Chapter 3

Particle Swarm Optimization

This chapter describes the inception and development of the original particle swarm optimization paradigm. Several aspects concerning the improvement and refinement of the algorithm are reviewed. Particle trajectories and the concept of neighbourhoods in a swarm are briefly discussed. A classification of single-solution particle swarm optimization algorithms is given followed by the description of a number of relevant strategies. Information sharing strategies and subswarm-based algorithms are emphasized.

3.1 Introduction

In keeping with the interest in natural phenomena as the force behind the development of strategies to solve optimization problems, studies of the coordinated movements of bird flocks inspired a fascinating new optimization paradigm. In their landmark publication, *Particle Swarm Optimization* [48], James Kennedy and Russell Eberhart reason that, similar to humans, underlying collective memory could be profitable to bird flocks. Although the original intent of the study was to graphically simulate the movement of a bird flock, the realization that birds flock towards a food source by capitalizing on one another's knowledge, led to the idea that the movement of each agent is guided by two values: the agents's own previous best position as well as the best position of the entire flock. In the case of a bird flock, the objective would be to reach a food source and the best position would be the nearest position to that source.

A unique feature of this paradigm is the association of a velocity with each agent, emulating

the movement of bird flocks through space. Position changes are effected by adjusting the velocity over a number of steps or iterations. The original experiments done by Kennedy and Eberhart [48] added a random amount, weighted by a parameter of the system, to the previous velocity at each step. The velocity is adjusted in the direction of an agent's previous best position, called *pbest*, as well as the group's previous best position, called *gbest*. If the new position of an agent is better, that is, nearer to the food source, pbest is updated. If the new value for pbest proves to be better than the previous best position of the entire flock, gbest is also updated [48]. Results from these experiments showed that, with velocity parameters set to relatively small values, this simulation of a bird flock is very realistic, exhibiting much of the flock's graceful movements before the target, the food source, is reached.

3.2 Particle swarm optimization: The algorithm

Further experiments by Kennedy and Eberhart on their flock simulation yielded a simplified algorithm that could optimize multidimensional functions that proved to be difficult to optimize using conventional methods [48]. In accordance with models developed for applications in artificial life [66], the behaviour of the simulation was perceived to be more like a swarm than a flock. Also, although the initial candidate solutions or agents are only points in the search space, velocities and accelerations are more appropriate for particles. Thus the paradigm was labeled *particle swarm optimization* (PSO).

Because the concept of PSO stems from the observation of group dynamics in nature, it can be classified as a population-based algorithm [28]. Although not similar to evolutionary algorithms, both these paradigms have roots in the observation of natural phenomena. The use of an objective function for evaluating a candidate solution is practised in evolutionary algorithms as well as PSO. However, it must be emphasized that the inception of all these algorithms have only been inspired by the natural phenomena after which, to a certain extent, they have been named and do not purport to simulate nature in all its complexities.

The particle swarm optimizer maintains a population of particles in n-dimensional space. Initially, particles are randomly distributed throughout the search space, where each particle represents a potential solution to an optimization problem. If S is the size of the swarm and i denotes a specific particle, characteristics of i are represented by the following symbols [98]:

$\mathbf{x}_i$: The current *position* of the particle;

$\mathbf{v}_i$: The current *velocity* of the particle;

$\mathbf{y}_i$: The *personal best position* of the particle.

Together with each particle's position and velocity, a personal best position is maintained where the objective function yields the best fitness of all positions visited so far by that particle. Such a personal best position represents part of the memory of a particle, one of the unique characteristics of the PSO paradigm.

In 'flying' through hyperdimensional space, particles continually search for better positions. If such a position is found, the personal best is updated as follows, assuming minimization:

$$\mathbf{y}_i(t+1) = \begin{cases} \mathbf{y}_i(t) & \text{if } f(\mathbf{x}_i(t+1)) \geq f(\mathbf{y}_i(t)) \\ \mathbf{x}_i(t+1) & \text{if } f(\mathbf{x}_i(t+1)) < f(\mathbf{y}_i(t)) \end{cases} \tag{3.1}$$

Another strength of PSO is the concept of social memory. Particles benefit by capitalizing on the knowledge of their neighbours. If a better solution is discovered by a single member, the entire population will move in the direction of that solution. Simultaneously the best position of the entire swarm, or part of it, is updated. The symbol $\hat{\mathbf{y}}$ indicates the best position discovered by any member of the swarm or grouping within the swarm, defined as follows:

$$\hat{\mathbf{y}}(t) \in \{\mathbf{y}_0(t), \mathbf{y}_1(t), \ldots, \mathbf{y}_s(t)\} | \, f(\hat{\mathbf{y}}(t)) = min\{f(\mathbf{y}_0(t)), f(\mathbf{y}_1(t)), \ldots, f(\mathbf{y}_s(t))\} \tag{3.2}$$

