Abstract: This chapter presents fundamentals of particle swarm optimization (PSO) techniques. While a lot of evolutionary computation techniques have been developed for combinatorial optimization problems, PSO has been basically developed for continuous optimization problem, based on the backgrounds of artificial life and psychological research. PSO has several variations including integration with selection mechanism and hybridization for handling both discrete and continuous variables. Moreover, recently developed constriction factor approach is useful for obtaining high quality solutions.

Key words: Continuous optimization problem, Mixed-integer nonlinear optimization problem, Constriction factor

1. INTRODUCTION

Natural creatures sometimes behave as a swarm. One of the main streams of artificial life researches 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 CG animation [1].

From the beginning of 90's, new optimization technique researches using analogy of swarm behavior of natural creatures have been started. Dorigo developed ant colony optimization (ACO) mainly based on the social insect, especially ant, metaphor [2]. Each individual exchanges information through pheromone implicitly in ACO. Eberhart and Kennedy developed particle swarm optimization (PSO) based on the analogy of swarm of bird and fish school [3]. Each individual exchanges previous experiences in PSO. These researches are called "Swarm Intelligence" [4][5]. This chapter describes mainly about PSO as one of swarm intelligence techniques.

Other evolutionary computation (EC) techniques such as genetic algorithm (GA) also utilize some searching points in the solution space. While GA can handle combinatorial optimization problems, PSO can handle continuous optimization problems originally. 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 (MINLP) is one of the most difficult problems in optimization field. Moreover, unlike other EC techniques, PSO can be realized with only small program. Namely PSO can handle MINLP with only small program. The feature of PSO is one of the advantages compared with other optimization techniques.

This chapter is organized as follows: Chapter II explains basic PSO method and chapter III explains variation of PSO such as discrete PSO and hybrid PSO. Chapter IV describes parameter sensitivities and constriction factor approach. Chapter V shows some applications of PSO and Chapter VI concludes this chapter with some remarks.

2. BASIC PARTICLE SWARM OPTIMIZATION

2.1 Background of Particle Swarm Optimization

Natural creatures sometimes behave as a swarm. One of the main streams of artificial life researches is to examine how natural creatures behave as a swarm and reconfigure the swarm models inside a computer. Swarm behavior can be modeled with a few simple rules. School of fishes and swarm 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 called this kind of agent as boid and generated complicated swarm behavior by CG animation [1]. He utilized the following three vectors as simple rules.

1) to step away from the nearest agent
2) to go toward the destination
3) to go to the center of the swarm

Namely, behavior of each agent inside the swarm can be modeled with simple vectors. This characteristic is one of the basic concepts of PSO.

Boyd and Richerson examine the decision process of human being and developed the concept of individual learning and cultural transmission [6]. 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 are most positive so far and how positive the best pattern of choices was. Namely each agent decides his decision using his own experiences and other peoples' experiences. This characteristic is another basic concept of PSO.

2.2 Basic method

According to the background of PSO and simulation of swarm of bird, Kennedy and Eberhart developed a PSO concept. Namely, PSO is basically developed through simulation of bird flocking in two-dimension space. The position of each agent is represented by XY axis position and also the 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 XY position. This information is analogy of personal experiences of each agent. Moreover, each agent knows the best value so far in the group (gbest) among pbests. This information is analogy of knowledge of how the other agents around them have performed. Namely, 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. Velocity of each agent can be modified by the following equation:

\[
v_{k+1}^{i} = v_{k}^{i} + c_{i} \cdot \text{rand} \cdot (p_{k}^{i} - s_{k}^{i}) + c_{i} \cdot \text{rand} \cdot (g_{k} - s_{k}^{i})
\]

(1)

where, \(v_{k}^{i}\) : velocity of agent \(i\) at iteration \(k\),
\(w\) : weighting function,
\(c_{i}\) : weighting factor,
\(\text{rand}\) : random number between 0 and 1,
\(s_{k}^{i}\) : current position of agent \(i\) at iteration \(k\),
\(p_{k}^{i}\) : pbest of agent \(i\),
\(g_{k}\) : gbest of the group.

The following weighting function is usually utilized in (1):

\[
w = w_{\text{max}} - \frac{w_{\text{min}} - w_{\text{max}}}{\text{iter}_{\text{max}}} \times \text{iter}
\]

(2)

where, \(w_{\text{max}}\) : initial weight,
\(w_{\text{min}}\) : final weight,
\(\text{iter}_{\text{max}}\) : maximum iteration number,
\(\text{iter}\) : current iteration number.

