Chapter 3

Particle Swarm Optimisation

Particle Swarm Optimisation is discussed as an algorithm for optimising unconstrained problems. The chapter looks into standard topologies used in the algorithm, and touches on a number of improvements to Particle Swarm Optimisation.

3.1 Introduction to unconstrained optimisation

Numerical optimisation techniques have their application in many fields, including natural science, engineering, finance, medicine and telecommunications. The objective of such techniques is to assign values from a given domain to a set of parameters such that a specific function is optimised. The function that is minimised or maximised (optimised) is called the objective function, and it depends on a set of solution-defining variables. Let $\mathbf{x} = (x_1, x_2, \ldots x_n)^T \in \mathbb{R}^n$ represent the domain of the objective function, or the optimisation (solution-defining) variable. Let f, the function that needs to be optimised, assign values from $\mathbb{R}^n$ to $\mathbb{R}$ such that $f : \mathbb{R}^n \to \mathbb{R}$.

For minimisation problems, the ideal is to find a global minimum $\mathbf{x}^\star$ such that

$$f(\mathbf{x}^\star) \leq f(\mathbf{x}), \quad \forall \mathbf{x} \in \mathbb{R}^n \tag{3.1}$$

For some applications, a local minimum $\mathbf{x}_L^\star$ on a domain $L \subset \mathbb{R}^n$ is an acceptable solution. In such cases

$$f(\mathbf{x}_L^\star) \leq f(\mathbf{x}), \quad \forall \mathbf{x} \in L \tag{3.2}$$

In both cases, finding a global minimum or a local minimum, the search space can be unconstrained or constrained by a set of constraints. This chapter focuses on Particle Swarm Optimisation (PSO) for unconstrained optimisation, the constrained case is examined in detail in Chapter 4.

Traditionally, numerical optimisation techniques have mainly been developed from the operations research community [19, 38]. The past decade has witnessed an increase in contributions from the artificial intelligence community, most notably from the evolutionary computing field [3]. Recently, PSO has been introduced as a successful technique for numerical optimisation [17, 26, 28]. Other recent methods for optimisation include artificial immune systems, differential evolution, memetic algorithms and scatter search [13].

3.2 Introduction to Particle Swarm Optimisation

Many efficient optimisation algorithms can be constructed from the study of ants working as a colony, birds migrating in a flock toward some destination, or fish swimming in a school. While the individual behaviour of an organism may seem inefficient, the collective effort of individuals inside a swarm can become complex and intelligent [5].

One such a method is Particle Swarm Optimisation (PSO), originally introduced by Kennedy and Eberhart [26]. PSO represents an optimisation method where individuals, called particles, collaborate as a population, or swarm, to reach a collective goal, for example minimising an n-dimensional function f.

Each particle is n-dimensional, and is a potential minimum of f. A particle has memory of the best solution that it has found, called its *personal best*. The particles fly through the search space with a velocity, which is dynamically adjusted according to its personal best and the best solution found by a neighbourhood of particles.

There is thus a sharing of information that takes place. Particles profit from the discoveries and previous experience of other particles during the exploration and search for lower objective function values.

There exist a great number of schemes in which this information sharing can take place. One of two sociometric principles is usually implemented [28], with more recent tolopogies investigated in [27, 29]. The first, called *gbest* (global best), conceptually connects all the particles in the population to one another. Thus each particle is influenced by the very best performance of the entire population. The second, called *lbest* (local best), creates a neighbourhood for each individual comprising itself and its k nearest neighbours in the population. Neighbourhoods are usually determined using particle indices, although topological neighbourhoods have also been used [54].

PSO differs from traditional optimisation methods, in that a population of potential solutions are used in the search. The direct fitness information instead of function derivatives or other related knowledge is used to guide the search. This search is based on probabilistic, rather than deterministic, transition rules.

Let i indicate a particle's index in the swarm. Then

$$S = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_s\}$$

is a swarm of s particles. In PSO each of the s particles has a current position

$$\mathbf{p}_i = (p_{i1}, p_{i2}, \ldots, p_{in})^T$$

and fly through the n-dimensional search space $\mathbb{R}^n$ with a current velocity

$$\mathbf{v}_i = (v_{i1}, v_{i2}, \ldots, v_{in})^T,$$

which is dynamically adjusted according to its own previous best solution

$$\mathbf{z}_i = (z_{i1}, z_{i2}, \ldots, z_{in})^T$$

and the current best solution $\widehat{\mathbf{z}}$ of the entire swarm (*gbest*), or the particle's neighbourhood (*lbest*).