The core of the particle swarm optimizer comprises the adjustment of the velocity associated with each particle during a single iteration. The previous velocity vector, the velocity vector towards the particle's personal best (pbest) and the velocity vector towards the entire swarm's best value (gbest), are linearly combined. The functions to be optimized are not always simple, and many suboptimal solutions may exist in the search space. To diversify the search and explore new regions in multidimensional space, random coefficients are introduced. Two independent random sequences, $\mathbf{r}_1 \sim U(0,1)^n$ and $\mathbf{r}_2 \sim U(0,1)^n$, are used. The constants, $0 < c_1, c_2 \leq 2$, called the *acceleration constants*, control the maximum values of $\mathbf{r}_1$ and $\mathbf{r}_2$. The velocity of a particle in a swarm is updated as follows:

$$v_{i,j}(t+1) = v_{i,j}(t) + c_1 r_{1,j}(t)[y_{i,j}(t) - x_{i,j}(t)] + c_2 r_{2,j}(t)[\hat{y}_j(t) - x_{i,j}(t)] \tag{3.3}$$

where $j = 1, \ldots, n$.

It is clear from equation (3.3) that the values of c_1 and c_2 can be manipulated to increase or decrease the movement in the direction of either the personal best or global best positions. Depending on the values of c_1 and c_2, a particle may also fly past the target before being pulled back in the direction of the previous best values.

The velocity is used to update the position of each particle:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{3.4}$$

The original PSO algorithm consists of, after initializing all particle positions, personal best positions and velocities, repeated updating of the velocity and position of each particle. Simultaneously the personal best position of each particle is updated if better values are found. The best position of the entire swarm, or neighbourhood when a subswarm is optimized, is also updated when appropriate. After a number of iterations, or when specific stopping conditions are met, the algorithm terminates and the solution is the last global best position, or the best neighbourhood best position.

3.2.1 Global and local particle swarm optimizers

Eberhart and Kennedy [28] implemented two versions of the initial particle swarm optimizer: the *global* version that keeps track of the best value, *gbest*, of the entire swarm, and the *local* version where the best value of a particle's nearest neighbours are retained and used to update the velocity of those particles [28]. The two versions are called the GBEST and LBEST models respectively. The GBEST PSO is summarized in Algorithm 1.

Algorithm 1 The GBEST particle swarm algorithm

Initialize a population of particles with random positions
  in n dimensions;
Set velocities associated with each particle to 0;
repeat
  for *each particle* $i \in [1, \mathrm{S}]$ **do**
    if $f(S.x_i) < f(S.y_i)$ **then**
      $S.y_i = S.x_i$;
    end
    if $f(S.y_i) < f(S.\hat{y})$ **then**
      $S.\hat{y} = S.y_i$;
    end
    Update velocity using equation (3.3);
    Update position using equation (3.4);
  end
until *stopping condition is true*;

The local version of the particle swarm optimizer is similar to the global version except that

particles can only access information of their nearest neighbours' best positions [28]. The best evaluation found in this group (nbest) is used together with each particle's previous best value (pbest) to calculate a new velocity and new position for each particle. Neighbourhoods consist of a predetermined number of particles adjacent to one another. The notion of adjacency is, however, based on the assumption that particle information is stored in arrays, and that the indices of particles of a neighbourhood are adjacent. Because particle positions are stochastically determined, the positions of the neighbourhood particles are not necessarily adjacent in the search space.

The purpose of this version of the particle swarm optimizer was apparently to create different groups of particles that explore different regions of the search space and so increase the diversity and the ability to optimize a function that may contain local optima. Particles within such a neighbourhood have no relationship to each other, as selection of the particles forming a neighbourhood is based on particle indices. Thus, neighbourhoods overlap and a particle may be part of more than one neighbourhood. The main differences between the GBEST PSO and LBEST PSO are [32]:

- The GBEST PSO converges faster, but swarm diversity is lost.
- As the result of improved diversity, the LBEST PSO is less likely to converge on a local minimum.

Suganthan [95] proposed a number of improvements for the standard PSO algorithm to improve the quality of the solutions as the number of iterations was increased. A variable neighbourhood operator was introduced. Starting with a neighbourhood consisting of a single particle, the size of the neighbourhood is gradually increased to include all particles. The neighbourhood is defined in two different ways: by using particle indices and by choosing a fraction of particles that are physically close to the particle for which a neighbourhood is sought. The second method uses a spatial neighbourhood and can be computationally intensive as Euclidian distances between all particles have to be calculated. Experimental results showed that the quality of the solutions improved when using dynamic neighbourhoods [95].

3.2.2 Cognition-only and social-only models

The particle swarm paradigm originates from the observation of bird flocks where individual members profit from the discoveries and previous experience of the flock in their search for

food [48]. The idea of knowledge sharing can also be found in human societies and its social structures [49].

