CHAPTER 4

Fundamentals of Particle Swarm Optimization Techniques

YOSHIKAZU FUKUYAMA

4.1 INTRODUCTION

Natural creatures sometimes behave as a swarm. One of the main streams of artificial life research is to examine how natural creatures behave as a swarm and reconfigure the swarm models inside a computer. Reynolds developed *boid* as a swarm model with simple rules and generated complicated swarm behavior by computer graphic animation [1]. Boyd and Richerson examined the decision process of human beings and developed the concept of *individual learning and cultural transmission* [2]. According to their examination, human beings make decisions using their own experiences and other persons' experiences.

A new optimization technique using an analogy of swarm behavior of natural creatures was started in the beginning of the 1990s. Dorigo developed ant colony optimization (ACO) based mainly on the social insect, especially ant, metaphor [3]. Each individual exchanges information through pheromones implicitly in ACO. Eberhart and Kennedy developed particle swarm optimization (PSO) based on the analogy of swarms of birds and fish schooling [4]. Each individual exchanges previous experiences in PSO. These research efforts are called *swarm intelligence* [5, 6]. This chapter focuses on PSO as one of the *swarm intelligence* techniques.

Other evolutionary computation (EC) techniques such as genetic algorithms (GAs), utilize multiple searching points in the solution space like PSO. Whereas GAs can treat combinatorial optimization problems, PSO was aimed to treat nonlinear optimization problems with continuous variables originally. Moreover, PSO has been expanded to handle combinatorial optimization problems and both discrete and continuous variables as well. Efficient treatment of mixed-integer nonlinear optimization problems (MINLPs) is one of the most difficult problems in practical optimization. Moreover, unlike other EC techniques, PSO can be realized with only a small

Modern Heuristic Optimization Techniques. Edited by K. Y. Lee and M. A. El-Sharkawi

program; namely, PSO can handle MINLPs with only a small program. This feature of PSO is one of its advantages compared with other optimization techniques.

This chapter is organized as follows: Section 4.2 explains the basic PSO method and Section 4.3 explains variations of PSO. Section 4.4 shows some applications of PSO, and Section 4.5 concludes this chapter with some remarks.

4.2 BASIC PARTICLE SWARM OPTIMIZATION

4.2.1 Background of Particle Swarm Optimization

Swarm behavior can be modeled with a few simple rules. Schools of fishes and swarms of birds can be modeled with such simple models. Namely, even if the behavior rules of each individual (agent) are simple, the behavior of the swarm can be complicated. Reynolds utilized the following three *vectors* as simple rules in the researches on *boid*.

1. Step away from the nearest agent
2. Go toward the destination
3. Go to the center of the swarm

The behavior of each agent inside the swarm can be modeled with simple *vectors*. The research results are one of the basic backgrounds of PSO.

Boyd and Richerson examined the decision process of humans and developed the concept of *individual learning and cultural transmission* [2]. According to their examination, people utilize two important kinds of information in decision process. The first one is their *own experience*; that is, they have tried the choices and know which state has been better so far, and they know how good it was. The second one is *other people's experiences*; that is, they have knowledge of how the other agents around them have performed. Namely, they know which choices their neighbors have found most positive so far and how positive the best pattern of choices was. Each agent decides its decision using its own experiences and the experiences of others. The research results are also one of the basic background elements of PSO.

4.2.2 Original PSO

According to the above background of PSO, Kennedy and Eberhart developed PSO through simulation of bird flocking in a two-dimensional space. The position of each agent is represented by its x, y axis position and also its velocity is expressed by vx (the velocity of x axis) and vy (the velocity of y axis). Modification of the agent position is realized by the position and velocity information.

Bird flocking optimizes a certain objective function. Each agent knows its best value so far (pbest) and its x, y position. This information is an analogy of the *personal experiences* of each agent. Moreover, each agent knows the best value so far in the group (gbest) among pbests. This information is an analogy of *the*

knowledge of how the other agents around them have performed. Each agent tries to modify its position using the following information:

The current positions (x, y),
The current velocities (vx, vy),
The distance between the current position and pbest
The distance between the current position and gbest

This modification can be represented by the concept of velocity (modified value for the current positions). Velocity of each agent can be modified by the following equation:

$$v_i^{k+1} = wv_i^k + c_1\text{rand}_1 \times (\text{pbest}_i - s_i^k) + c_2\text{rand}_2 \times (\text{gbest} - s_i^k) \tag{4.1}$$

where v_i^k is velocity of agent i at iteration k, w is weighting function, c_j is weighting coefficients, rand is random number between 0 and 1, s_i^k is current position of agent i at iteration k, pbest_i is pbest of agent i, and gbest is gbest of the group.

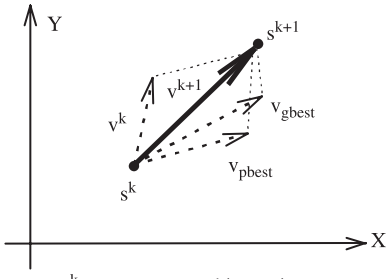
Namely, velocity of an agent can be changed using three *vectors* such like *boid*. The velocity is usually limited to a certain maximum value. PSO using (4.1) is called the Gbest model.

