CHAPTER

PARTICLE SWARM OPTIMIZATION 8

Particle swarm optimization (PSO) was developed by Kennedy and Eberhart in 1995, based on the swarm behavior in nature, such as fish and bird schooling. Since then, PSO has generated much wider interests and forms an exciting, ever-expanding research subject, called swarm intelligence (SI). PSO has been applied to almost every area in optimization, computational intelligence, and design applications. There are at least two dozen PSO variants, and hybrid algorithms by combining PSO with other existing algorithms are also investigated extensively.

8.1 SWARM INTELLIGENCE

Many algorithms such as ant colony optimization and the firefly algorithm use the behavior of the so-called *swarm intelligence* (Kennedy et al., 2001; Engelbrecht, 2005; Yang et al., 2013). Particle swarm optimization, or PSO, was developed by Kennedy and Eberhart in 1995 (Kennedy and Eberhart, 1995) and has become one of the most widely used SI-based algorithms due to its simplicity and flexibility. Rather than use the mutation/crossover or pheromone, it uses the real-number randomness and the global communication among the swarm particles. Therefore, it is also easier to implement because there is no encoding or decoding of the parameters into binary strings as with those in genetic algorithms where real-number strings can also be used.

Many new algorithms that are based on swarm intelligence may have drawn inspiration from different sources, but they have some similarity to some of the components that are used in PSO. In this sense, PSO pioneered the basic ideas of SI-based computation.

8.2 PSO ALGORITHM

The PSO algorithm searches the space of an objective function by adjusting the trajectories of individual agents, called *particles*, as the piecewise paths formed by positional vectors in a quasi-stochastic manner (Kennedy and Eberhart, 1995; Kennedy et al., 2001; Engelbrecht, 2005). The movement of a swarming particle consists of two major components: a stochastic component and a deterministic component. Each particle is attracted toward the position of the current global best $\boldsymbol{g}^*$ and its own best location $\boldsymbol{x}_i^*$ in history, while at the same time it has a tendency to move randomly.

When a particle finds a location that is better than any previously found locations, PSO updates that location as the new current best for particle i. There is a current best for all n particles at any time t during iterations. The aim is to find the global best among all the current individual best solutions until the objective no longer improves or after a certain number of iterations. The movement of particles is

Nature-Inspired Optimization Algorithms. https://doi.org/10.1016/B978-0-12-821986-7.00015-9

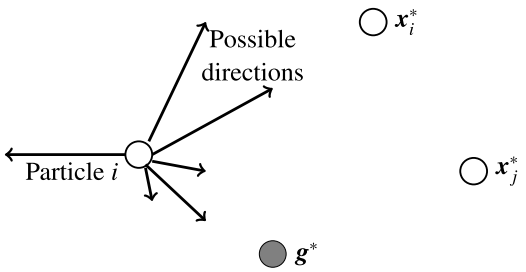

FIGURE 8.1

Schematic representation of the motion of a particle in PSO, moving towards the global best g^* and the current individual best x_i^* for each particle i.

schematically represented in Fig. 8.1, where x_i^* is the current best for particle i, and $g^* \approx \min\{f(x_i^*)\}$ for $(i = 1, 2, ..., n)$ is the current global best at t.

The essential steps of the particle swarm optimization can be summarized as the pseudo code shown in Algorithm 8.1.

Algorithm 8.1: Particle swarm optimization.

Data: Objective functions $f(x)$
Result: Best or optimal solution
Initialize locations x_i and velocity v_i of n particles;
Find g^* from $\min\{f(x_1), ..., f(x_n)\}$ (at $t = 0$);
while *(criterion)* **do**
    for *all n particles and all d dimensions* **do**
        Generate new velocity v_i^{t+1} using equation (8.1);
        Calculate new locations $x_i^{t+1} = x_i^t + v_i^{t+1}$;
        Evaluate the objective function at new locations x_i^{t+1};
        Find the current best for each particle x_i^*;
    end
    Find the current global best g^*;
    Update $t = t + 1$ (pseudo time or iteration counter);
end
Output the final results x_i^* and g^*;

Let x_i and v_i be the position vector and velocity for particle i, respectively. The new velocity vector is determined by the following formula:

$$v_i^{t+1} = v_i^t + \alpha\epsilon_1[g^* - x_i^t] + \beta\epsilon_2[x_i^{*(t)} - x_i^t], \tag{8.1}$$

where $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$ are two random vectors, and each entry takes the values between 0 and 1. The parameters α and β are the learning parameters or acceleration constants, which can typically be taken as, say, $\alpha \approx \beta \approx 2$.

