Chapter 3

Particle swarm variants

3.1 Overview

As with any newly proposed optimization algorithm, the original PSOA displayed several shortcomings, which we hinted at in the previous chapter. We will now elaborate on these shortcomings. Furthermore, a number of variants which have been proposed to improve the performance of Kennedy and Eberhart's original PSOA are also presented.

3.2 On 'local' search capability

The original PSO algorithm displays great initial efficiency at converging to the approximate location of the minima during the global search phase (e.g. see Figure 3.1 for a typical example), but after this the convergence rate decreases rapidly when the refined search phase is entered [11, 15]. This decrease in efficiency is caused by the velocity (2.2) not being reduced adequately, leading to detrimentally large distances between sampling points in (2.1). Large velocities also cause the particles to overshoot the search area, a desireable property for the initial global search, but detrimental in the refined search stage. In order to improve the performance during this final phase it is thus necessary to implement methods for decreasing the distance between successive sampling points, by reducing the velocity during the search. This will result in more finely spaced evaluation points and less overshoot. The reduced overshoot will have the effect of concentrating the swarm in a smaller overall search volume. A number of ways have previously been proposed by which the PSO algorithm could be improved.

3.3 Sequential particle swarm algorithm

The following is an outline of the sequential algorithm structure. The original algorithm is modified and implemented in a sequential or asynchronous manner, implying that particle

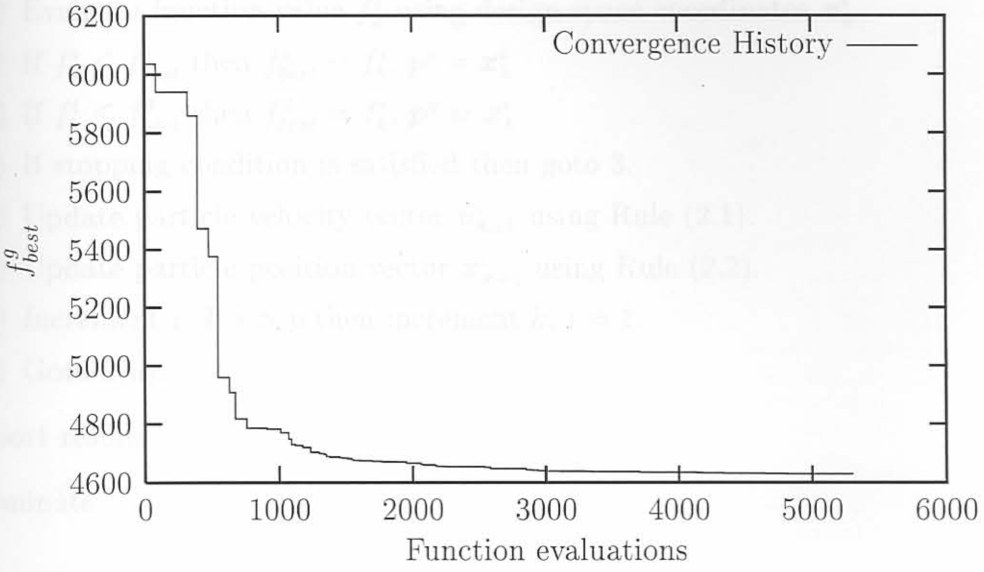

Figure 3.1: Typical history plot

function values, velocities, cognitive and social best remembered positions are updated on a per individual particle basis, rather than a per swarm basis.

Numerical studies by Carlisle and Dozier [20] indicate that the asynchronous method is in general less costly than the synchronous method. The asynchronous method yields improved reaction time to changes in the overall best fitness value and limits unnecessary function evaluations when the stopping condition is satisfied.

This modification was previously proposed by Carlisle and Dozier to limit computational expenses for large swarms [20]. If, for instance, the optimum is found halfway through the swarm's individual particle function value evaluations, the algorithm is stopped without performing the remaining fitness evaluations. Also, if there is an improvement in the swarm best value f^g_{best}, the remainder of the swarm reacts immediately to the swarm best value. (With the classic particle swarm algorithm the improved swarm best value information is only available after the entire swarm's particles have been evaluated in a single pseudo timestep.)