Using the above equation, a certain velocity, which gradually gets close to pbest and gbest can be calculated. The current position (searching point in the solution space) can be modified by the following equation:

\[s_{k+1}^{i} = s_{k}^{i} + v_{k+1}^{i}\]

(3)

Fig. 2 shows a concept of modification of a searching point by PSO and Fig. 3 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. 2.

The general flow chart of PSO can be described as follows:

Step 1 Generation of initial condition of each agent
Initial searching points (\(s_{0}^{i}\)) and velocities (\(v_{0}^{i}\)) 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, 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 (1)(2)(3).

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

Fig. 4 shows the general flow chart of PSO. The features of the searching procedure of PSO can be summarized as follows:

(a) As shown in (1)(2)(3), PSO can essentially handle continuous optimization problem.
(b) PSO utilizes several searching points like genetic algorithm (GA) and the searching points gradually get close to the optimal point using their pbest and the gbest.
(c) The first term of right-hand side (RHS) of (1) is corresponding to diversification in the search procedure. The second and third terms of that are corresponding to intensification in the search procedure. Namely, the method has a well-balanced mechanism to utilize...
(d) The above concept is explained using only XY-axis (two-dimension space). However, the method can be easily applied to n-dimension problem. Namely, PSO can handle continuous optimization problems with continuous state variables in a n-dimension solution space.

The above feature (c) can be explained as follows [7]. The RHS of (2) consists of three terms. 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 is corresponding to 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 are corresponding to intensification in the search procedure. The basic PSO has been applied to a learning problem of neural networks and Schaffer f6, the famous benchmark function for GA, and efficiency of the method has been confirmed [3].

3. VARIATIONS OF PARTICLE SWARM OPTIMIZATION

3.1 Discrete PSO [8]

The original PSO described in II is basically developed for continuous optimization problems. However, lots of practical engineering problems are formulated as combinatorial optimization problems. Kennedy and Eberhart developed a discrete binary version of PSO for the problems [8]. 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:

\[ P(s_i^{t+1} = 1) = f(s_i^{t}, v_i^{t}, \text{pbest}_i, \text{gbest}) \]  \hspace{1cm} (4)

The parameter \( v \), an agent’s predisposition 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 the 0 choice. Such a threshold requires staying in the range [0, 1]. One of the functions accomplishing this feature is the sigmoid function, which usually utilized with neural networks.

\[ \text{sig}(v_i^{t}) = \frac{1}{1 + \exp(-v_i^{t})} \]  \hspace{1cm} (5)

The agent’s disposition 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. Namely, like the basic continuous version, the formula for binary version of PSO can be described as follows:

\[ v_{i}^{k+1} = v_{i}^{k} + \text{rand} \times (\text{pbest}_i - s_i^{k}) + \text{rand} \times (\text{gbest} - s_i^{k}) \]  \hspace{1cm} (6)

\[ \rho_{i}^{k+1} = \text{sig}(v_{i}^{k+1}) \text{then } s_i^{k+1} = 1; \text{else } s_i^{k+1} = 0 \]  \hspace{1cm} (7)

where, rand: a positive random number drawn from a uniform distribution with a predefined upper limit. \( \rho_{i}^{k+1} \): a vector of random numbers of [0.0, 1.0].

In the binary version, the limit of rand is often set so that the two rand limits sum to 4.0. These formulas are iterated repeatedly over each dimension of each agent. The second and third term of RHS of (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 \( V_{\text{max}} \) can be set at the start of a trial. In practice, \( V_{\text{max}} \) 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 the above decision equations.

3.2 PSO for MINLP[9]

Lots of engineering problems have to handle both discrete and continuous variables using nonlinear objective functions. Kennedy and Eberhart discussed about integration of binary and continuous version of PSO [5]. Fukuyama, et al., presented a PSO for MINLP by modifying the continuous version of PSO [9]. The method can be briefly described as follows.

Discrete variables can be handled in (1) and (3) with little modification. Discrete numbers instead of continuous
numbers can be used to express the current position and velocity. Namely, discrete random number is used for \( \text{rand} \) in (1) and the whole calculation of RHS of (1) is discritized to the existing discrete number. Using this modification for discrete numbers, both continuous and discrete number can be handled in the algorithm with no inconsistency. In [9], the PSO for MINLP was successfully applied to a reactive power and voltage control problem with promising results.

