Chapter 4

Niching

This chapter describes and elaborates on the concept of niching or speciation. Niching algorithms locate more than one optimum of an objective function in a search space. Genetic algorithm niching techniques are briefly reviewed. A number of techniques that adapt particle swarm optimization to locate and optimize functions with multiple optima are discussed in detail. These algorithms use different strategies to identify candidate solutions, estimate niche boundaries and optimize separate subswarms to converge on different optima. Algorithms where these processes take place sequentially as well as in parallel, have been developed. In the case of parallel niching algorithms, a niche merging component is incorporated. Improvements and refinements of some of the algorithms are also discussed.

4.1 Introduction

Function optimization is usually understood as the process of locating a position where the function has the best possible value. The quest for efficient and effective techniques to accomplish this goal has been described in the preceding chapters. However, for various optimization tasks more than one optimum needs to be located. In engineering design, alternative designs that perform equally well, often exist [4]. If several optima can be located, the user can choose a design by considering criteria that have not been incorporated in the objective function. Another example of multi-solution optimization is locating all the resonance points in mechanical or electrical systems [23]. Depending on the application, the designer will need to maximize or

minimize all such resonances. Solving systems of linear equations is also required in many scientific and engineering problems, for example, in robotics and signal processing. Multi-solution optimization provides an efficient technique to solve such problems.

The Oxford dictionary describes a *niche* as a shallow recess, a definition that can also be used figuratively to indicate a container of entities exhibiting some common traits. This description is used in population-based optimization strategies such as evolutionary algorithms and swarm intelligence when all optima of a multi-modal function have to be located in a demarcated search space. When a function has more than one optimum, the part of the search space where individuals have a natural tendency to converge on a specific optimal solution, is known as a niche. Therefore algorithms designed to locate multiple optima, are referred to as *niching* algorithms.

In keeping with the biological origins of the various kinds of population-based optimization algorithms, the niching metaphor also has its roots in nature [4]. In ecology, a niche describes the relational position of a species or population in its ecosystem. Such an ecological niche describes how an organism or population responds to the distribution of resources and competitors and how those same factors may be altered by the organism or population. Thus different populations survive and can exist together by utilizing the environment in different ways. Different species evolve to fill different ecological niches. Therefore niching algorithms are also referred to as *speciation* or *species-based algorithms*.

Niching has originally been studied for evolutionary algorithms, in particular genetic algorithms [4]. Recently, a number of niching algorithms for particle swarm optimization have also been developed. A new PSO niching algorithm is presented in chapter 5.

4.2 Genetic algorithm niching techniques

Similar to the development of single-solution population-based algorithms, the development of niching methods for evolutionary algorithms preceded that of particle swarm optimization. Much of the concepts and terminology used in the study of niching algorithms, stem from their use with evolutionary algorithms, particularly genetic algorithms. Therefore, a brief discussion of relevant genetic algorithm niching techniques is presented.

Fitness sharing, a strategy to promote stable sub-populations or species, is described by Goldberg and Richardson [40]. The idea comes from natural ecosystems where different species evolve to fill each ecological niche. To implement fitness sharing, it must be known if individ-

uals occupy the same niche. Although the strategy is primarily concerned with encouraging diversity, useful concepts like calculating the distances between all individuals and introducing a *niche radius*, are fundamental to many niching techniques for genetic algorithms as well as particle swarm optimization.

Crowding, originally proposed by De Jong [24], and *deterministic crowding* proposed by Mahfoud [61], are based on the competition for limited resources in a natural population. Deterministic crowding recombine pairs of individuals to produce offspring, that replace their closest parent if the fitness is better. The idea of closeness, determined here by means of the phenotypic distance function, as well as the identification of some form of similarity among individuals, can be found in a number of niching algorithms.

Pétrowski [74] proposed a clearing procedure as a niching method for genetic algorithms. The method was inspired by the sharing of limited resources within subpopulations of individuals characterized by some similarities. The population is divided into subpopulations by first sorting individuals from best to worst according to the fitness values. To determine if individuals belong to the same subpopulation, a dissimilarity measure, referred to as the *clearing radius*, σ, is used. All solutions within distance σ from the best dominant individual are then *cleared*, that is, their fitness values are set to zero. The process is repeated for the next fittest solution, until a list of dominant individuals or *winners* remain. Each of these winners represents the fittest individual in a niche.

The clearing procedure was shown to significantly improve the performance of genetic algorithms applied to multimodal optimization. The use of a radius to identify a niche and the process to determine niches also feature in later algorithms such as the species conserving genetic algorithm [56] and the species-based PSO [57].

4.2.1 A sequential niching technique

A sequential niching technique for multimodal function optimization using genetic algorithms was proposed by Beasley *et al* [4]. For this technique, maximization is assumed. Maxima are located in sequence. However, repetition of the GA does not guarantee that a different solution will be found each time. Therefore, the sequential niching technique uses knowledge gained from previous executions of the optimization algorithm to prevent the region where a solution has been found from being searched again. A PSO niching technique, objective function stretching [71] [72] shows some similarities to the sequential niching technique for genetic algorithms. Therefore, a more detailed description of the said technique is deemed to

be relevant to the study of PSO niching.

The sequential niching technique maintains two fitness functions in order to carry over knowledge from previous executions of the algorithm:

- the original fitness function or *raw fitness function*, F, and
- the modified fitness function, M.