The following weighting function is usually utilized in (4.1):

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{\text{iter}_{\max}} \times \text{iter}, \tag{4.2}$$

where $w_{\max}$ is initial weight, $w_{\min}$ is final weight, $\text{iter}_{\max}$ is maximum iteration number, and iter is current iteration number.

The meanings of the right-hand side (RHS) of (4.1) can be explained as follows [7]. The RHS of (4.1) consists of three terms (*vectors*). The first term is the previous velocity of the agent. The second and third terms are utilized to change the velocity of the agent. Without the second and third terms, the agent will keep on "flying" in the same direction until it hits the boundary. Namely, it tries to explore new areas and, therefore, the first term corresponds with *diversification* in the search procedure. On the other hand, without the first term, the velocity of the "flying" agent is only determined by using its current position and its best positions in history. Namely, the agents will try to converge to the their pbests and/or gbest and, therefore, the terms correspond with *intensification* in the search procedure. As shown below, for example, $w_{\max}$ and $w_{\min}$ are set to 0.9 and 0.4. Therefore, at the beginning of the search procedure, *diversification* is heavily weighted, while *intensification* is heavily weighted at the end of the search procedure such like simulated annealing (SA). Namely, a certain velocity, which gradually gets close to pbests and gbest, can be calculated. PSO using (4.1), (4.2) is called *inertia weights approach* (IWA).



s^k : current searching point,
s^{k+1} : modified searching point,
V^k : current velocity,
V^{k+1} : modified velocity,
V_{pbest} : velocity based on pbest
V_{gbest} : velocity based on gbest

FIGURE 4.1 Concept of modification of a searching point by PSO.

The current position (searching point in the solution space) can be modified by the following equation:

$$s_i^{k+1} = s_i^k + v_i^{k+1}. \tag{4.3}$$

Figure 4.1 shows a concept of modification of a searching point by PSO, and Fig. 4.2 shows a searching concept with agents in a solution space. Each agent changes its current position using the integration of vectors as shown in Fig. 4.1.

The general flowchart of PSO with IWA can be described as follows:

Step 1. Generation of initial condition of each agent. Initial searching points (s_i^0) and velocities (v_i^0) of each agent are usually generated randomly within the allowable range. The current searching point is set to pbest for each agent. The best-evaluated value of pbest is set to gbest, and the agent number with the best value is stored.

Step 2. Evaluation of searching point of each agent. The objective function value is calculated for each agent. If the value is better than the current pbest of the agent,

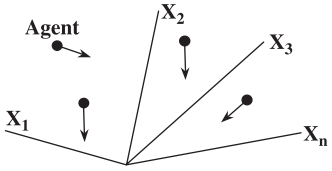


FIGURE 4.2 Searching concept with agents in a solution space by PSO.

the pbest value is replaced by the current value. If the best value of pbest is better than the current gbest, gbest is replaced by the best value and the agent number with the best value is stored.

Step 3. Modification of each searching point. The current searching point of each agent is changed using (4.1), (4.2), and (4.3).

Step 4. Checking the exit condition. The current iteration number reaches the predetermined maximum iteration number, then exits. Otherwise, the process proceeds to step 2.

Figure 4.3 shows the general flowchart of PSO. The features of the searching procedure of PSO can be summarized as follows:

(a) As shown in (4.1), (4.2), and (4.3), PSO can essentially handle continuous optimization problems.

(b) PSO utilizes several searching points, and the searching points gradually get close to the optimal point using their pbests and the gbest.

(c) The first term of the RHS of (4.1) corresponds with diversification in the search procedure. The second and third terms correspond with intensification in the search procedure. Namely, the method has a well-balanced mechanism to utilize diversification and intensification in the search procedure efficiently.

(d) The above concept is explained using only the x, y axis (two-dimensional space). However, the method can be easily applied to n-dimensional problems. Namely, PSO can handle continuous optimization problems with continuous state variables in a n-dimensional solution space.

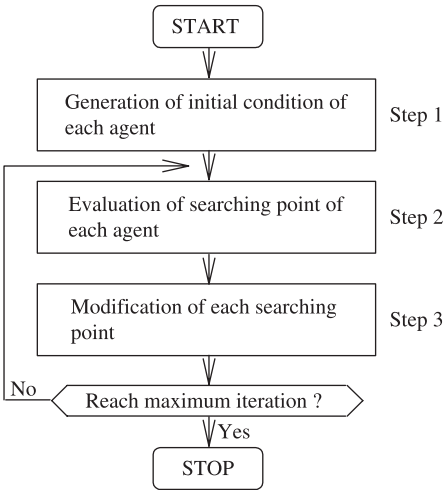


FIGURE 4.3 A general flowchart of PSO.

Shi and Eberhart tried to examine the parameter selection of the above parameters [7, 8]. According to their examination, the following parameters are appropriate and the values do *not* depend on problems:

$$c_i = 2.0, \qquad w_{\max} = 0.9, \qquad w_{\min} = 0.4.$$

The values are also proved to be appropriate for power system problems [9, 10]. The basic PSO has been applied to a learning problem of neural networks and Schaffer f6, a famous benchmark function for GA, and the efficiency of the method has been observed [4].

