Chapter 3

Niching Techniques

Niching techniques maintain multiple solutions in multimodal domains, in contrast to existing evolutionary and swarm intelligence optimization techniques that have been designed to only locate single solutions. This chapter introduces well-known GA-based niching techniques. The applicability of GA techniques to PSO is considered, and a number of niching applications are presented.

3.1 Introduction

Particle swarm optimizers have proven to be useful in locating optimal solutions to optimization problems. This fact is supported by almost all papers published in this research field. The PSO technique's effectiveness can be attributed to its efficient propagation of information regarding a single, *global* solution. The original update equations for velocity and position vectors were designed to lead all particles to the same solution [49].

When attempting to optimize multimodal functions, the situation is changed. In a multimodal function, multiple positions in the function's search space may have optimal fitness. The global best solution found by a swarm of particles when using the *gbest* or *lbest* algorithms may not necessarily be the *only* possible solution. Any particle that occupies a position close to a potential global solution, or even an acceptable local optimum, that is not close to the swarm's global best in the search space, will be forced to move towards the global best position $\hat{\mathbf{y}}$. Alternatively, equally acceptable solutions

that happened not to be close to the swarm's current best at any given time, are ignored in favor of a limited collective swarm consciousness. Consequently, portions of the search space are effectively ignored in favor of a potentially limited view of the search space.

A number of evolutionary techniques have been suggested to locate multiple solutions to multimodal problems. These algorithms have been almost exclusively explored using GAs. In GA parlance, optimization techniques that locate multiple optima to multimodal function optimization problems are known as *niching*, or *speciation* techniques.

For niching, both GAs and PSOs use a population of individuals that are partitioned in some way to focus and locate different possible solutions in a single search space (note that the term individual here applies both to individuals in GAs and particles in the PSO algorithm). Each subgroup in a partitioned population or swarm, is known as a *species*. The behavioral patterns of individuals competing for the use of a resource in a subgroup and between elements in a subgroup, is known as *speciation*.

This chapter explores the evolutionary theory behind niching and speciation. Section 3.2 presents a theoretical base for niching, followed by a number of existing GA-based niching techniques in section 3.3. Section 3.4 analyses a single known PSO attempt at niching. The simple application of GA niching techniques to PSOs is discussed in section 3.5, and the chapter is concluded with a number of niching applications in section 3.6.

3.2 What is Niching?

In an environment where a large number of individuals compete for the use of available resources, behavioral patterns emerge where individuals are organized into subgroups based on their resource requirements. Horn defines niching as a

> *"form of cooperation around finite, limited resources, resulting in the lack of competition* between *such areas, and causing the formation of* species *for each niche"* [37].

Niches are thus partitions of an environment, and species are partitions of a population competing within the environment. Localization of competition is introduced by simply *sharing* resources among individuals competing for it. The terms *niche* and *species* can

be used interchangeably. As an example, a school of fish that live in a certain part of the ocean compete with each other for access to a potentially limited food supply. Food may not be available everywhere in their environment. Certain fish may learn to live in a small area around a food source, while others may learn to roam their environment and only feed when they require nourishment. If there was to be a single food source, it is a reasonable expectation that all the fish would eventually exhibit similar behavior. They would all be required to find food in the same place, and encounter the same resistance from other fish.

The social interaction and adaptation of individuals competing in an environment around multiple resources form the basis for the study of niching techniques with evolutionary optimization algorithms. In the evolutionary optimization context, Horn defines *implicit niching* as the sharing of resources, and *explicit niching* as the sharing of fitness [37].

Niching methods can be categorized as either being *sequential* or *parallel*.

Sequential niching (or temporal niching) develops niches sequentially over time. The approach can be summarized as searching for a possible solution until it is found, then removing all references to it in the search space and repeating the search until convergence criteria are met. Because of the removal of 'confusing' optima, a technique that assumes a single global solution may be used.

Parallel niching forms and maintains several different niches simultaneously. The search space is not modified. *Parallel niching* techniques thus not only depend on finding a good measure to locate possible solutions, but also need to organize individuals in a way that maintains their organization in the search space over time to populate locations around solutions.

Regardless of the way in which niches are located (i.e. in parallel or sequentially), the distribution of individuals can be formalized in a number of ways, according to their *speciation* behavior [61]:

Sympatric speciation occurs when individuals form species that coexist in the same search space, but evolve to exploit different resources (or more formally, different ecological niches). For example, different kinds of fish feed of different food sources

in the same environment. Cannibalism is explicitly excluded here, although it may be an interesting measure to consider. It may act as a deterrent in overpopulated niches.

Allopatric speciation differentiates individuals based on spatial isolation in a search space. No interspecies communication takes place, and subspecies can develop only through deviation from the available 'genetic' information. Such an event could be triggered by mutation. Here, different fish species would effectively live and play around their food sources, and not be concerned with other species living in different areas.