To compute the modified fitness function, a *distance metric* is required to indicate how close two chromosomes are. Beasley *et al* uses the Euclidian distance between two points occupied by individuals in n-dimensional space. The modified fitness function, $M(\mathbf{x})$, for an individual $\mathbf{x}$, is computed by multiplying the raw fitness function, $F(\mathbf{x})$, by a *single-peak derating function*. Initially, $M_0(\mathbf{x}) \equiv F(\mathbf{x})$. The optimization algorithm is run using the modified fitness function, keeping a record of the best individual, $\hat{\mathbf{y}}$, found in the run. The modified fitness function is updated to give a depression in the region near the best individual, producing a new modified fitness function, according to

$$M_{n+1}(\mathbf{x}) = M_n(\mathbf{x}) * G(\mathbf{x}, \hat{\mathbf{y}}) \tag{4.1}$$

where $G(\mathbf{x}, \hat{\mathbf{y}})$ is a single-peak derating function.

Various derating functions can be used, for example, the *power law* and *exponential* functions, respectively defined as

$$G_p(\mathbf{x}, \hat{\mathbf{y}}) = \begin{cases} (d_{x\hat{y}}/r)^\alpha & \text{if } d_{\mathbf{x}\hat{y}} < r \\ 1 & \text{otherwise} \end{cases} \tag{4.2}$$

and

$$G_e(\mathbf{x}, \hat{\mathbf{y}}) = \begin{cases} \exp(\log m * (r - d_{\mathbf{x}\hat{y}})/r) & \text{if } d_{\mathbf{x}\hat{y}} < r \\ 1 & \text{otherwise} \end{cases} \tag{4.3}$$

The above functions assume maximization. For minimization, stretching functions are used.

The distance between $\mathbf{x}$ and $\hat{\mathbf{y}}$, as determined by the distance metric, is given by $d_{\mathbf{x}\hat{y}}$. The values of the derating functions increase as this distance increases. The curve described by the power law function can be concave, convex or linear. The power factor, α, determines how concave ($\alpha > 1$) or convex ($\alpha < 1$) the derating curve is. Curves of the exponential function are concave. In equation (4.3), m is the minimum value of the derating function, G, at which point $d_{\mathbf{x}\hat{y}} = 0$. The value of m also determines how concave the derating curve will be. Smaller values of m produce more concavity.

The niche radius, r, an estimated value depending on the inter-niche distance, requires prior knowledge of the objective function. However, Deb [23] proposed a technique to calculate r, provided that the number of maxima is known or can be estimated. Assume that a hypersphere of radius r surrounds each of the p maxima in the function, and that these hyperspheres do not overlap. The hyperspheres must completely fill the n-dimensional space. If the parameter range for each dimension is normalized to be 0 or 1, r is given by:

$$r = \frac{\sqrt{n}}{2 * \sqrt[n]{p}} \tag{4.4}$$

The purpose of the derating function is to modify the search space such that previously traversed parts of the search space are not re-explored. The derating function reduces the fitness of each individual by an amount that depends on the distance between that individual and each best individual found in previous runs. The modified fitness function now has a minimum imposed in areas where maxima were located, thus preventing the maxima to be located again. Figure 4.1 illustrates the raw fitness function, $F(\mathbf{x}) = sin^2(2\pi\mathbf{x})$, and the modified function after the raw function has been multiplied by the power law derating function with $x = 0.25$, $r = 0.25$, and $\alpha = 2$. In this example two small peaks or *false optima* remain. Larger values for α can reduce the height of these optima, but with too high values maxima of interest may be lost or the solutions may be incorrect.

After each execution of the optimization algorithm, the modified fitness function is updated by:

$$M_{n+1}(\mathbf{x}) \equiv M_n(\mathbf{x}) * G(\mathbf{x}, \hat{\mathbf{y}}) \tag{4.5}$$

Algorithm 3 describes sequential niching genetic algorithm of Beasley *et al.* [4]. A single application of this algorithm is referred to as a *sequence*, since it consists of several runs of the optimization algorithm (a GA or other search technique).

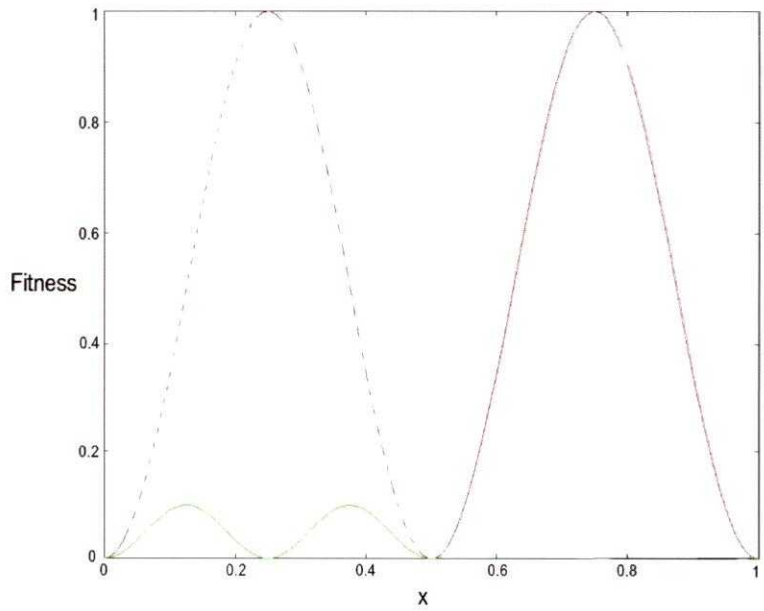

Figure 4.1: Suppression of one peak by a derating function

Algorithm 3 The sequential niching genetic algorithm

Initialize the *modified fitness function* to the initial, or *raw* fitness function;
repeat
  Run the GA to optimize the modified fitness function, keeping a record of the best individual found in the run;
  Update the modified fitness function to give a depression in the region near the best individual, producing a new modified fitness function;
  if *fitness of best individual* > *solution threshold* **then**
    display as a *solution*;
  end
until *the required number of solutions have been found*;

According to Algorithm 3, the best individual found in a run is displayed as a solution if the fitness of that individual is larger than the solution threshold. The *solution threshold* represents the lower fitness limit for maxima of interest. It is assumed that there is a known number, p, of maxima with fitnesses greater than the solution threshold. If the likely fitness values of the maxima of interest are not known, the solution threshold is set to 0 and the algorithm is terminated after the first p maxima have been located. The value of p is set beforehand.