At iteration time t of the PSO algorithm, the velocity and particle updates are specified seperately for each dimension j of the velocity and particle vectors. A particle $\mathbf{p}_i$ will interact and move according to the following equations [26]:

$$v_{ij}^{(t+1)} = v_{ij}^{(t)} + c_1 r_1^{(t)}[z_{ij}^{(t)} - p_{ij}^{(t)}] + c_2 r_2^{(t)}[\widehat{z}_j^{(t)} - p_{ij}^{(t)}] \quad (3.3)$$

$$p_{ij}^{(t+1)} = v_{ij}^{(t+1)} + p_{ij}^{(t)} \quad (3.4)$$

Equation (3.3) takes three terms into consideration to calculate the velocity of particle i: the particle's previous velocity, the distance between the particle and its personal best, and the distance between the particle and the best solution found by its neighbourhood, which may be the entire swarm.

The stochastic nature of the algorithm is determined by $r_1^{(t)}, r_2^{(t)} \sim UNIF(0,1)$, two uniform random numbers between zero and one. In the second and third terms these numbers are scaled by acceleration coefficients c_1 and c_2, where $0 \leq c_1, c_2 \leq 2$. Coefficient c_1 has been called the *cognitive learning rate* [2], since it scales the second term in (3.3), the term that defines the particle's movement in the direction of its personal best. In the same way, c_2 is called the *social learning rate*, scaling the influence of the neighbourhood's best solution on the particle.

After determining particle i's velocity, it moves toward its new position, as shown in (3.4).

At iteration time t of the PSO algorithm, the personal best of each particle is compared to its current performance. The personal best $\mathbf{z}_i^{(t)}$ is set to the better performance, i.e.

$$\mathbf{z}_i^{(t)} = \begin{cases} \mathbf{z}_i^{(t-1)} & \text{if } f(\mathbf{p}_i^{(t)}) \geq f(\mathbf{z}_i^{(t-1)}) \\ \mathbf{p}_i^{(t)} & \text{if } f(\mathbf{p}_i^{(t)}) < f(\mathbf{z}_i^{(t-1)}) \end{cases} \quad (3.5)$$

The definition of a particle's neighbourhood determines the vector $\widehat{\mathbf{z}}$, the best solution found by either the entire swarm or the particle's neighbourhood. Information sharing takes place through the neighbourhood - the most common, *gbest* and *lbest*, are discussed below. More recent tolopogies are investigated in [27, 29].

3.2.1 Global best (*gbest*)

The global best (*gbest*) PSO conceptually connects all the particles in the population to one another, so that each particle is influenced by the very best performance of the entire population. The global best particle pulls all particles towards itself, and particles move in its direction. If the global best is not updated regularly, the entire swarm may converge to it, resulting in premature convergence.

The global best $\widehat{\mathbf{z}}^{(t)}$ is set to the position of the particle with the best performance within the swarm, i.e.

$$\begin{aligned}\widehat{\mathbf{z}}^{(t)} \in \{\mathbf{z}_1^{(t)}, \mathbf{z}_2^{(t)}, \ldots, \mathbf{z}_s^{(t)}\} \mid f(\widehat{\mathbf{z}}^{(t)}) \\ = \min\{f(\mathbf{z}_1^{(t)}), f(\mathbf{z}_2^{(t)}), \ldots, f(\mathbf{z}_s^{(t)})\}\end{aligned} \tag{3.6}$$

3.2.2 Local best (*lbest*)

The *lbest* (local best) version of the PSO creates a neighbourhood for each individual comprising itself and its k nearest neighbours in the population. Neighbourhoods are usually determined using particle indices, although topological neighbourhoods have also been used [27]. Assuming that particle indices wrap around at s, let N_i be the neighbourhood of particle i.

$$N_i = \{\mathbf{z}_{i-k}^{(t)}, \mathbf{z}_{i-k+1}^{(t)}, \ldots, \mathbf{z}_i^{(t)}, \ldots, \mathbf{z}_{i+k-1}^{(t)}, \mathbf{z}_{i+k}^{(t)}\} \tag{3.7}$$

The neighbourhood best $\widehat{\mathbf{z}}_{N_i}^{(t)}$ at time t is defined as the best solution in particle i's neighbourhood:

$$\widehat{\mathbf{z}}_{N_i}^{(t)} \in N_i \mid f(\widehat{\mathbf{z}}_{N_i}^{(t)}) = \min\{f(\mathbf{z}_j^{(t)})\} \qquad \forall\, \mathbf{z}_j \in N_i \tag{3.8}$$

It is possible to let the neighbourhood size k be equal to zero, in which case each particle $\mathbf{p}_i$ only compares its current position with its own best position $\mathbf{z}_i^{(t)}$, and no information sharing takes place. A neighbourhood size of k equal to the swarm size s is equivalent to the *gbest* version of the PSO.

It was shown by [17, 49] that, although *lbest* is slower in convergence than *gbest*, *lbest* results in better solutions and searches a larger part of the search space.