The initial locations of all particles should distribute relatively uniformly so that they can sample over most regions, which is especially important for multimodal problems. The initial velocity of a particle can be taken as zero, that is, $\boldsymbol{v}_i^{t=0} = 0$. The new position can then be updated by

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1}\Delta t, \tag{8.2}$$

where Δt is the time increment. Since PSO is iterative with a discrete integer time counter, we can set $\Delta t = 1$ for all implementations. Although $\boldsymbol{v}_i$ can be any values, it is usually bounded in some range $[0, \boldsymbol{v}_{\max}]$.

There are many variants that extend the standard PSO algorithm, and the most noticeable improvement is probably to use inertia function $\theta(t)$ so that $\boldsymbol{v}_i^t$ is replaced by $\theta(t)\boldsymbol{v}_i^t$

$$\boldsymbol{v}_i^{t+1} = \theta \boldsymbol{v}_i^t + \alpha\boldsymbol{\epsilon}_1[\boldsymbol{g}^* - \boldsymbol{x}_i^t] + \beta\boldsymbol{\epsilon}_2[\boldsymbol{x}_i^{*(k)} - \boldsymbol{x}_i^t], \tag{8.3}$$

where θ takes the values between 0 and 1 in theory (Kennedy et al., 2001; Chatterjee and Siarry, 2006). In the simplest case, the inertia function can be taken as a constant, typically $\theta \approx 0.5 \sim 0.9$. This is equivalent to introducing a virtual mass to stabilize the motion of the particles, and thus the algorithm is expected to converge more quickly.

8.3 ACCELERATED PSO

The standard particle swarm optimization uses both the current global best $\boldsymbol{g}^*$ and the individual best $\boldsymbol{x}_i^*$ at iteration t. One of the reasons of using the individual best is probably to increase the diversity in the quality solutions; however, this diversity can be simulated using some randomness. Subsequently, there is no compelling reason for using the individual best unless the optimization problem of interest is highly nonlinear and multimodal.

A simplified version that could accelerate the convergence of the algorithm is to use the global best only. The so-called *accelerated particle swarm optimization* (APSO) was developed by Xin-She Yang in 2008 and then has been developed further in subsequent studies (Yang et al., 2011; Gandomi et al., 2013). Thus, in APSO, the velocity vector is generated by a simpler formula

$$\boldsymbol{v}_i^{t+1} = \boldsymbol{v}_i^t + \beta(\boldsymbol{g}^* - \boldsymbol{x}_i^t) + \alpha(\boldsymbol{\epsilon} - \frac{1}{2}), \tag{8.4}$$

where $\boldsymbol{\epsilon}$ is a random variable with values from 0 to 1. Here the shift 1/2 is purely out of convenience. We can also use a standard normal distribution $\alpha\boldsymbol{\epsilon}_t$ where $\boldsymbol{\epsilon}_t$ is drawn from $N(0, 1)$ to replace the third term. Now we have

$$\boldsymbol{v}_i^{t+1} = \boldsymbol{v}_i^t + \beta(\boldsymbol{g}^* - \boldsymbol{x}_i^t) + \alpha\boldsymbol{\epsilon}_t, \tag{8.5}$$

where $\boldsymbol{\epsilon}_t$ can be drawn from a Gaussian distribution or any other suitable distributions.

The update of the position is simply

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1}. \tag{8.6}$$

In order to simplify the formulation even further, we can also write the update of the location in a single step

$$\boldsymbol{x}_i^{t+1} = (1-\beta)\boldsymbol{x}_i^t + \beta \boldsymbol{g}^* + \alpha \boldsymbol{\epsilon}_t. \tag{8.7}$$

The typical values for this accelerated PSO are $\alpha \approx 0.1 \sim 1$ and $\beta \approx 0.1 \sim 0.7$, though $\alpha \approx 1$ and $\beta \approx 0.5$ can be taken as the initial values for most unimodal objective functions. It is worth pointing out that the parameters α and β should in general be related to the scales of the independent variables $\boldsymbol{x}_i$ and the search domain. Surprisingly, this simplified APSO can have global convergence.