Sequential niching proved to be a vast improvement on fitness sharing methods [4]. However, it does suffer from several drawbacks, listed below:

- Calculation of the niche radius requires the number of maxima to be known in advance. An objective function where the maxima are spread evenly throughout the search space, is assumed.
- False maxima are introduced by the derating function and may be located as solutions.
- To find q maxima, a single-solution optimization algorithm needs to be executed at least q times, which increases the computational complexity.

In order to solve the problem involving false maxima, Beasley *et al* [4] suggested using the modified fitness function to find an approximate solution. Using the original or raw fitness function, a local search method can then locate a more accurate solution.

4.2.2 A species conserving genetic algorithm

Species conserving genetic algorithms by Li *et al.* [56] evolve parallel subpopulations using species conservation. A species is a class of individuals with common characteristics. In natural ecosystems some species may become extinct due to a failure to adapt to a changing environment. However, if these species might be useful to other ecosystems or to humanity, some form of intervention could preserve a few individuals.

Li *et al.* define a species as a subset of the population containing individuals where the distance between any two individuals is less than a parameter, σ_s, the species distance. The species distance is also used to determine which individuals are worth preserving from one generation to the next. The algorithm has close ties to the species-based PSO [57] that will be discussed later in this chapter.

The notion of species is exploited by the species conserving genetic algorithm to achieve niching when optimizing multimodal functions. To locate multiple optima, individuals that are copied into the next generation have to include highly fit individuals, as well as individuals that, while not highly fit, are different enough from the current best individuals to be worth keeping. To establish which individuals have to be conserved, the population is partitioned into several species. Each species is dominated by an individual called the *species seed*. X_s denotes the set of species seeds found in generation t. To build the set X_s, each individual in generation t is considered successively in decreasing order of fitness. If X_s does not contain

any seed that is closer to that individual than half the species distance, $\sigma_s/2$, the individual becomes a new species seed and is added to X_s. The procedure for determining the species seeds is performed for every generation in a GA run.

Once all the species seeds have been found, the new population is constructed by applying the usual genetic operators, i.e. selection, crossover and mutation. Some species may not be fit enough to survive following these operations, but is nevertheless worth keeping. To enable them to survive, the species seeds in X_s are copied into the new population. Species are conserved as follows [56]:

- Mark all individuals as unprocessed.
- The new generation, $G(t+1)$, is searched for solutions belonging to the same species as each species seed $\mathbf{x} \in X_{(s)}$ identified in the previous generation.
- Species seed $\mathbf{x}$ replaces the worst of these "similar" solutions, provided $\mathbf{x}$ has better fitness and is marked as processed.
- If there are no solutions in the same species as $\mathbf{x}$ in $G(t+1)$, $\mathbf{x}$ replaces the worst unmarked solution in $G(t+1)$.
- As the species seeds are drawn from the previous generation, the number of species seeds N_s is always less than the population size, N, and therefore unmarked solutions must always exist.

Thus species seeds found in the current generation are conserved by moving them into the next generation.

Solutions to a multimodal optimization problem can, depending on the objective function, be a number of global optima or a number of dissimilar high-quality solutions. After the generation loop of the species conserving genetic algorithm is exited, these optima are identified by defining a *solution acceptance threshold*, $r_f \in [0, 1]$. Solutions are all individuals $\mathbf{x} \in X_s$ that satisfy

$$f(\mathbf{x}) \geq |(f_{max} - f_{min})|r_f \tag{4.6}$$

where f_{max} is the fitness of the most fit species seed and f_{min} is the minimum fitness in the final population.

4.3 PSO niching techniques

The purpose of the original particle swarm algorithm was to find a single optimum of an n-dimensional function which might be difficult to optimize in any other way [48], [51], [88]. Suboptimal solutions may be encountered while a PSO is in the process of converging, but niches are not formed where a subswarm can converge on such a suboptimum. One of the main reasons why niche formation is inhibited, is the influence of the social component of the velocity update equation [34]. Particles are attracted to the single best solution in a neighbourhood. Therefore, if a better solution is encountered, the entire swarm changes direction towards this more promising area in the search space. In the case of the GBEST PSO, the neighbourhood is the entire swarm, and all particles are attracted to the swarm's global best position. Therefore the GBEST PSO is incapable of niching [34]. The LBEST PSO constructs overlapping neighbourhoods consisting of a particle and its l immediate neighbours. Due to the overlapping neighbourhoods, all particles will eventually converge on the same point. Engelbrecht *et al.* [34] have empirically shown that the standard LBEST PSO is inefficient in locating and maintaining multiple solutions and cannot be used as a niching algorithm . Hence, if the problem is such that all the optimal local and global solutions in the range of values where the function is being investigated, are required, the algorithm needs to be modified.

The idea of optimizing portions of a swarm separately has been implemented in a variety of PSO algorithms, mainly to improve diversity. Eberhart and Kennedy introduced a local version of the original PSO model, called the LBEST model, where the velocity of each particle is updated depending on its own personal best value as well as the best solution in a topological neighbourhood, called lbest [28]. Particles constituting a neighbourhood are selected using particle indices. Most of the particles in these neighbourhoods will eventually converge, but convergence is slowed down, ensuring a better chance to locate a good solution. Suganthan proposed a partitioning scheme based on the spatial location of particles [95], and found that performance was improved for a number of test functions. Various other population structures and neighbourhood topologies were implemented, all with the purpose of improving the performance of the particle swarm optimizer [51], [52], [53].