The original position (2.1) and velocity (2.2) rules remain unchanged with this modification. The asynchronous algorithm becomes:

1. Initialize
   (a) Set constants k_{max}, c_1, c_2
   (b) Randomly initialize particle positions $\boldsymbol{x}^i_0 \in \boldsymbol{D}$ in $\mathbb{R}^n$ for $i = 1, ..., p$
   (c) Randomly initialize particle velocities $0 \leq \boldsymbol{v}^i_0 \leq \boldsymbol{v}^{max}_0$ for $i = 1, ..., p$
   (d) Set $k = 1$
2. Optimize

   (a) Evaluate function value f_k^i using design space coordinates $\boldsymbol{x}_k^i$
   (b) If $f_k^i \leq f_{best}^i$ then $f_{best}^i = f_k^i$, $\boldsymbol{p}^i = \boldsymbol{x}_k^i$
   (c) If $f_k^i \leq f_{best}^g$ then $f_{best}^g = f_k^i$, $\boldsymbol{p}^g = \boldsymbol{x}_k^i$
   (d) If stopping condition is satisfied then goto 3.
   (e) Update particle velocity vector $\boldsymbol{v}_{k+1}^i$ using Rule (2.1).
   (f) Update particle position vector $\boldsymbol{x}_{k+1}^i$ using Rule (2.2).
   (g) Increment i. If $i > p$ then increment k, $i = 1$.
   (h) Goto 2(a).

3. Report results

4. Terminate

The asynchronous (sequential) algorithm is also depicted in Figure 3.2.

3.4 Variants on Kennedy and Eberhart's PSOA

The variants on the original PSOA of Kennedy and Eberhart are detailed in this section. By no means are the modifications listed in the following exhaustive. However, they probably represent the most significant and most commonly used variants. In the implementations of the modifications that follow, an asynchronous method for updating the swarm best value $\boldsymbol{p}_k^g$ and particle best value $\boldsymbol{p}_k^i$ is used. A global neighborhood [14, 31] is used throughout when exchanging information about the swarm best values and positions.

3.4.1 Introduction of constant inertia weight

This variant, due to Shi and Eberhart [11], constitutes the first significant variation on the original particle swarm algorithm. An inertia term w is introduced into the original velocity rule (2.2) as follows:

$$\boldsymbol{v}_{k+1}^i = w\boldsymbol{v}_k^i + c_1 r_1 \left(\boldsymbol{p}_k^i - \boldsymbol{x}_k^i\right) + c_2 r_2 \left(\boldsymbol{p}_k^g - \boldsymbol{x}_k^i\right). \tag{3.1}$$

The scalar w performs a scaling operation on the velocity $\boldsymbol{v}_k$, analogous to introducing 'momentum' to the particle. Higher values for w results in relatively straight particle trajectories, with significant 'overshooting' or 'overflying' at the target, resulting in a good global search characteristic. Lower values for w result in erratic particle trajectories with a reduction in overshoot, both desireable properties for a refined localized search.

The most serious drawback of the introduction of constant inertia is the problem dependency of w. In a typical implementation, an intermediate value for w is selected, resulting in a search that is unoptimal during both the 'global' and 'local' phases of the search.