A further improvement to APSO is to reduce the randomness as iterations proceed. This means that we can use a monotonically decreasing function such as

$$\alpha = \alpha_0 e^{-\gamma t}, \tag{8.8}$$

or

$$\alpha = \alpha_0 \gamma^t, \quad (0 < \gamma < 1), \tag{8.9}$$

where $\alpha_0 \approx 0.5 \sim 1$ is the initial value of the randomness parameter. Here t is the number of iterations or time steps. $0 < \gamma < 1$ is a control parameter. For example, in our implementation, we can use $\gamma = 0.9$ to 0.97. Obviously, other non-increasing function forms $\alpha(t)$ can also be used as we can see in our demo implementation later. In addition, these parameters should be fine-tuned to suit your optimization problems of interest.

8.4 IMPLEMENTATION

There are many different implementations for the standard PSO. Thus, we will not implement it here. Instead, we now focus on the implementation of the simplified APSO using both Matlab® and Octave, and a simple program is provided below. This program can find the global optimal solution of most nonlinear functions in less a minute on a desktop computer.

Now let us look at the 2D Michalewicz function

$$f(x, y) = -\left\{ \sin(x)\Big[\sin(\frac{x^2}{\pi})\Big]^{2m} + \sin(y)\Big[\sin(\frac{2y^2}{\pi})\Big]^{2m} \right\},$$

where $m = 10$. The stationary conditions $f_x = f_y = 0$ require that

$$-\frac{4m}{\pi} x \sin(x)\cos(\frac{x^2}{\pi}) - \cos(x)\sin(\frac{x^2}{\pi}) = 0,$$

and

$$-\frac{8m}{\pi} y \sin(x)\cos(\frac{2y^2}{\pi}) - \cos(y)\sin(\frac{2y^2}{\pi}) = 0.$$

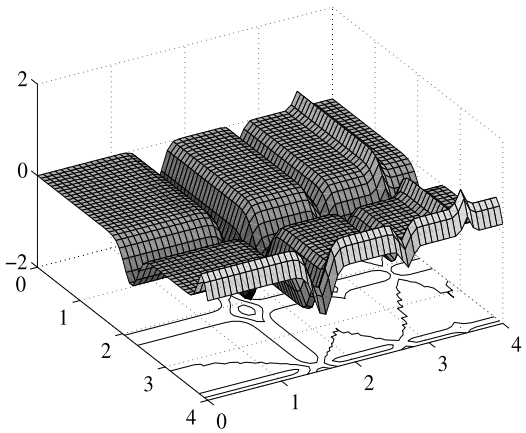

FIGURE 8.2

Michalewicz's function with the global optimality at (2.20319, 1.57049).