The notion of neighbourhoods form a natural starting point when designing niching algorithms. A neighbourhood can be conceptualized as a collection of particles that will eventually converge on some optimum. However, to facilitate niching, spatial neighbourhoods will be more appropriate.

This section reviews a number of PSO niching strategies.

4.3.1 Objective function stretching

A function stretching technique proposed by Parsopoulos *et al.* was originally devised to overcome the problem of occasional convergence to local optima [71]. The principle of function stretching was also used by Parsopoulos and Vrahatis to modify the particle swarm optimizer in order to locate all the global minima of a function [72]. In some applications several minima exist where the minimum value of the function is similar. According to Parsopulos and Vrahatis [72], the original particle swarm will exhibit a cyclic movement over the search space in such cases, being unable to converge on one minimum. Such problems, as well as instances where a global minimum as well as some suboptimal solutions need to be located, can be solved by using the stretching technique.

The technique, which assumes minimization, exhibit some similarities to that of Beasley *et al.* [4]. Each time a minimum is discovered, the original function landscape is transformed by applying a stretching function to the original function. The PSO is prevented from returning to the previously discovered minimum, as the function landscape has changed and all local minima above the current minimum has been eliminated. Minima below the current minimum are not affected.

The following two-stage transformation to a new objective function, $H(\mathbf{x})$, is applied soon after a local minimum $\mathbf{x}^*$ of the function f has been detected:

$$G(x) = f(\mathbf{x}) + \gamma_1 \frac{\|\mathbf{x} - \mathbf{x}^*\| \cdot (sign(f(\mathbf{x}) - f(\mathbf{x}^*)) + 1)}{2} \tag{4.7}$$

$$H(\mathbf{x}) = G(\mathbf{x}) + \gamma_2 \frac{sign(f(\mathbf{x}) - f(\mathbf{x}^*)) + 1}{2\tanh(\mu(G(\mathbf{x}) - G(\mathbf{x}^*)))} \tag{4.8}$$

where γ_1, γ_2 and μ are arbitrary chosen positive constants and the sign function is defined as:

$$sign(p) = \begin{cases} -1, & \text{if } p < 0 \\ 0, & \text{if } p = 0 \\ +1, & \text{if } p > 0. \end{cases} \tag{4.9}$$

The sign function can also be approximated by

$$sign(p) \simeq \tanh(\lambda p)$$

for large values of λ.

The algorithm that uses the objective function stretching technique locate minima sequentially. Each time a minimum is discovered, the value of $\mathbf{x}^*$ is recorded. The function $f(\mathbf{x})$ is

set equal to $H(\mathbf{x})$ and all the particles in the swarm are re-initialized. Since particles have converged and will not be able to discover new minima, re-initialization is required. New random positions and personal best positions are calculated for all particles. The entire process is repeated until some stopping criterion is met. If the number of minimizers is known, the algorithm can run until all minimizers have been found. If the number of minimizers is unknown, a specific number can be requested, or the algorithm can run until the maximum number of iterations is reached.

However, Van den Bergh [98] has shown that the stretching technique introduces false local minima as well as misleading gradients. PSO exhibits a tendency to move 'down' a slope, and the function landscape of the new objective function, $H(\mathbf{x})$, will be such that the slope leads away from the minimizer. Therefore the PSO may converge to a boundary of the search space instead of the minimum.

4.3.2 The nBest PSO

Brits *et al.* developed a particle swarm optimization algorithm, *nbest*, to solve systems of unconstrained equations [12] [14]. While the standard *gbest* PSO easily solves systems of equations with one optimal solution, systems with multiple solutions require the implementation of a niching strategy. Brits *et al.* adapted the standard *gbest* PSO algorithm to locate multiple solutions in one run of the algorithm.

Using PSO requires solving systems of equations to be restated as an optimization problem. Each particle represents a candidate solution for each parameter in a system of equations. For example, in a system of two equations with two variables, x_1 and x_2, values for these variables have to be found, indicating positions where the lines intersect. A particle's fitness is determined by how close it is to a solution. Each equation of the system is converted to represent an error function for that equation. Therefore, for a system of two equations, where f_1 and f_2 are functions representing the error,

$$f(x_1, x_2) = |f_1(x_1, x_2)| + |f_2(x_1, x_2)| \tag{4.10}$$

The objective is to minimize $f(x_1, x_2)$ which is straightforward if there is only one solution. However, when the system of equations is such that it has more than one solution, a niching technique must be considered. The standard particle swarm optimizer is modified by initially extending the fitness function to reward a particle when that particle is close to any of the possible solutions in the system of equations. The fitness function for solving a system of K

equations,

$$f(\mathbf{x}_i) = \sum_{k=1}^{K} |f_k(\mathbf{x}_i)| \tag{4.11}$$

is redefined for multiple solutions. For example, if a system of linear equations consists of three equations that intersect in turn at three different positions, the fitness of particle $\mathbf{x}_i$ is calculated by equation (4.11) for each intersection, using the two linear equations of which the lines intersect. The fitness of $\mathbf{x}_i$ is the minimum of the three results. The new fitness function is then defined as:

$$f(\mathbf{x}_i) = \min(f_k(\mathbf{x}_i)) \tag{4.12}$$

An additional modification to the standard PSO model is the definition of neighbourhoods. While the *lbest* model uses topological neighbourhoods to diversify the search for better solutions and slow down convergence before locating a single optimum solution, the *nbest* model uses a different approach. An *Euclidian neighbourhood* is defined for each particle $\mathbf{x}_i$ as the n_i closest particles to $\mathbf{x}_i$. The Euclidian distances between $\mathbf{x}_i$ and other particles in the swarm are calculated to find the closest particles. The *neighbourhood best (nbest)*, $\hat{\mathbf{y}}_i$, of each Euclidian neighbourhood is defined as the center of mass of the positions of all the particles in the neighbourhood:

$$\hat{\mathbf{y}}_i = \frac{1}{k} \sum_{j=1}^{k} B_{i,j} \tag{4.13}$$

where the set B_i consists of the k closest particles to $\mathbf{x}_i$.