4.3 VARIATIONS OF PARTICLE SWARM OPTIMIZATION

4.3.1 Discrete PSO

The original PSO described in Section 4.2 treats nonlinear optimization problems with continuous variables. However, practical engineering problems are often formulated as combinatorial optimization problems. Kennedy and Eberhart developed a discrete binary version of PSO for these problems [11]. They proposed a model wherein the probability of an agent's deciding yes or no, true or false, or making some other decision is a function of personal and social factors as follows:

$$P(s_i^{k+1} = 1) = f(s_i^k, v_i^k, \text{pbest}_i, \text{gbest}). \tag{4.4}$$

The parameter v, an agent's tendency to make one or the other choice, will determine a probability threshold. If v is higher, the agent is more likely to choose 1, and lower values favor 0 choice. Such a threshold requires staying in the range [0, 1]. One of the functions accomplishing this feature is the sigmoid function, which is usually utilized in artificial neural networks.

$$\text{sig}(v_i^k) = \frac{1}{1 + \exp(-v_i^k)}. \tag{4.5}$$

The agent's tendency should be adjusted for success of the agent and the group. In order to accomplish this, a formula for each v_i^k that will be some function of the difference between the agent's current position and the best positions found so far by itself and by the group should be developed. Namely, like the basic continuous version, the formula for the binary version of PSO can be described as follows:

$$v_i^{k+1} = v_i^k + \text{rand}_1 \times (\text{pbest}_i - s_i^k) + \text{rand}_2 \times (\text{gbest} - s_i^k) \tag{4.6}$$

$$\begin{aligned} \rho_i^{k+1} < \text{sig}(v_i^{k+1}) \quad & \text{then } s_i^{k+1} = 1; \\ & \text{else } s_i^{k+1} = 0, \end{aligned} \tag{4.7}$$

where rand is a positive random number drawn from a uniform distribution with a predefined upper limit, and ρ_i^{k+1} is a vector of random numbers of [0.0, 1.0].

These formulas are iterated repeatedly over each dimension of each agent. The second and third term of the RHS of (4.6) can be weighted like the basic continuous version of PSO. v_i^k can be limited so that $\text{sig}(v_i^k)$ does not approach too closely to 0.0 or 1.0. This ensures that there is always some chance of a bit flipping. A constant parameter $V_{\max}$ (limited value of v_i^k) can be set at the start of a trial. In practice, $V_{\max}$ is often set in $[-4.0, +4.0]$. The entire algorithm of the binary version of PSO is almost the same as that of the basic continuous version except for the above state equations.

4.3.2 PSO for MINLPs

Engineering problems often pose both discrete and continuous variables using nonlinear objective functions. Kennedy and Eberhart discussed the integration of binary and continuous versions of PSO [6]. Fukuyama et al. presented a PSO for MINLPs by modifying the continuous version of PSO [9].

The method of Ref. 9 can be described briefly as follows. Discrete variables can be handled in (4.1) and (4.3) with little modification. Discrete numbers instead of continuous numbers can be used to express the current position and velocity. Namely, a discrete random number is used for rand in (4.1), and the whole calculation of RHS of (4.1) is discretized to the existing discrete number. Using this modification for discrete numbers, both continuous and discrete numbers can be handled in the algorithm with no inconsistency. In Ref. 9, the PSO for MINLPs was applied successfully to a reactive power and voltage control problem in electric power systems with promising results. The details can be found in Chapter 16.

4.3.3 Constriction Factor Approach (CFA)

The basic system equation of PSO [(4.1), (4.2), and (4.3)] can be considered as a kind of difference equation. Therefore, the system dynamics, that is, the search procedure, can be analyzed using eigenvalues of the difference equation. Actually, using a simplified state equation of PSO, Clerc and Kennedy developed CFA of PSO by eigenvalues [6, 12]. The velocity of the constriction factor approach (simplest constriction) can be expressed as follows instead of (4.1) and (4.2):

$$v_i^{k+1} = K[v_i^k + c_1 \times \text{rand}_1 \times (\text{pbest}_i - s_i^k) + c_2 \times \text{rand}_2 \times (\text{gbest} - s_i^k)] \tag{4.8}$$

$$K = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \quad \text{where } \varphi = c_1 + c_2,\ \varphi > 4, \tag{4.9}$$

where φ and K are coefficients.

For example, if $\varphi = 4.1$, then $K = 0.73$. As φ increases above 4.0, K gets smaller. For example, if $\varphi = 5.0$, then $K = 0.38$, and the damping effect is even more

pronounced. The convergence characteristic of the system can be controlled by φ. Namely, Clerc et al. found that the system behavior can be controlled so that the system behavior has the following features:

(a) The system does not diverge in a real-valued region and finally can converge.
(b) The system can search different regions efficiently by avoiding premature convergence.