If the PSO paradigm is viewed as simulating the ability of human societies to process knowledge, two distinct parts can be identified, a cognitive part and a social part. The term $c_1 r_{1,j}(t)(y_{i,j}(t) - x_{i,j}(t))$ is perceived to represent private thinking, as a new position in the search space is calculated using the experience of the best previous position. A particle swarm optimizer using only the cognitive component is known as the *cognition-only model* [49]. For this model particles tend to search the regions where they have been initialized, and behave as independent hill-climbers. The velocity update formula is modified as follows:

$$v_{i,j}(t+1) = v_{i,j}(t) + c_1 r_{1,j}(t)(y_{i,j}(t) - x_{i,j}(t)) \tag{3.5}$$

According to Kennedy [49] and Carlisle and Dozier [17], the cognition-only model is less successful and slower than the model with a cognitive as well as a social component, referred to as the full model. However, the cognition-only model proved to be useful when combined with other techniques to develop a niching algorithm [13] [16].

The velocity update formula for the *social-only* PSO version [49] is:

$$v_{i,j}(t+1) = v_{i,j}(t) + c_2 r_{2,j}(t)(\hat{y}_j(t) - x_{i,j}(t)) \tag{3.6}$$

For the *social-only model*, individuals have no tendency to return to positions that previously proved to be successful for themselves. All particles are attracted towards the best position in their neighbourhood so that this version converges faster [17] [49]. However, depending on the problem, there is a tendency to converge on local optima, as particles does not explore their own neighbourhoods sufficiently.

3.2.3 The inertia weight

The addition of velocity adds several unique characteristics to the particle swarm [50] [88]. Considering the metaphor of a swarm of particles flying through hyperdimensional space, the addition of velocity at every step increases the resulting velocity of a particle until it flies past the target and is pulled back towards previous personal best and neighbourhood best values. Thus the search for an optimum is diversified and other more remote regions of the search space can be explored. However, for the GBEST model it is necessary to control the velocity in order to prevent particles from leaving the search space. Therefore the velocity of a particle is limited by some value V_{max} [49]. Limiting the velocity is also referred to as *velocity clamping*, which is

implemented by clamping speed to control the global exploration of particles. If $V_{max,j}$ is the maximum allowed step size in dimension j, the speed is adjusted before particles positions are updated:

$$v_{i,j}(t+1) = \begin{cases} v'_{ij}(t+1) & \text{if } v'_{ij}(t+1)) < V_{max,j} \\ V_{max,j} & \text{if } v'_{ij}(t+1)) \geq V_{max,j} \end{cases} \tag{3.7}$$

where $v_{i,j}(t+1)$ is computed using equation (3.3).

A value for $V_{max,j}$ should be selected carefully, as the swarm will not explore the search space sufficiently if the value is too small, while a too large value might have the effect that good regions of the search space are not exploited sufficiently.

To control the velocity and to improve the performance of the particle swarm optimizer, a new parameter, the *inertia weight*, was introduced by Shi and Eberhart [88]. It was argued that a very small velocity causes particles to statistically contract to the current global optimum, resembling a local search algorithm. A larger velocity causes exploration of new regions, resulting in a global search ability. Different problems require different balances between the local search ability and global search ability. To implement this, an inertia weight w was brought into the equation as follows:

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_{1,j}(t)(y_{i,j}(t) - x_{i,j}(t)) + c_2 r_{2,j}(t)(\hat{y}_j(t) - x_{i,j}(t)) \tag{3.8}$$

$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1) \tag{3.9}$$

where w is the inertia weight.

The inertia weight acts as a scaling factor that modifies the contribution of the previous velocity to the current update equation. In their initial empirical studies, Shi and Eberhart [88] investigated the effect of different values for $w \in [0, 1.4]$. Algorithms with an inertia weight in the range [0.8,1.2] were found to converge faster. Dynamically changing inertia values, where large inertia values decrease over time to smaller values, also improve the performance of the particle swarm optimizer significantly [88]. These results can be explained by noting that an optimization algorithm generally needs more exploration ability at the beginning of a run while exploitation ability is necessary when the optimum is approached. Further studies by Shi and Eberhart [89] concluded that the selection of the inertia parameter and maximum velocity may be problem-dependent.