The number of particles in a neighbourhood is a user-defined parameter, k, that has to be chosen carefully. Considering the definition of $\hat{\mathbf{y}}_i$, k should not be too small, as a particle will blindly trail its closest neighbour. If k is too large, the algorithm will become similar to GBEST, yielding no information about a possible good result. Particles are updated as for the standard particle swarm optimizer, but the global best particle, $\hat{\mathbf{y}}$, is replaced with $\hat{\mathbf{y}}_i$. Therefore, particles move towards their neighbourhood bests, while neighbourhoods shrink over time. Eventually particles converge on optima in their regions, resulting in multi-solution optimization.

The *nbest* algorithm performed well when tested on a number of systems of linear and non-linear equations, as well as on other multimodal functions. A small inertia weight and

relatively large values of c_1 and c_2 allowed the particles to explore the search space more thoroughly, preventing premature convergence.

4.3.3 NichePSO

The *NichePSO* algorithm developed by Brits *et al.*, is a niching particle swarm optimizer where multiple solutions are located in parallel [13], [14], [16]. Multiple swarms are grown from an initial particle population. As niches are detected, subswarms are formed. Eventually each subswarm represents one of the potential solutions to the problem.

As in the case of the standard PSO, the swarm is initialized to distribute particles uniformly throughout the search space. The *NichePSO* uses *Fauré* sequences to generate initial particle positions. At this stage no niches have been detected and the entire swarm is referred to as the main swarm. All particles are trained using one iteration of the *cognition only model*. Only the cognitive component is used to update particle velocities [49]. Therefore particles move in the direction of more promising regions of the search space, facilitating subswarm formation.

Niching algorithms rely to a large extent on a good strategy to identify initial candidate solutions in the search space. Parsopoulos *et al.* identify potential solutions by using a threshold value, ϵ, where $f(\mathbf{x}_i) < \epsilon$. However, the effectiveness of this approach cannot be guaranteed where the objective function's landscape is unknown. *NichePSO* uses a similar approach to identify candidate solutions but, instead of using a threshold ϵ, the fitness of a particle is monitored by tracking its normalized standard deviation over a number of iterations. If the particle's fitness changes very little, that particle is identified as a candidate solution and a subswarm is created with the particle's closest topological neighbour.

Formally, the standard deviation, σ_i, of the fitness of particle i is tracked over a number of iterations, e_σ, set to 3 in the experiments of Brits *et al.* [14] [16]. A subswarm is created when $\sigma_i < \delta$, where δ is a small problem-dependent value. To avoid problem dependence, σ_i is normalized according to possible maximum and minimum values. The closest neighbour to particle $\mathbf{x}_i$ is particle $\mathbf{x}_k$, where c, the distance between the particles, is defined as:

$$c = \arg\min_{\mathbf{x}_k}\{\|\mathbf{x}_i - \mathbf{x}_k\|\} \tag{4.14}$$

where k is the index of any particle in the main swarm, with $k \neq i$.

Each subswarm initially consists of two particles, namely the candidate solution that acts as the neighbourhood best value, and its closest neighbour. While the main swarm is still searching the solution space, using the cognitive-only PSO for updating particle positions,

subswarms are updated separately. Initially, a subswarm consists of very few particles in a small region. To prevent swarm stagnation and premature convergence of these subswarms, the guaranteed convergence particle swarm optimization (GCPSO) algorithm [100] is used to update particle positions. GCPSO is described in Chapter 3 of this thesis.

To demarcate the subswarms that have been created, a subswarm radius is maintained and updated during iterations. Initially, the radius of each subswarm is the Euclidian distance between the candidate solution or the neighbourhood best value, and its closest neighbour, the only other particle in the subswarm and situated on the boundary of the subswarm. While the subswarms are updated, particles from the main swarm are updated at the same time, and will move towards better positions. Should such a particle move into a region already occupied by a subswarm, the particle will be absorbed into that subswarm. A particle i is absorbed into a subswarm S_j when

$$\|\mathbf{x}_i - \hat{\mathbf{y}}_{S_j}\| \leq R_j \tag{4.15}$$

where $\hat{\mathbf{y}}_{S_j}$ is the neighbourhood best position of subswarm S_j. R_j signifies the radius of subswarm S_j, and is defined as

$$R_j = \max\{\|\hat{\mathbf{y}}_{S_j} - \mathbf{x}_{S_{j,i}}\|\} \tag{4.16}$$

where $\mathbf{x}_{S_{j,i}}$ is a particle in subswarm S_j.

Existing subswarms may also attempt to optimize the same solution, given those subswarms have been created from particles near to each other. These subswarms are merged when the hyper-space defined by their particle positions and radii intersect in the search space, creating a new larger swarm. Subswarms S_{j_1} and S_{j_2} *intersect* when

$$\|\hat{\mathbf{y}}_{S_{j,1}} - \hat{\mathbf{y}}_{S_{j,2}}\| < (R_{S_{j,1}} + R_{S_{j,2}}) \tag{4.17}$$

In the case where the radii of both subswarms are zero, the subswarms are merged if

$$\|\hat{\mathbf{y}}_{S_{j,1}} - \hat{\mathbf{y}}_{S_{j,2}}\| < \mu \tag{4.18}$$

where μ is a small number, normalized to the interval [0,1] in order to make the merging process more generic.