3.2.3 The PSO algorithm

All that is left to complete from the above sections is the PSO algorithm itself. The definition of a particle's personal and global or local best position was defined. Using these best positions to determine each particle's velocity, the swarm of particles can successfully traverse the search space, looking for an optimum solution to a problem. The standard PSO algorithm, used to minimise a function

$$f : \mathbb{R}^n \to \mathbb{R} \tag{3.9}$$

is presented below:

Algorithm 3.1 - Particle Swarm Optimisation

1. Set the iteration number t to zero, and initialise the swarm S of n-dimensional particles $\mathbf{p}_i^{(0)}$: each component $p_{ij}^{(0)}$ of $\mathbf{p}_i^{(0)}$ is randomly initialised to a value in the initial domain of the swarm, an interval $[p_{min}, p_{max}]$. Since the particles are already randomly distributed, the velocities of particles are initialised to the zero vector $\mathbf{0}$.
2. Evaluate the performance $f(\mathbf{p}_i^{(t)})$ of each particle.
3. Compare the personal best of each particle to its current performance, and set $\mathbf{z}_i^{(t)}$ to the better performance, as shown in (3.5).
4. Use (3.6) to set the global best $\widehat{\mathbf{z}}^{(t)}$ to the position of the particle with the best performance within the entire swarm (*gbest*). When a *lbest* PSO is implemented, equation (3.8) is used to set the neighbourhood best $\widehat{\mathbf{z}}_{N_i}^{(t)}$ for each particle i.
5. Change the velocity vector for each particle according to equation (3.3).
6. Move each particle to its new position, according to equation (3.4).
7. Let $t := t + 1$.
8. Go to step 2, and repeat until convergence or $t = t_{max}$.

The algorithm has converged if the difference between the best solution found over a specified number of iterations remains within a certain bound. The algorithm iterates until either one of two conditions is met: the algorithm has converged, or the maximum number of iterations t_{max} have been reached.

3.2.4 Improvements

A number of methods have been proposed to improve the convergence and probability of convergence of the standard PSO algorithm, and are discussed in this section. Apart from changes to the PSO update equation (3.3), most of these methods make no changes to the PSO algorithm itself.

Maximum velocity

The probability of particles leaving the current search space can be reduced by clamping the velocity updates – the velocity update vectors in the first term of (3.3) can be restricted by specifying upper and lower bounds v_{max} and $-v_{max}$ on $v_{ij}^{(t)}$. If $v_{ij}^{(t)}$ is greater than v_{max}, then $v_{ij}^{(t)}$ is set to v_{max}. Similary, if $v_{ij}^{(t)}$ is smaller than $-v_{max}$, then $v_{ij}^{(t)}$ is set to the value of $-v_{max}$. The value of v_{max} is usually a function of the range of the problem. If the range of each component p_{ij} of particle $\mathbf{p}_i$ is between -10 and 10, v_{max} will be proportional to 10.

Inertia weight

The previous velocity in the first term of (3.3) can be scaled with an intertia weight w, i.e.

$$v_{ij}^{(t+1)} = wv_{ij}^{(t)} + c_1 r_1^{(t)}[z_{ij}^{(t)} - p_{ij}^{(t)}] + c_2 r_2^{(t)}[\hat{z}_j^{(t)} - p_{ij}^{(t)}] \tag{3.10}$$

The inertia weight was introduced to improve the rate of convergence of the PSO algorithm [48], and determines how much the velocity at time t should influence the velocity at time $t+1$. A large inertia weight causes the PSO to explore larger parts of the search space, while a smaller inertia weight results in exploitation of a smaller and more focussed region of the search space. An inertia weight of one results in an update equation equivalent to (3.3).

It is possible, through careful selection of the inertia weight, to create a balance between local and global exploration abilities, and therefore create a faster rate of convergence. The balance can be achieved with a linearly decreasing inertia weight

$$w = w_{max} - \frac{t}{t_{max}}(w_{max} - w_{min}) \tag{3.11}$$

where w_{max} is the initial (starting) inertia weight, and w_{min} is the final weight. The values t_{max} and t respectively indicate the maximum and current iteration number. Setting w_{max} to 0.9 and w_{min} to 0.4 has been shown to give good conversion, independent of the problems tested [47, 48].

Constriction coefficient

Maurice Clerc has introduced a *constriction factor* to PSO, which improves PSO's ability to control velocities [12]. The constriction factor analytically chooses values for w, c_1 and c_2 such that control is allowed over the dynamical characteristics of the particle swarm, including its exploration versus exploitation abilities. Clamping the velocities is not necessary when a constriction coefficient χ is used in (3.3), changing the velocity update to

$$v_{ij}^{(t+1)} = \chi\left(v_{ij}^{(t)} + c_1 r_1^{(t)}[z_{ij}^{(t)} - p_{ij}^{(t)}] + c_2 r_2^{(t)}[\widehat{z}_j^{(t)} - p_{ij}^{(t)}]\right) \tag{3.12}$$

with

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \tag{3.13}$$

and $\varphi = c_1 + c_2$, $\varphi > 4$.