The ultimate purpose of PSO is to explore the search space where the objective function is defined and, though the landscape may contain many suboptimal solutions, locate the overall or global best position. Through exploitation of the surrounding search space, the accuracy

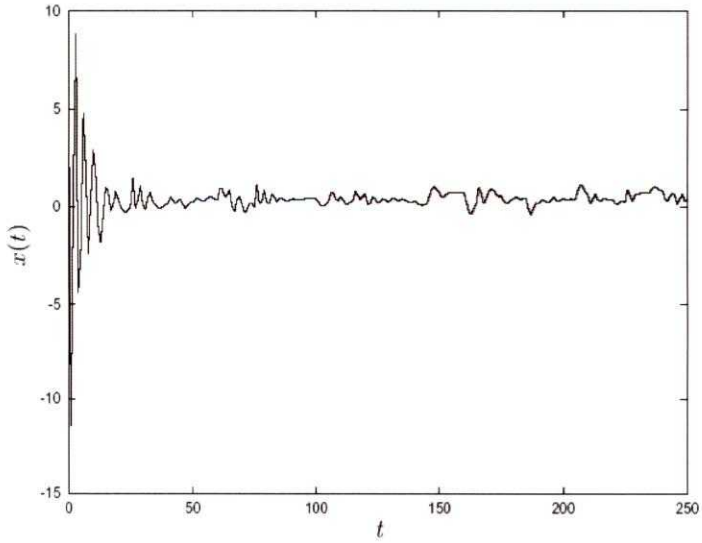


Figure 3.1: Stochastic particle trajectory, obtained using $w = 0.7$ and $c_1 = c_2 = 1.4$ [100]

of the solution is improved. The choice of control parameters such as inertia and acceleration coefficients determines whether the PSO will exhibit divergent, convergent or cyclic behaviour, and whether exploration of the search space is sufficient. Control parameter settings can not be seen in isolation. Choosing a value for w has to be made in conjunction with the selection of values for c_1 and c_2 [33]. Some studies empirically found certain parameter choices to work very well, for example Eberhart and Shi's choice of $w = 0.7298$, $c_1 = c_2 = 1.49618$ [29]. However, these choices should not be generalized as they are based on a limited sample of problems and may also be problem-dependent.

Theoretical studies of particle trajectories played a large part in gaining insight into optimal parameter choices. Ozcan and Mohan [68] conducted formal analyses showing that, in the absence of stochastic influences, the trajectory of a particle follows a sinusoidal wave. Each particle acquires a random frequency and amplitude during the search. The analyses were extended to incorporate a multidimensional and multi-particle system [69].

Particle trajectory studies done by Ozcan and Mohan [68] [69] and Clerc and Kennedy [20] use a simplified PSO system without an inertia term. Van den Bergh and Engelbrecht [100] conducted an analysis of particle trajectories with the inertia weight included. Formal proof is presented that each particle i of a *gbest* PSO converges to a stable point $\mathbf{p}_i$, that is, if $\{\mathbf{x}_i(t)\}_{t=0}^{+\infty}$

is a sequence of particle positions, it is shown that

$$\lim_{t \to +\infty} \mathbf{x}_i(t) = \mathbf{p}_i \tag{3.10}$$

Thus the trajectory of a simple particle with inertia converges to a stable point, which is a weighted average of $\mathbf{y}$ (the personal best position) and $\hat{\mathbf{y}}$ (the global best position).

A heuristic to select the best values for w, c_1 and c_2 could now be derived [100]. Van den Bergh and Engelbrecht found that, to guarantee convergence, the following relation has to be satisfied:

$$w > \frac{1}{2}(c_1 + c_2) - 1 \tag{3.11}$$

An example of the trajectory of a particle using the parameter settings $w = 0.7$ and $c_1 = c_2 = 1.4$ that satisfy the relationship given in equation (3.11) is shown in Figure 3.1 [100].

3.2.4 The constriction factor

Conditions to guarantee the convergence of the particle swarm to an equilibrium state were also studied by Clerc [19]. Clerc and Kennedy [20] proposed the incorporation of a *constriction factor* into the particle swarm update equation to balance the exploration-exploitation trade-off and prevent the velocity from growing out of bounds. Velocities have traditionally been contained by implementing a V_{max} parameter. However, the need for such a parameter is eliminated by the implementation of properly defined constriction coefficients. The effect of a constriction factor is similar to that of the incorporation of an inertia weight, but velocity clamping is not necessary. A value for the constriction factor χ was derived in terms of c_1 and c_2.

The modified velocity update equation is presented in the following equation:

$$v_{i,j}(t+1) = \chi(v_{i,j}(t) + c_1 r_{1,j}(t)(y_{i,j}(t) - x_{i,j}(t)) + c_2 r_{2,j}(t)(\hat{y}_j(t) - x_{i,j}(t))) \tag{3.12}$$

where

$$\chi = \frac{2}{|2 - \phi - \sqrt{\phi(\phi - 4)}|} \tag{3.13}$$

with

$$\phi = \phi_1 + \phi_2$$

$$\phi_1 = c_1 r_1$$

$$\phi_2 = c_2 r_2$$

The purpose of the constriction coefficient, χ, is to reduce the velocity at each time step in order to ensure convergence to a stable point [33]. Applying the constraints $\phi \geq 4$ and $\kappa \in [0, 1]$ guarantees swarm convergence, as χ evaluates to a value in the range $[0, 1]$. The value of κ determines the extent to which either exploration or exploitation dominates behaviour of the swarm. Local exploitation is favoured for $\kappa \approx 0$, resulting in fast convergence, while $\kappa \approx 1$ has the effect that convergence is slow with a high degree of exploration.