The NichePSO algorithm is summarized in Algorithm 4 [13] [14] [16].

Algorithm 4 NichePSO Algorithm

```
Create and initialize a n-dimensional main swarm, S;
repeat
    Train the main swarm, S, for one iteration using the
      cognition-only model;
    Update the fitness of each main swarm particle, x_{S,i};
    for each subswarm S_j do
        Train subswarm particles, x_{S_j,i}, using a full model PSO
          which incorporates the social component;
        Update each particle's fitness;
        Update the swarm radius R_{S_j};
    end
    if a merging condition is satisfied then
        Merge the corresponding subswarms;
    end
    Allow subswarms to absorb any particles from the main swarm
      that moved into the subswarm;
    if a particle in the main swarm converged then
        Create a new subswarm with this particle and its closest
          neighbor;
    end
until stopping condition is true;
```

NichePSO was tested on a select group of benchmark functions. Four one-dimensional functions were investigated in the range $x \in [0, 1]$, where each function had 5 optima with differing positions and fitness of the optima. The two-dimensional modified Himmelblau function was tested in the region $x_1, x_2 \in [-5, 5]$, where the function has four equal optima [13] [14] [16]. Experimental results showed that NichePSO successfully located and maintained multiple optimal solutions. However, performance of the algorithm does depend on tunable parameters μ and δ. Merging of subswarms is controlled by the value of μ [14] [16]. A too small value of μ will impede the merging of subswarms while subswarms that are supposed to converge on separate optima, will merge when the threshold approaches the interniche distance. Therefore, if μ is too large, not all optima will be located. According to Brits *et al.* [16], μ should not be greater than the smallest interniche distance for optimal NichePSO performance.

Brits *et al.* [14] [16] investigated the sensitivity to changes in δ as well. δ is a parameter used as an indication of whether a particle could be identified as a candidate solution. Results showed that NichePSO is not dependent on a finely tuned δ. Smaller δ values effected a slight

increase in the number of fitness function evaluations, but did not influence the number of optima that was located.

Scalability of the NichePSO

The NichePSO algorithm discussed in the previous section showed good results when tested on one and two-dimensional multimodal functions. Brits *et al.* also investigated NichePSO's ability to scale to massively multimodal functions [14] [15] [16]. NichePSO was tested on the Griewank and Rastrigin functions. Both these functions have a single global minimum and local optima at regular intervals along each dimension. For a specific dimension size, the number of optima increases exponentially as the number of dimensions increases. With a corresponding increase in swarm size, NichePSO performed consistently with slight degradation as the number of dimensions increased [15].

4.3.4 The species-based PSO

The notion of different species existing together in a population was the inspiration behind a PSO for solving multimodal optimization problems [57]. In Biology a species can be described as a *group of actually or potentially interbreeding individuals who are reproductively isolated from other such groups.* The concept of species that evolve to fill a role referred to as an ecological niche, has been used in Goldberg and Richardson's fitness sharing genetic algorithm [40], the sequential niche technique of Beasley *et al.* [4], and the species conserving genetic algorithm (SCGA) [56].

Li [57] proposed the species-based PSO (SPSO) that incorporates the idea of classifying the population into groups or species. The best-fit individual, or particle in the case of PSO, in a species is referred to as the species seed. The definition of a species also includes a parameter, r_s, referred to as the species radius which is the Euclidian distance from the species seed to the boundary of the species. All particles that fall within the r_s distance from the species seed, are part of that species. Figure 4.2 illustrates how particles are assigned to species by using a species radius.

The algorithm determining the species seeds for the SCGA, introduced by Li *et al.* [56], is adopted by the species-based PSO. In PSO terminology the neighbourhood best (*lbest*), the best-fit particle in a neighbourhood, is similar to the species seed for that species. Species seeds are determined by first sorting all particles $\mathbf{x}_i$ in decreasing order of fitness. Initially the set of

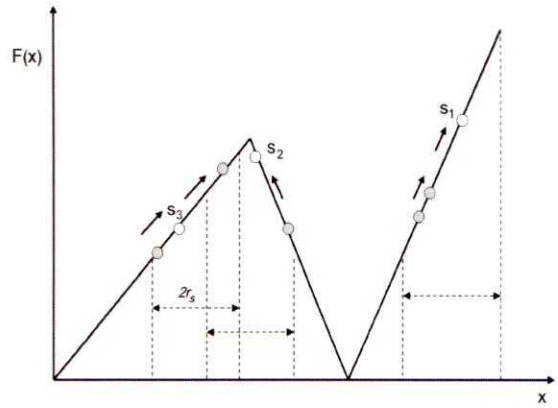

Figure 4.2: Determining the species seeds from the population at each iteration [57]

species seeds S is empty. Particles are checked in turn from best-fit to least-fit against species seeds found so far. The best-fit particle in the swarm is the first species seed and is added to S. Particles falling within a pre-determined radius of that seed, is part of that species. If a particle is found that falls outside the radius of that seed, the particle is added to S as a new seed. As S is built, subsequent particles are checked against all the species seeds in S. Only particles that do not fall within the niche radii of any of the seeds in S, are added to S as new seeds. The set of species seeds is complete once all particles have been checked.

Once the species seeds have been identified from the population, particle positions are updated using the species seeds as the neighbourhood best positions of each corresponding species. Each iteration consists of determining the set of species seeds and adjusting all particle positions once. The process is repeated for a number of iterations enabling the particles to converge on several optima in parallel.