The whole PSO algorithms by IWA and CFA are the same except that CFA utilizes a different equation for calculation of velocity [(4.8) and (4.9)]. Unlike other EC methods, PSO with CFA ensures the convergence of the search procedures based on mathematical theory. PSO with CFA can generate higher-quality solutions for some problems than PSO with IWA [10, 13]. However, CFA only considers dynamic behavior of only one agent and studies on the effect of the interaction among agents. Understanding the effects of pbests and gbest in the system dynamics remains for future work.

4.3.4 Hybrid PSO (HPSO)

Angeline developed HPSO by combining the selection mechanism of EC techniques with PSO [14]. The number of highly evaluated agents is increased whereas the number of lowly evaluated agents is decreased at each iteration by HPSO. Because the PSO search procedure depends greatly on pbest and gbest, the searching area is limited by pbest and gbest. However, the effect of pbest and gbest is gradually vanished by the selection mechanism, and a broader area search can be realized by HPSO. Agents with low evaluation values are replaced by those with high evaluation values using the selection. The replaced rate is called selection rate (Sr). It should be noted that the pbest information of each agent is maintained even if the agent position is replaced to another agent's position. Therefore, intensive search in a current attractive area and dependence on the past high-evaluation position are realized. Figure 4.4 shows a general flowchart of HPSO. As shown in the figure, only step 3 is added to the original PSO. Figure 4.5 shows the concepts of steps 2, 3, and 4 of the general flowchart.

The original PSO changes the current search point using the state equations (4.1), (4.2), and (4.3) step by step. Therefore, it sometimes takes time to get into an *attractive area* in the solution space. On the contrary, HPSO moves the lowly evaluated agents to the attractive area directly using the selection method, and concentrated search especially in the current attractive area is realized. However, a jump to the attractive area does not always lead to effective search, and a more attractive area may exist between the current searching point and the jumped area. As Angeline pointed out in Ref. 14, HPSO does not always work better than the original PSO. HPO with IWA calculates velocity and new searching points using (4.1), (4.2), and (4.3), whereas HPSO with CFA calculates them using (4.8), (4.9), and (4.3).

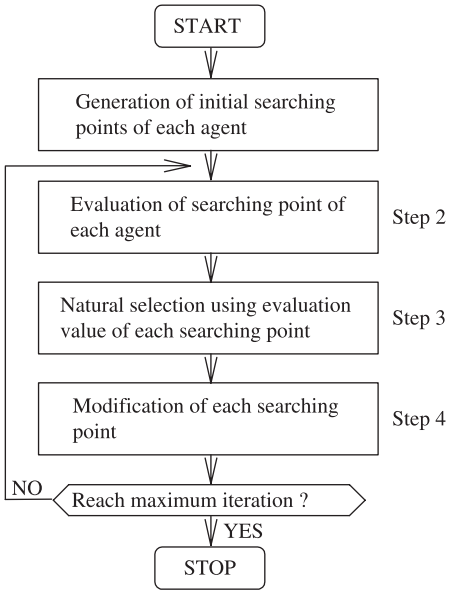


FIGURE 4.4 A general flowchart of HPSO.

4.3.5 Lbest Model

Eberhart and Kennedy also developed an *lbest model* [6]. In the model, agents have information only of their own and their nearest array neighbors' bests (lbests), rather than that of the entire group. Namely, in (4.1), gbest is replaced by lbests in the model. For example, lbest of agent no. 3 can be determined as the best pbest among pbests of agents no. 2, 3, and 4.

$$v_i^{k+1} = wv_i^k + c_1\text{rand}_1 \times (\text{pbest}_i - s_i^k) + c_2\text{rand}_2 \times (\text{lbest}_i - s_i^k), \tag{4.10}$$

where lbest_i is local best of agent i.

The whole PSO algorithms by lbest model and gbest model are the same except that lbest model utilizes a different equation for calculation of velocity [(4.10)].

4.3.6 Adaptive PSO (APSO)

The following points are improved to the original PSO with IWA.

1. The search trajectory of PSO can be controlled by introducing the new parameters (P_1, P_2) based on the probability to move close to the position of (pbest, gbest) at the following iteration.
2. The wv_i^k term of (4.1) is modified as (4.12). Using the equation, the center of the range of particle movements can be equal to gbest.