Eberhart and Shi compared the performance of particle swarm optimization using an inertia weight with the performance of a PSO with a constriction factor [29]. Empirical results showed that the constriction factor approach incorporating velocity clamping as well, performs better than the inertia weight approach. Comparing the equations representing the inertia weight approach with those representing the constriction factor approach, it can be shown that the inertia weight approach is similar to the constriction factor approach if the inertia weight w is set to χ, and c_1 and c_2 is chosen such that $\phi = c_1 + c_2$ where $\phi > 4$. Theoretical analyses by Van den Bergh and Engelbrecht [100], and Trelea [97] confirmed that careful selection of the inertia weight w as well as c_1 and c_2 result in improved performance.

3.2.5 Neighbourhoods

The discussion of two early PSO models, GBEST and LBEST in section 3.2.1 introduced the concept of a *neighbourhood*. Social structures and communication within social networks can, to a large extent, be seen as the driving force behind particle swarm behaviour. The LBEST version of the PSO as described by Eberhart and Kennedy [28] model social behaviour by making use of neighbourhoods. Overlapping neighbourhoods are created where particles of each are scattered throughout the problem space and will eventually converge to the neighbourhood best and then to the global best. Thus the diversity of the swarm is improved.

Neighbourhoods form a basis for communication within a swarm. Individuals in a neighbourhood influence one another; the more successful an individual is, the greater the influence on other members of the neighbourhood. Thus the quality of the entire neighbourhood will be enhanced. Group performance is influenced by the quality of communication within the group which is again affected by the structure of the social network [51]. In particle swarm optimization neighbourhoods may have different topologies. Particles assigned to a neighbourhood are selected according to some structure based on array indices if the swarm is implemented as an array of particles [51]. The different topologies indicate the degree of connectivity and

the amount of information interchange in a social network. Highly connected networks favour faster convergence, but the danger also exists that local minima may be reached. Sparsely connected networks converge slower, but if clustering occurs, the search space may also not be covered sufficiently [33].

Several social network structures have been empirically studied. A number of these are described below [51] [53] [65]:

The star social structure: A star structure, illustrated in Figure 3.2, describes a social structure where all particles are interconnected. Such a structure corresponds to the original GBEST PSO model. All particles communicate with one another. Information flow through the network is fast, as each individual is attracted to the best solution found so far. Populations tend to converge rapidly, but are susceptible to convergence on local optima [51].

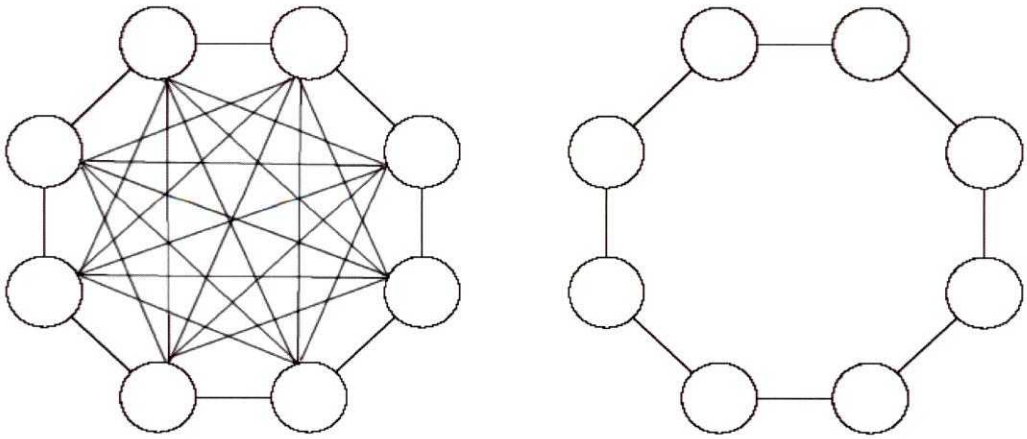
The ring social structure: The original LBEST PSO model uses a ring structure, illustrated in Figure 3.2, to form neighbourhoods. Each individual is affected by its K immediate neighbours. If $K = 2$, each particle communicates with its two adjacent neighbours. Neighbourhoods formed in such a manner will overlap, as adjacency must be understood in terms of particle indices and not physical adjacency in the search space.

The wheel social structure: For the wheel structure, illustrated in Figure 3.2, one individual is connected to all others, which are connected only to that individual. All information flow through this focal point. Too rapid conversion on local optima is prevented by the buffering effect of the focal particle, and the propagation of good solutions is slowed down.