Algorithm 5 formalizes the process to determine species seeds. The species-based PSO that incorporates the process to determine species seeds is given in Algorithm 6.

Algorithm 5 Determining the species seeds

Input: L_{sorted} - a list containing all particles $\mathbf{x}_i$ sorted in decreasing order of fitness if maximization is assumed
Output: S - a set containing particles s_j identified as species seeds

```
begin
    S = φ;
    while not reaching the end of L_sorted do
        found ← FALSE;
        for all s_j ∈ S do
            if distance d(s_j, x_i) ≤ r_s then
                found ← TRUE;
                break;
            end
        end
        if (not found) then
            let S ← S ∪ {x_i}
        end
    end
end
```

Algorithm 6 The species-based PSO

```
begin
    Create an initial population with randomly generated particles;
    repeat
        Evaluate all particle individuals in the population;
        Sort all particles in descending order of their fitness values
          i.e., from the best-fit to least-fit ones;
        Determine the species seed using Algorithm 5 for the current population;
        Assign particles identified in each species to the corresponding species seed;
        Adjust particle positions according to the PSO update equations;
    until termination condition is met;
end
```

The species-based PSO was tested on several benchmark functions [57]. The test functions used were those suggested by Beasley *et al.* [4], namely four one-dimensional functions, as the two-dimensional Himmelblau function, and the Rastrigin function in different dimensions. Tests showed that SPSO can find all the optima for all these functions in one and two dimensions reliably and with good accuracy [57]. However, one drawback of this approach is the size of

the species radius, r_s. Each objective function requires a unique species radius that has to be set in advance. Thus prior knowledge of the objective function is implied. For an unknown function landscape the algorithm can be run with varying values of the species radius. The species radius yielding the best performance is then chosen for that specific problem. Using this approach, the algorithm could still be described as robust and accurate.

However, in all these functions distances between niches do not differ much. Niche shapes are relatively symmetrical. If the shapes and sizes of niches differ a lot, niches require different niche radii. In addition, asymmetrical niche shapes require a different technique to determine the boundaries of the species. Thus the reliability of a single niche or species radius may be challenged when the function landscape is more convoluted and different niches have different niche radii.

4.3.5 The adaptive niching PSO

A niching method that removes the need to specify the niche radius in advance was proposed by Bird and Li [5]. Niching parameters are determined adaptively during a run. The method is called the *adaptive niching PSO* (ANPSO). Initially, the algorithm calculates the average distance, r, between each particle and its closest neighbour. This value is used to determine the formation of niches. ANPSO uses an undirected graph, g, with particles as nodes to keep track of the minimum distance between particles over a number of steps. At each iteration, an edge is added to g between every pair of particles that have been closer than r to one another during the last 2 steps. Niches are formed from the connected subgraphs of g, while all unconnected particles remain outside any niches. Particles can also be added to existing niches during a run. Thus the neighbourhood topology is redefined at every step. Once niches are determined, a GBEST topology is used to update particles in each niche, while a Von Neumann topology is used to update particles that have not yet been assigned to a niche. Therefore, particles that have formed a niche will tend to perform a local search around an optimum, while particles outside any niche will continue searching the whole problem space.

The ANPSO removes the need to specify a niche radius in advance, as it is calculated from the distances between particles. To prevent too many particles from converging on the same optimum, a limit is placed on the number of particles allowed in a single niche. Owing to the calculation of distances between particles, the algorithm is computationally more expensive than most other niching techniques, but, according to Bird and Li [5], the cost is offset by the time taken to tune the niche size parameter in other niching algorithms. The algorithm

performed well on a number of benchmark functions.

Algorithm 7 The adaptive niching PSO

```
begin
    Create an initial population with N randomly generated particles;
    for all particles x_i do
        Calculate distances
            d_i = min_{j≠i} ||x_i − x_j||;
        Calculate average distance
            r = (Σ_{i=1}^{N} d_i) / N;
    end
    Create an undirected graph g containing a node for each particle x_i;
    repeat
        for all particles x_i do
            if d_i < r during last 2 steps then
                Add and edge to g between x_i and nearest particle;
                if one of the particles is already assigned to a niche then
                    Add the other particle to the niche;
                else
                    Create a niche from x_i and nearest particle;
                end
            end
        end
        Update particles in each niche using GBEST;
        Update particles outside any niche using Von Neumann topology;
    until termination condition is met;
end
```

4.3.6 The fitness Euclidian-distance ratio-based PSO

Another effort to address the difficulty in pre-specifying parameters used for estimating how far apart optima in multimodal functions are, has been proposed by Li [58]. The fitness Euclidian-distance ratio-based PSO (FER-PSO) encourages the survival of fitter and closer particles. The FER-PSO was inspired by the fitness distance ratio-based PSO (FDR-PSO) of Veeramacheneni *et al.* [102]. The FDR-PSO was originally designed to locate a single global optimum. A new term based on FDR values was added to the canonical PSO velocity update equation. A canonical PSO modifies the velocity of each particle iteratively by its personal best position

and the position of the best particle from the entire swarm. As a result, each particle searches around a region defined by these two positions, and the influence from other fitter and nearer particles is increased. However, the concept of attracting each particle towards a fitter and closer point in the neighbourhood can effectively be applied to multimodal optimization a well.

FER-PSO uses two swarms, a *memory-swarm* formed by personal bests of particles, $\mathbf{y}_i$, and an *explorer-swarm* consisting of the current positions of particles, $\mathbf{x}_i$. The latter continues to explore the search space while the former acts as a stable repository where the best positions found so far are retained. Each point in the memory-swarm can be improved by moving towards its *fittest-and-closest* point. As the FER-PSO is designed for multimodal optimization, each particle is attracted towards a *fittest-and-closest* neighbourhood point, identified by computing the fitness Euclidian-distance ratio of that particle and all other particles in the swarm. The effect of such a strategy is that particles would form niches naturally around multiple optima, given that there are sufficient numbers of particles.