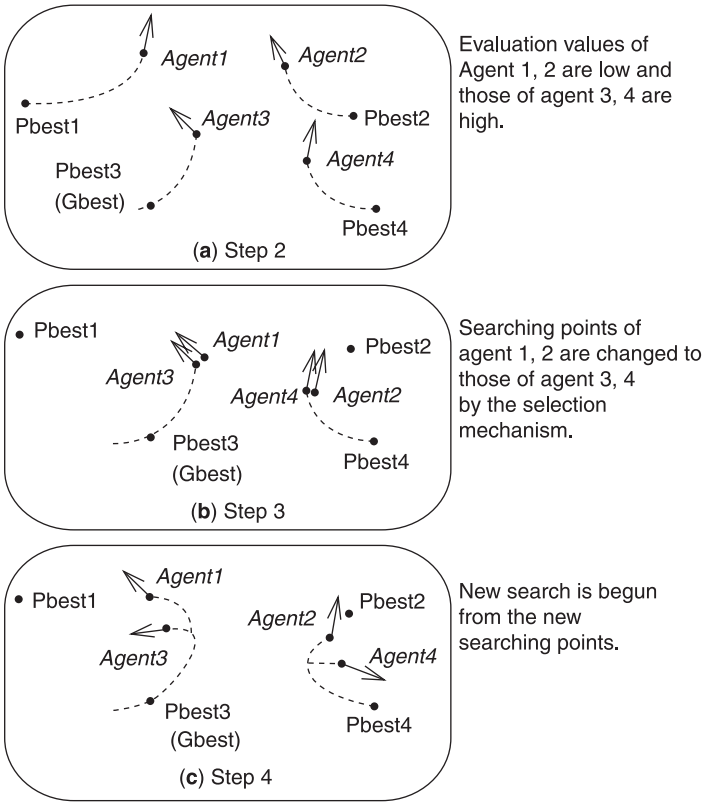


FIGURE 4.5 Concept of searching process by HPSO.

3. When the agent becomes gbest, it is perturbed. The new parameters (P_1, P_2) of the agent are adjusted so that the agent may move away from the position of (pbest, gbest).
4. When the agent is moved beyond the boundary of feasible regions, pbests and gbest cannot be modified.
5. When the agent is moved beyond the boundary of feasible regions, the new parameters (P_1, P_2) of the agent are adjusted so that the agent may move close to the position of (pbest, gbest).

The new parameters are set to each agent. The weighting coefficients are calculated as follows:

$$c_2 = \frac{2}{P_1}, \qquad c_1 = \frac{2}{P_2} - c_2, \tag{4.11}$$

The search trajectory of PSO can be controlled by the parameters (P_1, P_2). Concretely, when the value is enlarged more than 0.5, the agent may move close to the position of pbest/gbest.

$$w = \text{gbest} - (\{c_1(\text{pbest} - x) + c_2(\text{gbest} - x)\}/2 + x). \tag{4.12}$$

Namely, the velocity of the improved PSO can be expressed as follows:

$$v_i^{k+1} = w_i + c_1\text{rand}_1 \times (\text{pbest}_i - s_i^k) + c_2\text{rand}_2 \times (\text{gbest} - s_i^k). \tag{4.13}$$

The improved PSO can be expressed as follows (steps 1 and 5 are the same as PSO):

1. *Generation of initial searching points*: Basic procedures are the same as PSO. In addition, the parameters (P_1, P_2) of each agent are set to 0.5 or higher. Then, each agent may move close to the position of (pbest, gbest) at the following iteration.
2. *Evaluation of searching points*: The procedure is the same as PSO. In addition, when the agent becomes gbest, it is perturbed. The parameters (P_1, P_2) of the agent are adjusted to 0.5 or lower so that the agent may move away from the position of (pbest, gbest).
3. *Modification of searching points*: The current searching points are modified using the state equations (4.13), (4.3) of adaptive PSO.

4.3.7 Evolutionary PSO (EPSO)

The idea behind EPSO [16, 17] is to grant a PSO scheme with an explicit selection procedure and with self-adapting properties for its parameters. At a given iteration, consider a set of solutions or alternatives that we will keep calling particles. The general scheme of EPSO is the following:

1. REPLICATION: each particle is replicated R times
2. MUTATION: each particle has its weights mutated
3. REPRODUCTION: each mutated particle generates an offspring according to the particle movement rule
4. EVALUATION: each offspring has its fitness evaluated
5. SELECTION: by stochastic tournament the best particles survive to form a new generation.

The movement rule for EPSO is the following: given a particle s_i^k, a new particle s_i^{k+1} results from

$$s_i^{k+1} = s_i^k + v_i^{k+1} \tag{4.14}$$

$$v_i^{k+1} = w_{i0}^* v_i^k + w_{i1}^* \left(\text{pbest}_i - s_i^k\right) + w_{i2}^* \left(\text{gbest}^* - s_i^k\right). \tag{4.15}$$

So far, this seems like PSO—the movement rule keeps its terms of inertia, memory, and cooperation. However, the weights undergo mutation

$$w_{ik}^{*} = w_{ik} + \tau N(0, 1), \tag{4.16}$$

where $N(0, 1)$ is a random variable with Gaussian distribution, 0 mean, and variance 1; and the global best gbest is randomly disturbed to give

$$\text{gbest}^{*} = \text{gbest} + \tau' N(0, 1). \tag{4.17}$$

The τ, τ' are learning parameters (either fixed or treated also as strategic parameters and therefore also subject to mutation).

This scheme benefits from two "pushes" in the right direction: the Darwinistic process of selection and the particle movement rule, and therefore it is natural to expect that it may display advantageous convergence properties when compared with ES or PSO alone. Furthermore, EPSO can also be classified as a self-adaptive algorithm, because it relies on the mutation and selection of strategic parameters, just as any σ-SA evolution strategy.