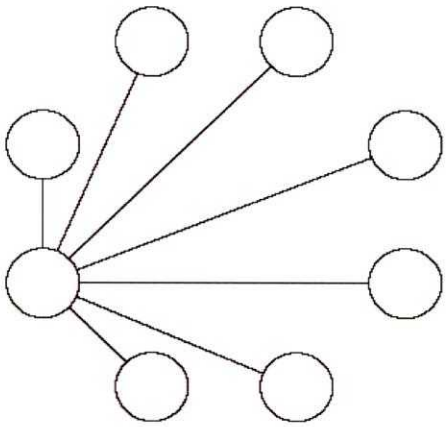
The pyramid social structure: This social structure has the shape of a three-dimensional wire-frame triangle [53]. Compared to the performance of other topologies, it produced relatively good results.

The four clusters social structure: Four clusters of particles are formed with two connections between clusters.

The Von Neumann social structure: Particles are connected in a grid structure [53]. Neighbours above, below, and on each side on a two-dimensional lattice are connected. The Von Neumann social structure was shown to perform very well in a number of empirical studies [53] [73].



(a)Star (b)Ring



(c)Wheel

Figure 3.2: Social network structures

The concept of different neighbourhoods of particles within a swarm is central to the development of a number of algorithms, referred to as *niching algorithms* to solve multi-solution optimization problems. Genetic algorithm niching techniques such as fitness sharing [40], clearing [74], the sequential niching technique of Beasley *et al.* [4] and species conserving genetic algorithms developed by Li *et al.* [56] all use some form of a *niche radius* to demarcate a portion of the swarm which is then optimized separately to converge on either the global or one of the local optima. Particle swarm niching techniques such as NichePSO [13], the species-based PSO [57] and the vector-based PSO developed in this study, also rely on a niche radius in order to divide the swarm into subswarms. These techniques are addressed in depth in chapters 4 and 5.

When neighbourhoods are created to facilitate niching, it is understood to be spatial neighbourhoods. Such neighbourhoods are defined as a subset of a swarm of particles in the same region of the problem space. Suganthan [95] proposed that neighbourhoods be formed on the basis of spatial proximity in order to improve the performance of single-solution PSO. The Euclidian distance between particles determines whether particles are close enough to one another to qualify being in the same neighbourhood. Given neighbourhoods of size n_N, Suganthan defines the neighbourhood of particle i as the n_N particles closest to particle i. The original PSO algorithm is modified to facilitate a finer grained search by employing neighbourhoods. A neighbourhood is created by calculating distances between particles and choosing a number of particles near to particle i. The best particle in the neighbourhood, *lbest*, is calculated. Neighbourhood sizes are gradually increased. Updating each neighbourhood means that each *lbest* moves in the direction of *gbest* but at a much slower rate of convergence, giving the swarm more exploratory power.

Using spatial neighbourhoods is computationally expensive as distances between particles have to be calculated at each iteration. Thus, if neighbourhoods are created to improve diversity, neighbourhoods based on particle indices are preferable.

3.3 Variations on the particle swarm optimizer

Since the inception of the particle swarm optimizer, many variations have been proposed and tested with the purpose of improving the performance of the algorithm [33]. Enhancements to the basic PSO comprise improvements to the accuracy of the solutions, the probability of locating the overall optimal value as well as the extent to which convergence could be guar-

anteed. Single-solution PSO algorithms for continuous, unconstrained problems are classified by Engelbrecht according to the technique that was implemented to improve the performance [33]:

Social-based algorithms: These algorithms use different information sharing strategies. Particles influence one another in different ways, thus changing the way in which the neighbourhood best position is calculated.

Hybrid algorithms: Aspects of evolutionary computation and ant colony optimization as well as strategies such as simulated annealing and gradient descent, are incorporated into the traditional PSO.

Sub-swarm-based: Algorithms are included where the swarm is divided into sub-swarms and manipulated in different ways in order to improve diversity and accuracy of the solutions.

Memetic algorithms: Exploration ability of PSO is improved by the incorporation of other techniques implementing local search.

Multi-start algorithms: Parts of the swarm or the entire swarm are restarted at certain stages.

Repelling methods: Particles are repelled from one another in order to improve diversity.

This study emphasizes optimization algorithms with the ability to locate multiple solutions. Such algorithms are applied to problems described by multi-modal functions in demarcated search spaces. These algorithms are referred to as *niching* or *speciation* algorithms and are described in depth in chapter 4. Some of the strategies used to facilitate niching, have their roots in techniques used in single-solution PSO algorithms. The concept of neighbourhoods as introduced in the LBEST PSO model [28], has already been discussed. A number of relevant single-solution PSO algorithms are presented in this section.