Parapatric speciation allows new species to form as a result of segregated species sharing a common border. Communication between the initial species may not have been encouraged or intended. As an example, new fish species may evolve based on the interaction of a small percentage of different schools of fish. The new species may have different food requirements and may eventually upset the environment's stability.

The PSO nichers presented in this thesis can be classified as using an allopatric speciation approach. Allopatric speciation will therefore be a more prevalent issue of discussion, as it defines the goals of multimodal function optimization.

The next section discusses existing niching techniques that have been introduced in the genetic algorithm and particle swarm optimization fields.

3.3 Genetic Algorithm based Niching Techniques

This section presents a number of well known niching algorithms, originally studied using GAs. GA niching is based on research originally done to maintain diverse populations. Unless indicated, each of the techniques described next assumes that a normal generational evolutionary optimization process takes place, as discussed in section 2.4.

3.3.1 Fitness Sharing

Fitness sharing is one of the earliest GA niching techniques. It was originally introduced as a population diversity maintenance technique [32]. It is a parallel, explicit niching approach. The algorithm regards each niche as a finite resource, and shares this resource among all individuals in the niche. Individuals are encouraged to populate a particular area of the search space by adapting their fitness based on the number of other individuals that populate the same area. The fitness f_i of individual u is adapted to its shared fitness:

$$f_i' = \frac{f_i}{\sum_j sh(d_{u,v})} \tag{3.1}$$

A common sharing function is:

$$sh(d) = \begin{cases} 1 - (d/\sigma_{share})^\alpha & \text{if } d < \sigma_{share} \\ 0 & \text{otherwise.} \end{cases}$$

The symbol $d_{u,v}$ represents a distance calculated between individual u and individual v. The distance measure implemented can be genotypic or phenotypic, depending on the optimization problem at hand. If the sharing function finds that $d_{u,v}$ is less than σ_{share}, it returns a value in the range $[0, 1]$ that increases as $d_{u,v}$ decreases. Therefore, the more similar u and v, the lower their individual fitnesses will become. The fitness sharing approach assumes that the number of niches can be estimated, i.e. the approximate number of niches must be known prior to the application of the algorithm. It is also assumed that niches occur at least a minimum distance $2\sigma_{share}$ from each other. The above assumptions cannot always be made: The number of niches will not always be estimatable, nor would the distance between them.

3.3.2 Dynamic Niche Sharing

Miller and Shaw introduced dynamic niche sharing as a computationally less expensive version of fitness sharing [66]. The same assumptions are made as with fitness sharing:

- Niches must occur at a minimum distance of $2\sigma_{share}$ from each other,
- and the number of optima must be known.

During the evolution of a population with the dynamic niche sharing technique, individuals will invariably start to form subspecies and populate niches. Dynamic niche sharing attempts to *classify* individuals in a population as belonging to one of the emerging niches, or to a non-niche category. Fitness calculation for individuals belonging to the non-niche category is done with the same equation that is used in the original fitness sharing technique, namely equation (3.1), in section 3.3.1. The fitness of individuals found to belong to one of the developing niches is diluted by dividing it by the size of the developing niche. Dynamically finding niches is a simple process of iterating through the population of individuals and constructing a set of non-overlapping areas in the search space. Dynamic sharing is computationally less expensive than 'normal' sharing. Miller and Shaw presented results showing that dynamic sharing has improved performance when compared to fitness sharing [66].

3.3.3 Sequential Niching

Sequential niching (SN) is a simple algorithm introduced by Beasley *et al* [4]. SN identifies multiple solutions by adapting an optimization problem's objective function's fitness landscape through the application of a *derating* function at a position where a potential solution was found [4]. A derating function is designed to lower the fitness appeal of previously located solutions. By repeated applications of the algorithm to an objective function's fitness landscape, all confusing local and global optima are removed. Sample derating functions, for a point $\mathbf{x}$ and a previous maximum $\mathbf{x}^*$ include:

$$G_1(\mathbf{x}, \mathbf{x}^*) = \begin{cases} \left(\frac{\|\mathbf{x}-\mathbf{x}^*\|}{r}\right)^{\alpha} & \text{if } \|\mathbf{x}-\mathbf{x}^*\| < r \\ 1 & \text{otherwise} \end{cases}$$

and

$$G_2(\mathbf{x}, \mathbf{x}^*) = \begin{cases} e^{\log m \frac{r-\|\mathbf{x}-\mathbf{x}^*\|}{r}} & \text{if } \|\mathbf{x}-\mathbf{x}^*\| < r \\ 1 & \text{otherwise} \end{cases}$$