4.4 RESEARCH AREAS AND APPLICATIONS

References 18 to 65 show other PSO-related papers. Most of these papers are related to PSO itself and its modification and comparison with other EC methods. PSO is a new EC technique, and there are a few applications. Table 4.1 shows early applications of PSO in general fields. The last eight applications are in power and energy system fields. Detailed descriptions of applications to reactive power and voltage control [9, 60] can be found in Chapter 16. Optimal operation of cogeneration [31] has actually been operated in some factories of an automobile company in Japan.

TABLE 4.1 PSO Applications

Application Field	Ref. No.
Neural network learning algorithm	[29, 51]
Human tremor analysis	[23]
Rule extraction in fuzzy neural network	[32]
Battery pack state-of-charge estimation	[6]
Computer numerically controlled milling optimization	[59]
Reactive power and voltage control	[9, 60, 64]
Distribution state estimation	[10]
Power system stabilizer design	[18]
Fault state power supply reliability enhancement	[46]
Dynamic security assessment	[61]
Economic dispatch	[62, 63]
Short-term load forecasting	[65]
Optimal operation of cogeneration	[31]

However, applications of PSO to various fields are at the early stage. More applications can be expected.

4.5 CONCLUSIONS

This chapter presented fundamentals of PSO techniques. Whereas many evolutionary computation techniques have been developed for combinatorial optimization problems, PSO was developed originally for nonlinear optimization problems with continuous variables. PSO has several variations, including integration with selection mechanism and hybridization for handling both discrete and continuous variables. Moreover, a recently developed constriction factor approach is based on mathematical analysis and useful for obtaining high-quality solutions. Advanced PSOs such as adaptive PSO (APSO) and evolutionary PSO (EPSO) have been developed. A few applications have already appeared using PSO including practical applications. PSO can be an efficient optimization tool for nonlinear continuous optimization problems, combinatorial optimization problems, and mixed-integer nonlinear optimization problems (MINLPs).

REFERENCES

1. Reynolds C. Flocks, herds, and schools: A distributed behavioral model. Computer Graphics 1987; 21(4):25–34.
2. Boyd R, Richerson PJ. Culture and the evolutionary process. Chicago: University of Chicago Press; 1985.
3. Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies. Proceedings of First European Conference on Artificial Life. Cambridge, MA: MIT Press; 1991. p. 134–142.
4. Kennedy J, Eberhart R. Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks (ICNN'95) Perth, Australia: IEEE Press; 1995. Vol. IV. p. 1942–1948.
5. Bonabeau E, Dorigo M, Theraulaz G. Swarm intelligence: From natural to artificial systems. Oxford: Oxford University Press; 1999.
6. Kennedy J, Eberhart R. Swarm intelligence. San Mateo, CA: Morgan Kaufmann; 2001.
7. Shi Y, Eberhart R. A modified particle swarm optimizer. Proceedings of IEEE International Conference on Evolutionary Computation (ICEC'98). Anchorage: IEEE Press; 1998. p. 69–73.
8. Shi Y, Eberhart R. Parameter selection in particle swarm optimization. Proceedings of the 1998 Annual Conference on Evolutionary Programming. San Diego: MIT Press; 1998.
9. Fukuyama Y. et al. A particle swarm optimization for reactive power and voltage control considering voltage security assessment. IEEE Trans Power Systems 2000; 15(4):1232–1239.
10. Naka S, Genji T, Yura T, Fukuyama Y. A hybrid particle swarm optimization for distribution state estimation. IEEE Trans Power Systems 2003; 18(1):60–68.