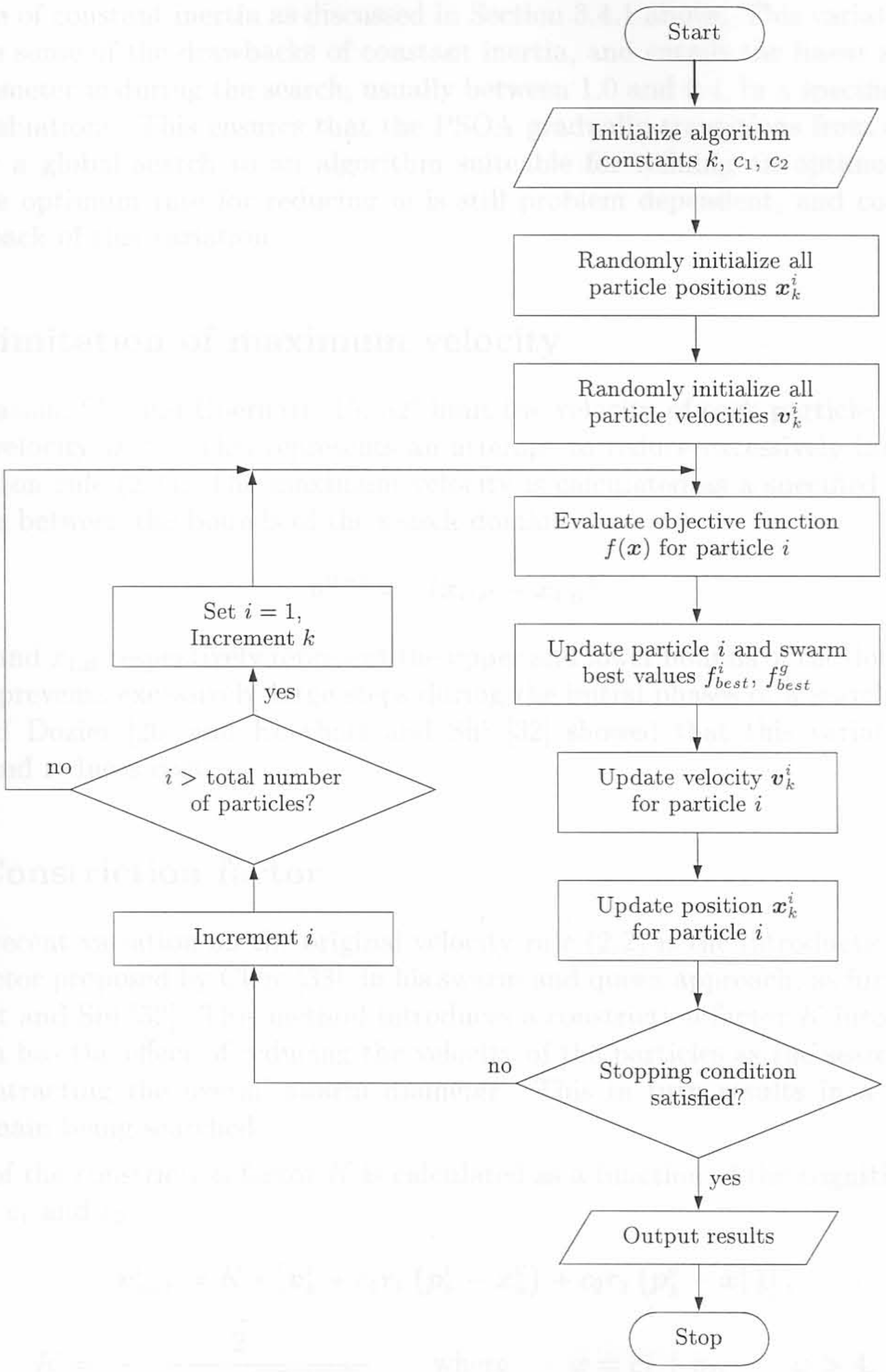
Start
Initialize algorithm constants k, c_1, c_2
Randomly initialize all particle positions x_k^i
Randomly initialize all particle velocities v_k^i
Evaluate objective function $f(x)$ for particle i
Update particle i and swarm best values f_{best}^i, f_{best}^g
Update velocity v_k^i for particle i
Update position x_k^i for particle i
Stopping condition satisfied?
no
yes
Output results
Stop
Increment i
$i >$ total number of particles?
no
yes
Set $i = 1$, Increment k

Figure 3.2: Flow diagram for the sequential particle swarm optimization algorithm

3.4.2 Linear inertia reduction

Linear inertia reduction, also proposed by Shi and Eberhart [11, 15], is a variation on the introduction of constant inertia as discussed in Section 3.4.1 above. This variation attempts to eliminate some of the drawbacks of constant inertia, and entails the linear scaling of the inertia parameter w during the search, usually between 1.0 and 0.4, in a specified number of function evaluations. This ensures that the PSOA gradually transitions from an algorithm suitable for a global search to an algorithm suiteable for refining an optimum in a local search. The optimum rate for reducing w is still problem dependent, and constitutes the main drawback of this variation.