The fitness Euclidian-distance ratio is calculated as follows:

$$FER_{(j,i)} = \alpha \cdot \frac{f(\mathbf{y}_j) - f(\mathbf{y}_i)}{\|\mathbf{y}_j - \mathbf{y}_i\|} \tag{4.19}$$

where α is a scaling factor to ensure that neither the fitness nor the Euclidian distance becomes too dominated over one another.

For each particle with personal best $\mathbf{y}_i$, the maximum fitness Euclidian-distance ratio found will indicate the particle identified as the neighbourhood best $\hat{\mathbf{y}}$ to be used in the canonical PSO velocity update equation for particle i, namely

$$\mathbf{v}_i = \chi(\mathbf{v}_i + \mathbf{R}_1[0, \frac{\varphi_{max}}{2}] \otimes (\mathbf{y}_i - \mathbf{x}_i) + \mathbf{R}_2[0, \frac{\varphi_{max}}{2}] \otimes (\hat{\mathbf{y}} - \mathbf{x}_i)) \tag{4.20}$$

where $\mathbf{v}_i$ is the velocity of particle i, and χ is the constriction factor, used to prevent each particle from exploring too far away in the search space. $\mathbf{x}_i$ is the current position of particle i, $\mathbf{y}_i$ the personal best position of particle i, and $\hat{\mathbf{y}}$ is the best position found so far in the entire swarm. $\mathbf{R}_1[0, \frac{\varphi_{max}}{2}]$ and $\mathbf{R}_2[0, \frac{\varphi_{max}}{2}]$ are two separate functions, each returning a vector consisting of random values in the range $[0, \frac{\varphi_{max}}{2}]$, where φ_{max} is a positive constant. Point-wise vector multiplication of these functions with the difference between $\mathbf{y}_i$ and $\mathbf{x}_i$, as well as $\hat{\mathbf{y}}$ and $\mathbf{x}_i$ repectively, is indicated by $\otimes$.

Results showed that good performance were reported on some widely-used multimodal functions without the need to pre-specify niching parameters. However, the algorithm is computationally expensive due to the calculation of the fitness Euclidian-distance ratio for every pair of particles.

4.3.7 The waves of swarm particles algorithm

The waves of swarm particles (WoSP) algorithm, developed by Hendtlass [41], uses a technique that locates multiple optima by forcing the swarm to explore different promising regions of the problem space. Particles are organized into waves, each of which optimizes a single candidate solution. The practice of only evaluating a particle's performance at discrete intervals, is used to adjust particle behaviour in situations where the swarm is converging on an optimum.

To alter the behaviour of particles when they are settling close together, a short-range interaction between particles is introduced, namely a gravitational style attraction that becomes more effective when particles are near to one another. To acquire such an effect, the magnitude of the short-range force (SRF) of particle i towards particle j is set inversely proportional to some power p of the distance between the particles. This short-range force produces a velocity component, $\mathbf{v}_{i,j}$, that is represented by

$$\mathbf{v}_{i,j} = \frac{K}{d_{i,j}^{p}} \tag{4.21}$$

where $d_{i,j}$ is the distance between particles i and j, and K is a constant.

WoSP combines the short-range force with non-continuous or discrete evaluation to effect exploration of the search space. A particle's performance is evaluated at discrete time intervals. By the time of the next evaluation, particles may have passed each other and be at such a distance apart that the short-term attraction is too weak to bring them back together.

Because of the additional velocity component, $\mathbf{v}_{i,j}$, velocities decrease as the distance between particles increases. Particles will not be pulled back, but continue moving apart, exploring beyond their previous positions. Instead of converging on a single optimum, some of the neighbourhood particles are ejected with significant velocities, thus exploring other regions of the search space. Such particles are organized in a *wave*. In particles assigned to the wave, knowledge of the previous optimum is replaced by knowledge of the wave, that is, the best-fit particle of the wave becomes the neighbourhood best. Particles assigned to the wave are now attracted to the best particle in the wave. The wave converges on another optimum, starting the process again until all or most of the optima have been located.
Some aspects of the algorithm need to be clarified:

- Waves formed by ejected particles are numbered. Initially all particles belong to wave number zero. Every time a particle is ejected, it is promoted by having its wave number increased to that of the highest numbered wave. A new wave is created if necessary.

Particles are commonly ejected in pairs, and in that case a wave will have an initial size of two. To reduce the probability that particles are pulled back to the region they have left, a particle is required to move at least a user-specified distance (the *search scale*)away from the previous position before the particle may become part of a wave.

- A particle is considered as ejected when the ratio of the velocity component introduced by the short-range force to the other velocity components exceeds a user-specified value, called the *promotion factor*. As the speed of particles decrease when a wave is settling on an optimum, this ratio increases and the particle is ejected. Therefore particles are ejected when a wave is settling on an optimum.

The algorithm was tested on a number of benchmark problems. Results showed that the WoSP algorithm is able to escape from local sub-optima and continue to search for other optima.

4.4 Conclusion

This chapter described the concept of niching. Some niching strategies for genetic algorithms were reviewed, followed by a thorough discussion of a number of well-known niching algorithms developed in the field of particle swarm optimization. The algorithms are based on different principles, and various approaches are followed to locate all the solutions in functions with multiple optima.

Chapter 5 proposes the vector-based PSO, a new niching algorithm that purports to address a number of drawbacks of other niching algorithms.