11. Kennedy J, Eberhart R. A discrete binary version of the particle swarm optimization algorithm. Proceedings of the 1997 Conference on Systems, Man, and Cybernetics (SMC'97). IEEE Press; 1997. p. 4104–4109.
12. Clerc M, Kennedy J. The particle swarm—explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evol Computat 2002; 6(1):58–73.
13. Eberhart R, Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization. Proceedings of the Congress on Evolutionary Computation (CEC2000). IEEE Press; 2000. p. 84–88.
14. Angeline P. Using selection to improve particle swarm optimization. Proceedings of IEEE International Conference on Evolutionary Computation (ICEC'98). Anchorage: IEEE Press; 1998.
15. Ide A, Yasuda K. The improvement of search trajectory of particle swarm optimization. National Convention Record I.E.E. Japan: IEEE of Japan; 2003. p. 41–42. (in Japanese.)
16. Miranda V, Fonseca N. New evolutionary particle swarm algorithm (EPSO) applied to voltage/var control. Proceedings of PSCC'02—Power System Computation Conference. Seville, Spain, Butterworth-Heinemann; June 24–28; 2002.
17. Miranda V, Fonseca N. EPSO—Best of two world of meat-heuristic applied to power system problems. Proceedings of the 2002 Congress of Evolutionary Computation: IEEE Press; 2002.
18. Abido MA. Particle swarm optimization for multimachine power system stabilizer design. Proceedings of IEEE Power Engineering Society Summer Meeting: IEEE Press; 2001.
19. Angeline P. Evolutionary optimization versus particle swarm optimization: philosophy and performance differences. Proceeding of the Seventh Annual Conference on Evolutionary Programming: IEEE Press; 1998.
20. Carlisle A, Dozier G. Adapting particle swarm optimization to dynamic environments. Proceedings of International Conference on Artificial Intelligence. Monte Carlo Resort, Las Vegas, Nevada Computer Science Research, Education, and Applications Press; 2000.
21. Carlisle A, Dozier G. An off-the-shelf particle swarm optimization. Proceedings of the Workshop on Particle Swarm Optimization. Indianapolis, IN: Purdue School of Engineering and Technology, 2001.
22. Eberhart R, Kennedy J. A new optimizer using particle swarm theory. Proceedings of the Sixth International Symposium on Micro Machine and Human Science. Nagoya, Japan. Piscataway, NJ: IEEE Service Center; 1995. p. 39–43.
23. Eberhart R, Hu X. Human tremor analysis using particle swarm optimization. Proceedings of Congress on Evolutionary Computation (CEC'99). Washington, DC. Piscataway, NJ: IEEE Service Center; 1999. p. 1927–1930.
24. Eberhart R, Shi Y. Comparison between genetic algorithms and particle swarm optimization. Proceedings of the Seventh Annual Conference on Evolutionary Programming; IEEE Press; 1998.
25. Eberhart R, Shi Y. Evolving artificial neural networks. Proceedings of International Conference on Neural Networks and Brain. Beijing, P.R.C., PL5-PL13; Publishing House of Electronics Industry; 1998.
26. Eberhart R, Shi Y. Comparison between genetic algorithms and particle swarm optimization. In: Porto VW, Saravanan N, Waagen D, Eiben AE, eds. Evolutionary programming VII: Proceedings 7th Annual Conference on Evolutionary Programming. San Diego, CA. Berlin: Springer-Verlag; 1998.

27. Eberhart R, Shi Y. Tracking and optimizing dynamic systems with particle swarms. Proceedings of Congress on Evolutionary Computation (CEC2001), Seoul, Korea. Piscataway, NJ: IEEE Service Center; 2001.
28. Eberhart R, Shi Y. Particle swarm optimization: developments, applications and resources. Proceedings of Congress on Evolutionary Computation (CEC2001). Seoul, Korea, Piscataway, NJ: IEEE Service Center; 2001.
29. Eberhart R, Simpson P, Dobbins R. Computational intelligence PC tools. Boston: Academic Press Professional; 1996.
30. Fan H-Y, Shi Y. Study of Vmax of the particle swarm optimization algorithm. Proceedings of the Workshop on Particle Swarm Optimization. Indianapolis, IN: Purdue School of Engineering and Technology; 2001.
31. Fukuyama Y, et al. Particle swarm optimization for optimal operational planning of a cogeneration system. Proceedings of the 4th IASTED International Conference on Modelling, Simulation, and Optimization (MSO2004); ACTA Press; 2004.
32. He Z, Wei C, Yang L, Gao X, Yao S, Eberhart R, Shi Y. Extracting rules from fuzzy neural network by particle swarm optimization. Proceedings of IEEE International Conference on Evolutionary Computation (ICEC'98). Anchorange, Alaska, IEEE Press; May 4–9; 1998.
33. Ismail A, Engelbrecht AP. Training product units in feedforward neural networks using particle swarm optimization. Proceedings of the International Conference on Artificial Intelligence. Durban, South Africa; CSREA Press; 1999. p. 36–40.
34. Kennedy J. The particle swarm: Social adaptation of knowledge. Proceedings of International Conference on Evolutionary Computation (CEC'97). Indianapolis, IN: Piscataway, NJ: IEEE Service Center; 1997. p. 303–308.
35. Kennedy J. Minds and cultures: Particle swarm implications. Socially intelligent agents: Papers from the 1997 AAAI Fall Symposium. Technical report FS-97-02, 67–72. Menlo Park, CA: AAAI Press; 1997.
36. Kennedy J. Methods of agreement: Inference among the elementals. Proceedings of International Symposium on Intelligent Control. Piscataway, NJ: IEEE Service Center; 1998.
37. Kennedy J. The behavior of particles. In: Porto VW, Saravanan N, Waagen D, Eiben AE, eds. Evolutionary programming VII: Proceedings 7th Annual Conference on Evolutionary Programming Conference. San Diego, CA. Berlin: Springer-Verlag; 1998. p. 581–589.
38. Kennedy J. Thinking is social: Experiments with the adaptive culture model. J Conflict Resolution 1998; 42:56–76.
39. Kennedy J. Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. Proceedings of Congress on Evolutionary Computation (CEC'99), 1931–1938. Piscataway, NJ: IEEE Service Center; 1999.
40. Kennedy J. Stereotyping: Improving particle swarm performance with cluster analysis. Proceedings of the 2000 Congress on Evolutionary Computation (CEC2000). San Diego, CA. Piscataway, NJ: IEEE Press; 2000.
41. Kennedy J. Out of the computer, into the world: externalizing the particle swarm. Proceedings of the Workshop on Particle Swarm Optimization. Indianapolis, IN: Purdue School of Engineering and Technology; 2001.
42. Kennedy J, Eberhart R. The particle swarm: Social adaptation in information processing systems. In: Corne D, Dorigo M, Glover F, eds. New ideas in optimization. London: McGraw-Hill; 1999.