3.4.3 Limitation of maximum velocity

In this variation, Shi and Eberhart [15, 32] limit the velocity of each particle to a specified maximum velocity $\boldsymbol{v}^{max}$. This represents an attempt to reduce excessively large step sizes in the position rule (2.1). The maximum velocity is calculated as a specified fraction γ of the distance between the bounds of the search domain:

$$\boldsymbol{v}^{max} = \gamma(\boldsymbol{x}_{UB} - \boldsymbol{x}_{LB}) \tag{3.2}$$

where x_{UB} and x_{LB} respectively represent the upper and lower bounds of the domain D. This once again prevents excessively large steps during the initial phases of a search. Previously, Carlisle and Dozier [20] and Eberhart and Shi [32] showed that this variation increases reliability and reduces cost.

3.4.4 Constriction factor

A notable recent variation on the original velocity rule (2.2) is the introduction of the constriction factor proposed by Clerc [33], in his swarm and queen approach, as further explored by Eberhart and Shi [32]. This method introduces a constriction factor K into velocity rule (2.2), which has the effect of reducing the velocity of the particles as the search progresses, thereby contracting the overall swarm diameter. This in turn results in a progressively smaller domain being searched.

The value of the constriction factor K is calculated as a function of the cognitive and social parameters c_1 and c_2:

$$\boldsymbol{v}^i_{k+1} = K * \left[\boldsymbol{v}^i_k + c_1 r_1 \left(\boldsymbol{p}^i_k - \boldsymbol{x}^i_k\right) + c_2 r_2 \left(\boldsymbol{p}^g_k - \boldsymbol{x}^i_k\right)\right], \tag{3.3}$$

$$K = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|} \quad \text{where} \quad \varphi = c_1 + c_2, \quad \varphi > 4. \tag{3.4}$$

In their search for an 'off-the-shelf' PSOA, Carlisle and Dozier [20] show that cognitive and social values of $c_1 = 2.8$ and $c_2 = 1.3$ yield good results for their test set.

3.4.5 Dynamic inertia and maximum velocity reduction

This variation, proposed by Fourie and Groenwold [25], aims to reduce the sensitivity to problem dependent parameters associated with previous implementations of inertia [15, 32]. In this approach, a simultaneous dynamic reduction in inertia and maximum velocity is implemented to decrease the swarm domain in a controlled fashion. The approach is outlined as follows: Firstly, the initial inertia w_0 is prescribed, while the initial maximum velocity vector $\boldsymbol{v}^{max}$ is again calculated as a fraction of the domain using (3.2). The swarm domain is then effectively reduced by decreasing the inertia and maximum velocity by fractions α and β respectively, if no improvement in the swarm fitness values $\boldsymbol{p}_k^g$ and $\boldsymbol{p}_k^i$ occur after a to be specified number of iterations h:

$$\text{if } f(\boldsymbol{p}_k^g) \geq f(\boldsymbol{p}_{k-h}^g), \text{ then } w_{k+1} = \alpha w_k, \ \boldsymbol{v}_k^{max} = \beta \boldsymbol{v}_k^{max}, \tag{3.5}$$

with $0 < \alpha, \beta < 1$, prescribed. Rather than reducing the inertia and maximum velocity in a linear fashion, dynamic inertia reduction allows for the adjustment of the algorithm parameters according to the success history of the swarm. For reasons of clarity, we will denote h the 'dynamic delay period' in the remainder of this work.

3.5 Other variants

Several other modifications to the PSOA have been proposed which will be briefly mentioned but not analysed, since they fall outside the scope of this thesis.

3.5.1 Discrete binary particle swarm

A discrete binary particle swarm optimizer was proposed by Kennedy and Eberhart [34], which was benchmarked with a multi modal problem generator against genetic algorithms [19] by Kennedy and Spears. This work was subsequently generalized and applied to the well known traveling salesman problem by Clerc.

3.5.2 Tracking moving extrema

Recently, a modification was proposed by Carlisle and Dozier [35, 36] and Eberhart and Shi [37], whereby a moving extrema in a dynamic problem environment could be tracked.

3.5.3 Hybridizing with other types of algorithms

A number of workers have attempted to improve the PSOA's performance by hybridizing it with other well known methods such as clustering [38], and evolutionary methods [39, 40]. In addition, the PSOA can of course be hybridized with efficient gradient based methods.