The solution at $(0, 0)$ is trivial, and the minimum $f^* \approx -1.801$ occurs at about (2.20319, 1.57049) (see Fig. 8.2).

```
% ---------------------------------------------------------%
% The Accelerated Particle Swarm Optimization (APSO)
% (written by X S Yang, Cambridge University 2008)
function [best]=pso_simpledemo
% n=number of particles
% Num_iterations=total number of iterations
n=20; Num_iterations=50;
% Michalewicz Function f*=-1.801 at [2.20319,1.57049]
% The objective function as a string
funstr='-sin(x)*(sin(x^2/3.14159))^20-sin(y)*(sin(2*y^2/3.14159))^20';
% Converting to an inline function and vectorization
f=vectorize(inline(funstr));
% range=[xmin xmax ymin ymax];
range=[0 4 0 4];
% -----------------------------------------------------
% Setting the parameters: alpha, beta, gamma
% The accelerated PSO with alpha=alpha0*gamma^t
alpha0=1; beta=0.5; gamma=0.9;
% -----------------------------------------------------
% Grid values of the objective function
% These values are used for visualization only
Ngrid=100;
dx=(range(2)-range(1))/Ngrid; dy=(range(4)-range(3))/Ngrid;
xgrid=range(1):dx:range(2); ygrid=range(3):dy:range(4);
[x,y]=meshgrid(xgrid,ygrid);
```

```
z=f(x,y);
% Display the shape of the function to be optimized
figure(1);
surfc(x,y,z);
% ------------------------------------------------
best=zeros(Num_iterations,3);   % initialize history
% ----- Start Particle Swarm Optimization -----------
% generating the initial locations of n particles
[xn,yn]=init_pso(n,range);
% Display the paths of particles in a figure
% with a contour of the objective function
 figure(2);
% Start iterations
for i=1:Num_iterations,
% Show the contour of the function
  contour(x,y,z,15); hold on;
% Find the current best location (xo,yo)
zn=f(xn,yn); zn_min=min(zn);
xo=min(xn(zn==zn_min)); yo=min(yn(zn==zn_min));
zo=min(zn(zn==zn_min));
% Trace the paths of all roaming particles
% Display these roaming particles
plot(xn,yn,'.',xo,yo,'*'); axis(range);
% The accelerated PSO with alpha=alpha0*gamma^t
 alpha=alpha0*gamma.^i;
% Move all the particles to new locations
[xn,yn]=pso_move(xn,yn,xo,yo,alpha,beta,range);
drawnow;
% Use "hold on" to display paths of particles
hold off;
% History
best(i,1)=xo; best(i,2)=yo; best(i,3)=zo;
end   %%%%% end of iterations
% ----- All subfunctions are listed here -----
% Intial locations of n particles
function [xn,yn]=init_pso(n,range)
xrange=range(2)-range(1); yrange=range(4)-range(3);
xn=rand(1,n)*xrange+range(1); yn=rand(1,n)*yrange+range(3);
% Move all the particles toward (xo,yo)
function [xn,yn]=pso_move(xn,yn,xo,yo,a,b,range)
nn=size(yn,2);  %a=alpha, b=beta
xn=xn.*(1-b)+xo.*b+a.*(rand(1,nn)-0.5);
yn=yn.*(1-b)+yo.*b+a.*(rand(1,nn)-0.5);
[xn,yn]=findrange(xn,yn,range);
% Make sure the particles are within the range
function [xn,yn]=findrange(xn,yn,range)
nn=length(yn);
for i=1:nn,
```

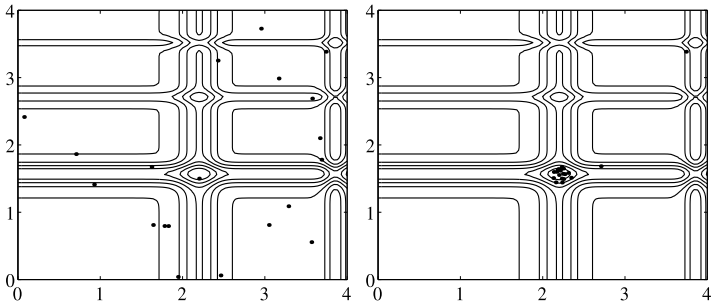

FIGURE 8.3

Initial locations of 20 particles and their new locations after 10 iterations.

```
    if xn(i)<=range(1), xn(i)=range(1); end
    if xn(i)>=range(2), xn(i)=range(2); end
    if yn(i)<=range(3), yn(i)=range(3); end
    if yn(i)>=range(4), yn(i)=range(4); end
end
% ------------------------------------------------------------%
```

In this implementation, we have used $n = 20$, $\alpha_0 = 1$, $\beta = 0.5$, and $\gamma = 0.9$. If we run the program, we can get the global optimum after about 100 iterations. The initial distribution of the population and the snapshot at $t = 10$ are shown in Fig. 8.3. Obviously, this is a demo implementation, and a vectorized implementation for any higher dimensions can produce much better results, as we have done in various applications (Yang et al., 2011). A vectorized version is provided in Appendix B of this book.

8.5 CONVERGENCE ANALYSIS

From the statistical point of view, each particle in PSO forms a Markov chain, though this Markov chain is biased towards to the current best, since the transition probability often leads to the acceptance of the move towards the current global best. In addition, the multiple Markov chains are interacting in terms of partially deterministic attraction movement. Therefore, any mathematical analysis concerning of the rate of convergence of PSO may be difficult. However, there are some good results using both the dynamical system and Markov chain theories.