43. Kennedy J, Spears W. Matching algorithms to problems: An experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. Proceedings of IEEE International Conference on Evolutionary Computation (ICEC'98). Anchorage, Alaska, IEEE Press; May 4–9; 1998.
44. Løvbjerg M, Rasmussen T, Krink T. Hybrid particle swarm optimization with breeding and subpopulations. Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO-2001); Morgan Kaufmann Press; 2001.
45. Mohan C, Al-kazemi B. Discrete particle swarm optimization. Proceedings of the Workshop on Particle Swarm Optimization. Indianapolis, IN: Purdue School of Engineering and Technology; 2001.
46. Nara K, Mishima Y. Particle swarm optimisation for fault state power supply reliability enhancement. Proceedings of IEEE International Conference on Intelligent Systems Applications to Power Systems (ISAP2001). Budapest; IEEE Press; 2001.
47. Ozcan E, Mohan C. Analysis of a simple particle swarm optimization system. Intelligent Engineering Systems Through Artificial Neural Networks 1998; 8:253–258.
48. Ozcan E, Mohan C. Particle swarm optimization: surfing the waves. Proceedings of 1999 Congress on Evolutionary Computation (CEC'99). Washington, DC, Institute of Electrical & Electronics Enginee, July 6–9, 1999.
49. Parsopoulos K, Plagianakos V, Magoulas G, Vrahatis M. Stretching technique for obtaining global minimizers through particle swarm optimization. Proceedings of the Workshop on Particle Swarm Optimization. Indianapolis, IN: Purdue School of Engineering and Technology; 2001.
50. Ray T, Liew KM. A swarm with an effective information sharing mechanism for unconstrained and constrained single objective optimization problems. Proceedings of the 2001 Congress on Evolutionary Computation (CEC2001). Seoul, Korea; IEEE Press; 2001.
51. Salerno J. Using the particle swarm optimization technique to train a recurrent neural model. Proceedings of 9th International Conference on Tools with Artificial Intelligence (ICTAI'97). IEEE Press, 1997.
52. Secrest B, Lamont G. Communication in particle swarm optimization illustrated by the traveling salesman problem. Proceedings of the Workshop on Particle Swarm Optimization. Indianapolis, IN: Purdue School of Engineering and Technology; 2001.
53. Shi Y, Eberhart R. Parameter selection in particle swarm optimization. In: Evolutionary programming VII: Proceedings EP98. New York: Springer-Verlag; 1998. p. 591–600.
54. Shi Y, Eberhart R. Empirical study of particle swarm optimization. Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99). Piscataway, NJ: IEEE Service Center; 1999. p. 1945–1950.
55. Shi Y, Eberhart R. Experimental study of particle swarm optimization. Proceedings of SCI2000 Conference. Orlando, FL; IIIS Press; 2000.
56. Shi Y, Eberhart R. Fuzzy adaptive particle swarm optimization. Proceedings of Congress on Evolutionary Computation (CEC2001). Seoul, South Korea. Piscataway, NJ: IEEE Service Center; 2001.
57. Shi Y, Eberhart R. Particle swarm optimization with fuzzy adaptive inertia weight. Proceedings of the Workshop on Particle Swarm Optimization. Indianapolis, IN: Purdue School of Engineering and Technology; 2001.
58. Suganthan P. Particle swarm optimiser with neighbourhood operator. Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99). Piscataway, NJ: IEEE Service Center; 1999. p. 1958–1962.

59. Tandon V. Closing the gap between CAD/CAM and optimized CNC end milling. Master's thesis, Purdue School of Engineering and Technology, Indiana University Purdue University Indianapolis; 2000.

60. Yoshida H, Kawata K, Fukuyama Y, Nakanishi Y. A particle swarm optimization for reactive power and voltage control considering voltage stability. In: Torres GL, Alves da Silva AP, eds. Proceedings of International Conference on Intelligent System Application to Power Systems (ISAP'99), Rio de Janeiro, Brazil; IEEE Press; 1999; p. 117–121.

61. Kassabalidis I, El-sharkawi M, et al. Dynamic security border identification using enhanced particle swarm optimization. IEEE Trans Power Systems 2002; 17(3):723–729.

62. Gaing Z-L. Particle swarm optimizatin to solving the economic dispatch considering the generator constraints. IEEE Trans Power Systems 2003; 18(3):1187–1195.

63. Park J-B, Lee KY, et al. A particle swarm optimization for economic dispatch with nonsmooth cost functions. IEEE Trans Power Systems 2005; 20(1):34–42.

64. Zhao B, et al. A multiagent-based particle swarm optimization approach for optimal reactive power dispatch. IEEE Trans Power Systems 2005; 20(20):1070–1078.

65. Huang C, et al. A particle swarm optimization to identifying the ARMAX model from short-term load forecasting. IEEE Trans Power Systems 2005; 20(2):1126–1133.