming. McGraw-Hill(1972)
10. Parsopoulos, K.E., Vrahatis, M.N.: Particle swarm optimization method for constrained optimization problems. In: Proceedings of the Euro-International Symposium on Computational Intelligence(E-ISCI2002), Slovakia(2002)
11. Lu, H.Y., Chen, W.Q.: Self-adaptive velocity particle swarm optimization for solving constrained optimization problems, Journal of Global Optimization, Vol. 41, No. 3, pp. 427--445, Springer Netherlands(2008)
12. Runarsson, T.P., Yao, X.: Stochastic ranking for constrained evolutionary optimization. IEEE Trans. Evol. Comput.7, pp. 19-44(1999)
13. Toscano Pulido, G., Coello Coello, C.A.: A constraint-handling mechanism for particle swarm optimization. In: Proceedings of the 2004 Congress on Evolutionary Computation, Vol. 2, pp. 1396-1403(2004)
14. He, Q., Wang, L.: A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. Mathematics and computation. Vol.186, pp. 1407-1422(2007)
15. Hu, X., Eberhart, R.C.:Solving constrained nonlinear optimization problems with particle swarm optimization. In: Proceedings of 6th World Multiconference on Systemics, Cybernetics and Informatics(SCI2002), Orlando, USA(2002)
16. Zhang, W.J., Xie, X.F.: DEPSO: hybrid particle swarm with differential evolution operator. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, October, IEEE Trans. Evol. Comput. 4, pp.284-294(2000)
17. Shi, Y.H., Eberhart, R.: A modified particle swarm optimizer. In: Proceedings IEEE International Conference on Evolutionary Computation, pp. 69-73. Anchorage(1998)
18. Kennedy, J., Eberhart, R.C., Shi, Y., Swarm Intelligence, Morgan Kaufman Publishers, San Francisco, CA(2001)
19. Michalewicz, Z., Schoenauer, M.: Evolutionary algorithms for constrained parameter optimization problems. Evol. Comput. 4, pp. 1-32(1996)
20. Muñoz Zavala, A.E., Hernández Aguirre, A., Villa Diharce, E.R.: Constrained optimization via particle evolutionary swarm optimization algorithm (PESO), GECCO'05, pp. 209–216. Washington, DC, USA (2005)
21. Efrén, M.M., Coello Coello, C.A.: A simple multimembered evolution strategy to solve constrained optimization problems. Evolutionary Computation, Vol. 4, No. 1, pp.1-32(1996)
22. Chang, J.F., Chu. S.C., Roddick J.F., Pan J.S.: A Parallel Particle Swarm Optimization Algorithm with Communication Strategies. Journal of Information Science and Engineering, Vol. 21, No. 4, pp. 809-818(2005)
23. Chu, S.C., Pan J.S.: Intelligent Parallel Particle Swarm Optimization Algorithms. In: Nedjah, N., Alba, E., Macedo Mourelle, L.. Studies in Computational Intelligence Series, Vol. 22, pp. 159-175. Springer, Berlin/Heidelberg (2006)

25. Chu, S.C., Tsai, P.W., Pan, J.S.: Parallel Particle Swarm Optimization Algorithms with Adaptive Simulated Annealing. In: Abraham, A., Grosan, C., Ramos, V.. Studies in Computational Intelligence Series. Vol. 31, pp. 261-279. Springer, Berlin/Heidelberg(2006)