8.5.1 DYNAMICAL SYSTEM

The first convergence analysis in terms of dynamical system theories was carried out by Clerc and Kennedy (2002). Mathematically, if we ignore the random factors, we can view the system formed by (8.1) and (8.2) as a dynamical system. If we focus on a single particle i and imagine there is only one particle in this system, then the global best $\boldsymbol{g}^*$ is the same as its current best $\boldsymbol{x}_i^*$. In this case, we have

$$\boldsymbol{v}_i^{t+1} = \boldsymbol{v}_i^t + \gamma(\boldsymbol{g}^* - \boldsymbol{x}_i^t), \quad \gamma = \alpha + \beta, \tag{8.10}$$

and

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1}. \tag{8.11}$$

Following the analysis of a 1D dynamical system for particle swarm optimization by Clerc and Kennedy (2002), we can replace $\boldsymbol{g}^*$ by a parameter constant p so that we can see whether or not the particle of interest will converge towards p. Now we can write the above system as a simple dynamical system

$$v(t+1) = v(t) + \gamma(p - x(t)), \quad x(t+1) = x(t) + v(t+1). \tag{8.12}$$

For simplicity, we only focus on a single particle. By setting $u_t = p - x(t+1)$ and using the notations for dynamical systems, we have

$$v_{t+1} = v_t + \gamma u_t, \tag{8.13}$$

$$u_{t+1} = -v_t + (1-\gamma)u_t, \tag{8.14}$$

or

$$Y_{t+1} = AY_t, \tag{8.15}$$

where

$$A = \begin{pmatrix} 1 & \gamma \\ -1 & 1-\gamma \end{pmatrix}, \quad Y_t = \begin{pmatrix} v_t \\ u_t \end{pmatrix}. \tag{8.16}$$

The general solution of this dynamical system can be written as

$$Y_t = Y_0 \exp[At], \tag{8.17}$$

where Y_0 is the initial condition and $\exp[At]$ is the matrix exponential. The main behavior of this system can be characterized by the eigenvalues λ of A

$$\lambda_{1,2} = 1 - \frac{\gamma}{2} \pm \frac{\sqrt{\gamma^2 - 4\gamma}}{2}. \tag{8.18}$$

It can be seen clearly that $\gamma = 4$ leads to a bifurcation when $\gamma^2 - 4\gamma = 0$.

Following a straightforward analysis of this dynamical system, we can have three cases. For $0 < \gamma < 4$, cyclic and/or quasi-cyclic trajectories exist because the two eigenvalues are real. In this case, when randomness is gradually reduced, some convergence can be observed. For $\gamma > 4$, the two eigenvalues become complex, which means that non-cyclic behavior can be expected, and the distance from Y_t to the center $(0, 0)$ is monotonically increasing with t. In a special case $\gamma = 4$, both eigenvalues become $1 - \lambda/2 = -1$, and some convergence behavior can be observed. For detailed analysis, refer to Clerc and Kennedy (2002). Since p is linked with the global best, as the iterations continue it can be expected that all particles will aggregate towards the global best. It is worth pointing out that this analysis is a simplified version based on the additional assumptions. Therefore, the standard PSO and its variants may behave very differently in practice.

In addition, the global best in PSO is the best solution found by the algorithm during iterations, which may be different from the true global optimality of the problem of interest. This point will also become clearer in the framework of Markov chain theory.

8.5.2 MARKOV CHAIN APPROACH

Various studies on the convergence properties of the PSO algorithm have diverse results (Clerc and Kennedy, 2002; Pan et al., 2013; Sudholt and Witt, 2010; Trelea, 2003; van den Bergh and Engelbrecht, 2006). However, care should be taken in interpreting these results in practice, especially for the discussion of the implications from a practical perspective.

Many studies can prove the convergence of the PSO under appropriate conditions, but the converged states are often linked to the current global best solution $\boldsymbol{g}^*$ found so far. There is no guarantee that this $\boldsymbol{g}^*$ is the true global solution $\boldsymbol{g}_{\text{true}}$ to the problem. In fact, many simulations and empirical observations by many researchers suggest that in many cases, this $\boldsymbol{g}^*$ is often stuck in a local optimum, which is often referred to as the *premature convergence*. All these theories do not guarantee

$$\boldsymbol{g}^* \approx \boldsymbol{g}_{\text{true}}. \tag{8.19}$$

In fact, many statistically significant test cases suggest that

$$\boldsymbol{g}^* \neq \boldsymbol{g}_{\text{true}}. \tag{8.20}$$

According to a study by Pan et al. (2013), the standard PSO does not guarantee that the global optimum is reachable, and the global optimality is searchable with a certain probability $p(\zeta_m^{(t)})$, where k is the kth iteration and m is the swarm size. Here ζ is the state sequences of the particles.