3.3.1 Information sharing strategies

The idea of neighbourhoods form a salient feature of these algorithms. The following strategies are discussed:

$$\sum_{m=1}^{n_{\mathcal{N}_i}} \frac{\mathbf{r}(t)(\mathbf{y}_m(t) - \mathbf{x}_i(t))}{n_{\mathcal{N}_i}}$$

where $n_{\mathcal{N}_i} = |\mathcal{N}_i|$, $\mathcal{N}_i$ is the set of particles in the neighbourhood of particle i and $\mathbf{r}(t) \sim U(0, c_1 + c_2)^n$. To calculate the velocity at the next time step, the above term is added to the current velocity and the result is scaled by either the inertia weight or the constriction factor. For the second approach, the contribution of each particle is scaled by a weight depending on the performance of that particle. Mendes *et al.* tested both approaches on a number of benchmark functions using different population topologies. Both approaches performed better than the standard PSO.

Fitness-distance ratio PSO

The fitness-distance ratio PSO (FDR-PSO), developed by Veeramachaneni *et al.* [102], addresses the problem of premature convergence by adjusting the velocity of each particle towards the best previous positions visited by its neighbours. Similar to the fully informed PSO [65], a particle is most likely to be influenced by more successful individuals in its neighbourhood. However, the influence of multiple other particles may cancel each other, resulting in a reduced possible benefit. The FDR-PSO algorithm counteracts this possibility by selecting only one other particle when updating each velocity dimension. Such a particle is chosen subject to the following criteria:

- It must be near to the particle being updated.
- It should have visited a position with fitness better than the particle being updated.

A simple and robust way to update each velocity dimension comprises selecting a particle that maximizes the ratio of the fitness difference to the one-dimensional distance. For dimension $j = 1, \ldots, n$, a particle referred to as *nbest* is chosen to maximize

$$FDR(i, m, j) = \frac{f(\mathbf{y}_m(t)) - f(x_i(t))}{|y_{mj}(t) - x_{ij}(t)|} \tag{3.14}$$

where FDR refers to the fitness-distance ratio. Veeramachaneni *et al.* [102] showed the FDR-PSO to perform significantly better than the original PSO and several of its variants, when tested on a number of benchmark functions.

The technique used by the FDR-PSO to quantify the influence of multiple particles on the updating equation in a neighbourhood inspired an enhanced speciation algorithm developed by Li [57]. The Fitness Euclidian-distance Ratio-based PSO (FER-PSO) is discussed in chapter 4.

3.3.2 Subswarm-based approaches

Grouping of particles into subswarms can primarily be seen as an effort to diversify the search process. In order to locate the overall optimal position, the entire search space should be explored and premature convergence inhibited. Engelbrecht [33] classifies subswarm-based PSO approaches into cooperative PSO algorithms and competitive PSO algorithms. A selection of cooperative PSO approaches is briefly discussed in this section.

Multi-phase PSO

Løvbjerg *et al.* [60] combined the particle swarm with concepts from evolutionary algorithms to develop two hybrid particle swarm optimizers. These algorithms can be classified as hybrid algorithms with the ideas of subpopulations and breeding incorporated to prevent premature convergence to suboptimal points. However, the breeding between subswarms is a form of cooperative PSO, as genetic material is exchanged between parents of different subswarms [33]. The structure of the hybrid model is illustrated in Algorithm 2.

Algorithm 2 The structure of the hybrid model

```
begin
    initialize;
    while not terminate-condition do
        evaluate;
        calculate new velocity vectors;
        move;
        breed;
    end
end
```

A breeding model as well as a subpopulation model are presented. The breeding model produces offspring by randomly selecting two parents. Offspring are produced by arithmetic crossover on the position of the parents for each dimension. Velocity of the offspring is calcu-

lated as the sum of the velocity vectors of the parents normalized to the original length of each parent velocity vector. Parent particles are then replaced by their offspring particles.

The subpopulation model extends the breeding hybrid PSO model by dividing the particles into a number of subpopulations, each having its own unique best known optimum. Breeding takes place between particles from different subswarms, which could result in an escape from a local optimum.

Al-kazemi and Mohan [1] describes an adaptation of the PSO algorithm to discrete optimization problems. Particles are divided into two subswarms of equal size. At any given time, each particle is in one of two possible phases, an attraction phase and a repulsion phase. Attraction and repulsion refer to the movement of particles towards or away from the global best position. Directions of these movements are controlled by the coefficients in the velocity update equation, which excludes the personal best position:

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 x_{ij}(t) + c_2 \hat{y}_j(t) \tag{3.15}$$

Phase switching takes place either after a set number of iterations, or if no global best fitness improvement is observed in a specified number of steps. Unlike other PSO variants, particle positions are updated by incorporating a hill-climbing procedure. A particle position is permitted to change only if such a change improves fitness.