As the value of φ tends to 4 (from above), the value of χ tends to 1 (from below), and the particle's velocity is almost not damped at all; as φ grows larger, χ tends to zero, and the particle's velocity is more strongly damped. A correct choice of the constriction factor makes velocity clamping unnecessary, although it was found that χ, combined with constraints on v_{max}, significantly improved PSO performance [18].

Guaranteed Convergence Particle Swarm Optimiser

The PSO described in this chapter, including the versions with an inertia weight (3.10) and constriction factor (3.12), all have a probability of converging prematurely. This can be clearly seen by considering the case when a particle's position and personal best coincide with the global best. The velocity of the particle will only depend on $v_{ij}^{(t)}$ (or $wv_{ij}^{(t)}$ or $\chi v_{ij}^{(t)}$), and if it is close to zero, or the position of the global best does not change, the particle will 'catch up' with the global best. This does not mean that the swarm has converged to a minimum, but merely that it has converged prematurely to the global best.

Van den Berg has introduced the Guaranteed Convergence PSO (GCPSO) [55, 56], which defines a different velocity update for the global best particle. If τ is the index of the global best particle, such that $\mathbf{z}_\tau = \widehat{\mathbf{z}}$, then the new velocity update ensures that a point is sampled from the support of a probability measure containing $\widehat{\mathbf{z}}$ or close to $\widehat{\mathbf{z}}$:

$$v_{\tau,j}^{(t+1)} = -p_{\tau,j}^{(t)} + \widehat{z}_j^{(t)} + wv_{\tau,j}^{(t)} + \rho^{(t)}(1 - 2r_2^{(t)}) \tag{3.14}$$

The value ρ is a scaling factor used to generate a random sample space with ρ as its side lengths, with $r_2^{(t)}$ again being uniformly distributed between zero and one. In essence the velocity update resets the particle's position to that of the global best, and adds the current search direction. To this result a random vector from $\rho^{(t)}(1-2r_2^{(t)})$ is added. By combining equations (3.4) and (3.14), the position of the new particle will be

$$p_{\tau,j}^{(t+1)} = \widehat{z}_j^{(t)} + wv_{\tau,j}^{(t)} + \rho^{(t)}(1-2r_2^{(t)}) \tag{3.15}$$

The size of the random search volume is changed by expanding ρ when better function evaluations are successfully found. The sampling volume is decreased when no improvements to the function evaluation is found over time; the smaller volume increases the probability of choosing a variable that gives a better objective function value. If a failure in decreasing the objective function is equivalent to $f(\widehat{\mathbf{z}}^{(t)}) = f(\widehat{\mathbf{z}}^{(t-1)})$, then the value of $\rho^{(t)}$ is adapted after each iteration of the GCPSO algorithm with

$$\rho^{(t+1)} = \begin{cases} 2\rho^{(t)} & \text{if } \#s > s_c \\ \frac{1}{2}\rho^{(t)} & \text{if } \#f > f_c \\ \rho^{(t)} & \text{otherwise} \end{cases} \tag{3.16}$$

The terms $\#s$ and $\#f$ respectively denote the number of consecutive successes and failures, with s_c and f_c being threshold parameters. To ensure the correctness of (3.16), $\#f$ is set to zero if $\#s$ increases from iteration t to iteration $t+1$ of the algorithm. In a similar fashion, $\#s$ is set to zero when $\#f$ increases. A rigorous analysis of GCPSO can be found in [55].

3.3 Concluding

The basic PSO algorithm was discussed in this chapter, and a (by no means exhaustive) number of improvements were shown. In particular, this chapter has focused on improvements to the PSO that are relevant to the rest of this thesis. The GCPSO is of particular interest, since it will be the basis for development of the Converging Linear PSO in Chapter 4. The interested reader is referred to [7, 28, 55], the proceedings of the *Particle Swarm Optimization Workshop (2001)*, and the proceedings of the *IEEE Swarm Intelligence Symposium (2003)* for a thorough treatment of research in Particle Swarm Optimisers.

An overview of unconstrained optimisation was given, but it will only serve as a platform from which PSO will be extended to optimise constrained problems. The following chapter takes care of this extension, by examining and analysing a method of linear constraint handling. Inequality constraints are also taken care of, and finally we not only have a PSO that can train Support Vector Machines, but can also optimise general problems with both linear equality and inequality constraints.