For a given swarm size m and iteration k, a Markov chain $\{\xi_i^{(k)}, k \geq 1\}$ formed by the ith particle at time k can be defined, and then the swarm state sequence $\zeta^{(k)} = (\xi_i^{(k)}, ..., \xi_m^{(k)})$ also forms a Markov chain for all $k \geq 1$. For the standard PSO with an inertia parameter θ shown in Eq. (8.3), the probability that $\zeta^{(k+1)}$ in the optimal set Ω_* can be estimated by

$$p(\zeta^{(k+1)} \in \Omega_* \big| k, m)$$

$$= 1 - p(\zeta^{(1)}) \cdot \prod_{j=1}^{k} \prod_{i=1}^{m} \left(1 - \frac{R_k}{\theta ||\boldsymbol{x}_i^k - \boldsymbol{x}_i^{k-1}||} \cdot \frac{R_k}{c||\boldsymbol{x}_i^{*(k)} - \boldsymbol{x}_i^k||} \cdot \frac{R_k}{c||\boldsymbol{g}_* - \boldsymbol{x}_i^k||}\right), \tag{8.21}$$

where $R_k > 0$ is a local radius and c is defined by $\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2 \sim U(0, c)$. Pan et al. showed that

$$0 \leq p(\zeta^{(k+1)} \big| k, m) \leq 1, \tag{8.22}$$

which means that the standard PSO will not guarantee to get to the global optimality nor guarantee to miss it. The equality $p = 1$ only holds for $m \to \infty$ and $k \to \infty$. Since the size is finite and the iteration is also finite, this probability is typically $0 < p < 1$.

However, that the standard PSO does not guarantee convergence does not mean that other variants cannot have global convergence.

8.6 BINARY PSO

In the standard PSO, the positions and velocities take continuous values. However, many problems are combinatorial and their variables take only discrete values. In some cares, the variables can only be 0

and 1, and such binary problems require modifications of the PSO algorithm. Kennedy and Eberhart in 1997 presented a stochastic approach to discretize the standard PSO, and they interpreted the velocity in a stochastic sense (Kennedy and Eberhart, 1997).

First, the continuous velocity $\boldsymbol{v}_i = (v_{i1}, v_{i2}, ..., v_{ik}, ..., v_{id})$ is transformed using a sigmoid transformation

$$S(v_{ik}) = \frac{1}{1 + \exp(-v_{ik})}, \quad k = 1, 2, ..., d, \tag{8.23}$$

which applies to each component of the velocity vector $\boldsymbol{v}_i$ of particle i. Obviously, when $v_{ik} \to \infty$, we have $S(v_{ik}) \to 1$, while $S_{ik} \to 0$ when $v_{ik} \to -\infty$. However, because the variations at the two extremes are very slow, a stochastic approach is introduced.

Secondly, a uniformly-distributed random number $r \in (0, 1)$ is drawn, then the velocity is converted to a binary variable by the following stochastic rule:

$$x_{ik} = \begin{cases} 1 & \text{if } r < S(v_{ik}), \\ 0 & \text{otherwise.} \end{cases} \tag{8.24}$$

In this case, the value of each velocity component v_{ik} is interpreted as a probability for x_{ik} taking the value 1. Even for a fixed value of v_{ik}, the actual value of x_{ik} is not certain before a random number r is drawn. In this sense, the binary PSO (BPSO) differs significantly from the standard continuous PSO.

In fact, since each component of each variable takes only 0 and 1, this BPSO can work for both discrete and continuous problems if the latter is coded in the binary system. Because the probability of each bit/component taking one is $S(v_{ik})$ and the probability of taking zero is $1 - S(v_{ik})$, the joint probability p of a bit change can be computed by

$$p = S(v_{ik})[1 - S(v_{ik})]. \tag{8.25}$$

Based on the runtime analysis of the binary PSO by Sudholt and Witt (2010), there are some interesting results on the convergence of BPSO. If the objective function has a unique global optimum in a d-dimensional space and the BPSO has a population size n with $\alpha + \beta = O(1)$, then the expected number N of iterations/generations of BPSO is

$$N = O\Big(\frac{d}{\log d}\Big), \tag{8.26}$$

and the expected freezing time is $O(d)$ for single bits and $O(d \log d)$ for nd bits.