Cooperative split PSO

Stochastic optimization algorithms such as particle swarm optimizers and genetic algorithms almost always experience reduced performance with increased dimensionality. Van den Bergh and Engelbrecht introduced the cooperative split PSO (CPSO-S_K) [101] to improve the performance of the original algorithm. Multiple swarms are used to optimize different components of the solution vector cooperatively. If each subswarm represents one dimension, the number of subswarms will be equal to the dimensionality, n, of the problem. However, the function to be optimized requires an n-dimensional vector. Therefore subswarms cannot be optimized separately. To address this impediment, a *context* vector is constructed by concatenating the global best particles from each of the n swarms to form an n-dimensional vector. In swarm j, for example, the other $n-1$ components in the context vector are kept constant while the jth component of the context vector is replaced in turn by each particle from the jth swarm.

Van den Bergh and Engelbrecht reported a marked improvement in performance over the standard PSO when CPSO-S_K was tested on several benchmark optimization problems [101].

3.3.3 Memetic PSO algorithms

Empirical studies have shown that many optimization algorithms lack the ability to refine a solution once a promising region of the search space have been located. Mechanisms that have been provided to balance exploration and exploitation can be put into effect to favour exploitation in the later stages of the optimization process. In addition, special features can be incorporated to address this problem. For example, a local optimizer can be embedded between the iterations comprising the search process. Al-kazemi and Mohan [1] incorporated a hill-climbing process in their multi-phase PSO discussed earlier. To reduce computational complexity and yield similar performance, local search methods such as hill climbing can be applied only to neighbourhood best solutions.

Incorporating local search methods is also applicable to niching algorithms, especially since the subswarms and species that form an integral part of many of these algorithms consist of limited numbers of particles.

The guaranteed convergence PSO

The guaranteed convergence PSO (GCPSO) [100] incorporates a local search heurisitc related to basic hill-climbing. The algorithm forms an integral part of NichePSO, a niching PSO developed by Brits *et al.* [13] [14] [16]. NichePSO is discussed in chapter 4.

Particle swarm optimizers have a property that make them susceptible to premature convergence resulting in inaccurate solutions. Van den Bergh and Engelbrecht found that a particle may reach a state where $\mathbf{x}_i = \mathbf{y}_i = \hat{\mathbf{y}}$, meaning the particle's current position, its personal best position and the global best position is the same [100]. In such a case the velocity update will depend only on $wv_{i,j}(t)$, meaning that the previous velocity and inertia factor will be responsible for moving the particle to another position. If previous velocities are very close to zero, all particles will stop moving once they catch up with the global best position. The swarm will converge before a solution is reached. Such behaviour is referred to as *stagnation*.

A new parameter was introduced to the PSO algorithm. If τ is the index of the global best particle so that

$$\mathbf{y}_\tau = \hat{\mathbf{y}}$$

a new velocity update equation for the global best particle was suggested:

$$v_{\tau,j}(t+1) = -x_{\tau,j}(t) + \hat{y}_j(t) + wv_{\tau,j}(t) + \rho(t)(1 - 2r_{2,j}(t)) \tag{3.16}$$

The idea of the new equation is to search in a small region around the best global position, $(\hat{\mathbf{y}}_j)$, for a position with better fitness. The term $-x_{\tau,j}(t)$ resets the particle's position to $\hat{y}_j$. The search direction is the current search direction, thus $wv_{\tau,j}(t)$ is added. The term $\rho(t)(1-2r_{2,j}(t))$ generates random positions in a region with side lengths $2\rho(t)$. ρ is a scaling factor generated as follows:

$$\rho(t+1) = \begin{cases} 2\rho(t) & \text{if \# successes} > s_c \\ 0.5\rho(t) & \text{if \# failures} > f_c \\ \rho(t) & \text{otherwise} \end{cases} \tag{3.17}$$

The number of consecutive failures or successes are given by the terms #failures and #successes respectively. A failure is defined as $f(\hat{\mathbf{y}}(t)) \geq f(\hat{\mathbf{y}}(t-1))$, assuming minimization, while s_c and f_c are threshold parameters, depending on the objective function. More detail can be found in [99].

The position of the global best particle τ is updated by the following equation where the velocity is calculated by the new velocity update equation:

$$x_{\tau,j}(t+1) = \hat{y}_j(t) + wv_{\tau,j}(t) + \rho(t)(1-2r_2(t)) \tag{3.18}$$

The guaranteed convergence particle swarm optimizer (GCPSO) was tested on a number of unimodal and multimodal functions, and significantly faster convergence compared to the original PSO, was found. With smaller swarm sizes the effect was more pronounced. The algorithm can be classified as a local optimization algorithm and would be especially useful when swarms are partitioned into smaller subswarms to be optimized separately.

3.4 Conclusion

This chapter reviewed the inception and development of the particle swarm optimization paradigm. The original algorithm was discussed while an overview of the most important improvements was given. Aspects such as trajectories followed by particles as a result of the associated velocity, were touched upon. Neighbourhoods in the swarm and the forming of subswarms were emphasized, as these concepts form the basis of a number of niching strategies which will be discussed in chapter 4. A number of relevant single-solution PSO algorithms were described with the emphasis on information sharing strategies and subswarm-based algorithms.