3.3 Hybrid PSO (HPSO) [10]

HPSO utilizes the basic mechanism of PSO and the natural selection mechanism, which is usually utilized by EC methods such as GAs. Since search procedure by PSO deeply depends on pbest and gbest, the searching area may be limited by pbest and gbest. On the contrary, by introduction of the natural selection mechanism, effect of pbest and gbest is gradually vanished by the selection and broader area search can be realized. Agent positions with low evaluation values are replaced by those with high evaluation values using the selection. The exchange rate at the selection is added as a new optimization parameter of PSO. On the contrary, pbest information of each agent is maintained. Therefore, both intensive search in a current effective area and dependence on the past high evaluation position are realized at the same time. Fig. 5 shows a general flow chart of HPSO. Fig. 6 shows concept of step. 2, 3, and 4 of the general flow chart.

3.4 Lbest model

Eberhart and Kennedy called the above-mentioned basic method as "gbest model". They also developed "lbest model" [5]. In the model, agents have information only of their own and their nearest array neighbor’ bests (lbests), rather than that of the entire group. Namely, in (1). gbest is replaced by lbests in the model.

4. PARAMETER SELECTIONS AND CONSTRICTION FACTOR APPROACH

4.1 Parameter Selection

PSO has several explicit parameters whose values can be adjusted to produce variations in the way the algorithm searches the solution space. The parameters in (1)(2) are as follows:

\[ c_j : \text{weighting factor}, \]
\[ w_{\text{max}} : \text{initial weight of the weight function}, \]
\[ w_{\text{min}} : \text{final weight of the weight function}, \]

Shi and Eberhart tried to examine the parameter selection of the above parameters [11][12]. According to their examination, the following parameters are appropriate and the values do not depend on problems:

\[ c_j=2.0, w_{\text{max}}=0.9, w_{\text{min}}=0.4. \]

The values are also appropriate for power system problems [9][13].

4.2 Constriction factor

The basic system equation of PSO (equ. (1), (2), and (3)) can be considered as a kind of difference equations. Therefore, the system dynamics, namely, search procedure, can be analyzed by the eigen value analysis. The constriction
factor approach utilizes the eigen value analysis and controls the system behavior so that the system behavior has the following features [14]:
(a) The system does not diverge in a real value region and finally can converge,
(b) The system can search different regions efficiently.
The velocity of the constriction factor approach (simplest constriction) can be expressed as follows instead of (1) and (2):

\[ v_i^{k+1} = K \left[ v_i^k + c_1 \times \text{rand()} \times \left( p_{\text{best}} - x_i^k \right) + c_2 \times \text{rand()} \times \left( g_{\text{best}} - x_i^k \right) \right] \quad (8) \]

\[ K = \frac{2}{2 - \phi - \sqrt{\phi^2 - 4\phi}} \quad \text{where } \phi = c_1 + c_2, \phi > 4 \quad (9) \]

For example, if \( \phi = 4.1 \), then \( \chi = 0.73 \). As \( \phi \) increases above 4.0, \( \chi \) gets smaller. For example, if \( \phi = 5.0 \), then \( \chi = 0.38 \), and the damping effect is even more pronounced. The constriction factor approach results in convergence of the agent over time. Unlike other EC methods, the constriction factor approach of PSO ensures the convergence of the search procedures based on the mathematical theory. Namely, the amplitude of the each agent's oscillation decreases as it focuses on a previous best point. The constriction factor approach can generate higher quality solutions than the conventional PSO approach [15].

However, the constriction factor only considers dynamic behavior of each agent and the effect of the interaction among agents; namely, the effect of pbest and gbest in the system dynamics is one of the future works [14].

5. RESEARCH AREAS AND APPLICATIONS

Ref. [16]-[66] shows other PSO related papers. Most of papers are related to the method itself, and its modification and comparison with other EC methods. PSO is a new EC technique and there are a few applications. Table 1 shows applications of PSO in general fields. The last four applications are in power system fields. Detailed description of [9][32][66] and [50] can be found in Chap.13. Application of PSO to various fields is at the early stage. More applications can be expected.

6. CONCLUSIONS

This chapter presents fundamentals of particle swarm optimization (PSO) techniques. While a lot of evolutionary computation techniques have been developed for combinatorial optimization problems, PSO has been basically developed for continuous optimization problem. PSO has several variations including integration with selection mechanism and hybridization for handling both discrete and continuous variables. Moreover, recently developed constriction factor approach is based on mathematical analysis and useful for obtaining high quality solutions. A few applications are already appeared using PSO. PSO can be an efficient optimization tool for nonlinear continuous optimization problems, combinatorial optimization problems, and mixed-integer nonlinear optimization problem (MINLP).

### REFERENCES


C. Mohan, and B. Al-kazemi, "Discrete particle swarm optimization", 50
Proceedings of the Workshop on Particle Swarm Optimization, Indianapolis, IN: Purdue School of Engineering and Technology, IUPUI, 2001.