One of the advantages of this binary coding and discritization is to enable binary representations of even continuous problems. For example, Kennedy and Eberhart (1997) provided an illustrative example of solving the 2nd De Jong function and found that 110111101110110111101001 in a 24-bit string corresponds to the optimal solution 3905.929932 from this representation. However, a disadvantage is that the Hamming distance from other local optima is large; therefore, it is unlikely that the search will jump from one local optimum to another. This means that binary PSO can get stuck in a local optimum with premature convergence. New remedies are still under active research.

Various studies show that PSO algorithms can outperform genetic algorithms and other conventional algorithms for solving many optimization problems. This is partially due to the fact that the broadcasting ability of the current best estimates gives a better and quicker convergence towards the

optimality. However, PSO algorithms do have some disadvantages, such as premature convergence. Further developments and improvements are still under active research.

Furthermore, as we can see from other chapters in this book, other methods such as the cuckoo search (CS) and firefly algorithms (FA) can perform even better than PSO in many applications. In many cases, DE, PSO and SA can be considered special cases of CS and FA. Therefore, CS and FA have become an even more active area for further research. In fact, swarm intelligence is a very active area of research (Yang and Papa, 2016; Yang et al., 2018; Ser et al., 2019).

REFERENCES

Chatterjee, A., Siarry, P., 2006. Nonlinear inertia variation for dynamic adaptation in particle swarm optimization. Computers & Operations Research 33 (3), 859–871.

Clerc, M., Kennedy, J., 2002. The particle swarm: explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6 (1), 58–73.

Engelbrecht, A.P., 2005. Fundamentals of Computational Swarm Intelligence. Wiley, Hoboken, NJ, USA.

Gandomi, A., Yun, G., Yang, X., Talatahari, S., 2013. Chaos-enhanced accelerated particle swarm optimization. Communications in Nonlinear Science and Numerical Simulation 18 (2), 327–340.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks. IEEE, Piscataway, NJ, USA, pp. 1942–1948.

Kennedy, J., Eberhart, R., 1997. A discrete binary version of the particle swarm algorithm. In: International Conference on Systems, Man, and Cybernetics. Orlando FL, 12–15 Oct 1997, vol. 5. IEEE Publications, Piscataway, NJ, pp. 4104–4109.

Kennedy, J., Eberhart, R.C., Shi, Y., 2001. Swarm Intelligence. Academic Press, London, UK.

Pan, F., Li, X., Zhou, Q., Li, W., Gao, Q., 2013. Analysis of standard particle swarm optimization algorithm based on Markov chain. Acta Automatica Sinica 39 (4), 381–389.

Ser, J.D., Osaba, E., Molina, D., Yang, X.S., Salcedo-Sanz, S., Camacho, D., Das, S., Suganthan, P.N., Coello Coello, C.A., Herrera, F., 2019. Bio-inspired computation: where we stand and what's next. Swarm and Evolutionary Computation 48, 220–250.

Sudholt, D., Witt, C., 2010. Runtime analysis of a binary particle swarm optimizer. Theoretical Computer Science 411 (21), 2084–2100.

Trelea, I., 2003. The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters 85 (6), 317–325.

van den Bergh, F., Engelbrecht, A., 2006. A study of particle swarm optimization particle trajectories. Information Sciences 176 (8), 937–971.

Yang, X.S., Papa, J.P., 2016. Bio-inspired Computation and Applications in Image Processing. Academic Press, Elsevier, London.

Yang, X.S., Deb, S., Fong, S., 2011. Accelerated particle swarm optimization and support vector machine for business optimization and applications. In: Networked Digital Technologies. In: Communications in Computer and Information Science, vol. 136. Springer, Berlin, pp. 53–66.

Yang, X.S., Cui, Z.H., Xiao, R.B., Gandomi, A.H., Karamanoglu, M., 2013. Swarm Intelligence and Bio-Inspired Computation: Theory and Applications. Elsevier, London, UK.

Yang, X.S., Deb, S., Zhao, Y.X., Fong, S., He, X., 2018. Swarm intelligence: past, present and future. Soft Computing 22 (18), 5923–5933.