where r is the radius of the derating function's effect. In G_1, α determines whether the derating function is concave ($\alpha > 1$) or convex ($\alpha < 1$). For $\alpha = 1$, G_1 is a linear function. For G_2, m determines 'concavity'. Note that $\lim_{x \to 0} \log(x) = -\infty$, hence m must always be great than 0. Smaller values for m result in a more concave derating

function. The fitness function $f(\mathbf{x})$ is then redefined to be

$$M_{t+1}(\mathbf{x}) \equiv M_t(\mathbf{x}) \times G(\mathbf{x}, \mathbf{s}_t)$$

where $M_o(\mathbf{x}) \equiv f(\mathbf{x})$ and $\mathbf{s}_t$ is the best individual found during run t of the algorithm. G can be any derating function, such as G_1 and G_2.

3.3.4 Crowding

Crowding, or the crowding factor model, as introduced by De Jong [19], was originally devised as a diversity preservation technique. Crowding was inspired by a naturally occurring phenomenon in ecologies, namely competition amongst similar individuals for limited resources. Similar individuals compete to occupy the same ecological niche, while dissimilar individuals do not compete, as they do not occupy the same ecological niche. When a niche has reached its *carrying capacity* (i.e. being occupied by the maximum number of individuals that can exist within it) older individuals are replaced by newer (younger) individuals. The carrying capacity of the niche does not change, so population size remains constant.

For a genetic algorithm, crowding is performed as follows: It is assumed that a population of GA individuals evolve over several generational steps. At each step, the crowding algorithm selects only a portion of the current generation to reproduce. The selection strategy is fitness proportionate, i.e. more fit individuals are more likely to be chosen. After the selected individuals have reproduced, individuals in the current population are replaced by their offspring. For each offspring, a random sample is taken from the current generation, and the most similar individual is replaced by the offspring individual. To cover a complete search space, the initial position of individuals should be well distributed, as the algorithm is unlikely to evaluate any part of the search space that is not within the first generation.

3.3.5 Deterministic Crowding

Deterministic crowding (DC) is based on De Jong's crowding technique (see section 3.3.4), but uses the following improvements as suggested by Mahfoud [61]:

- Mahfoud found that phenotypic similarity metrics (such as Euclidean distance) were preferred to similarity metrics based on genotypes (e.g. Hamming distance). Phenotypic metrics embody domain specific knowledge that is most useful in multimodal optimization, as several different spatial positions can contain equally optimal solutions. Not only is the quality of potential solutions important, but also their proximity to each other.

- It was shown that there exists a high probability that the most similar individuals to offspring are their parents. The replacement strategy initially proposed by De Jong was changed to compare an offspring only to its parents and not to a random sampling of the population.

- De Jong's crowding used a traditional proportional method. Individuals are selected for reproduction based on their fitness. Mahfoud suggested selecting individuals randomly, and only replacing parents with their offspring if the offspring performs better.

Since the DC algorithm is used in later chapters, it is presented in figure (3.1) (algorithm pseudo-code taken from [61], symbols as defined in table 2.1). Note the $d(\cdot)$ is a phenotypic distance function.

Probabilistic crowding, described as an *"offspring of deterministic crowding,"* was introduced by Mengshoel *et al* [65]. It is based on Mahfoud's deterministic crowding, but employs a probabilistic replacement strategy.

Where the original crowding and DC techniques replaced an individual u with v if v was more fit than u, probabilistic crowding uses the following rule: If individuals u and v are competing against each other, the probability of u winning and replacing v is given by

$$p_u = \frac{f_u}{f_u + f_v}$$

where f_u is the fitness of individual u. The core of the algorithm is therefore to use a probabilistic tournament replacement strategy. Experimental results have shown it to be both fast and effective.

Repeat for g generations:

1. Do $n/2$ times:

   (a) Select 2 parents, p_1 and p_2, randomly from C_g.

   (b) Cross p_1 and p_2, yielding c_1 and c_2.

   (c) Apply mutation/other operators, yielding $c_1^{'}$ and $c_2^{'}$.

   (d) IF $[d(p_1, c_1^{'}) + d(p_2, c_2^{'})] \leq [d(p_1, c_2^{'}) + d(p_2, c_1^{'})]$

   - If $f(c_1^{'}) > f(p_1)$ replace p_1 with $c_1^{'}$
   - If $f(c_2^{'}) > f(p_2)$ replace p_2 with $c_2^{'}$

   ELSE

   - If $f(c_2^{'}) > f(p_1)$ replace p_1 with $c_2^{'}$
   - If $f(c_1^{'}) > f(p_2)$ replace p_2 with $c_1^{'}$

Figure 3.1: Deterministic Crowding Algorithm

3.3.6 Restricted Tournament Selection

Restricted Tournament Selection (RTS), introduced by Harik [35], is similar to DC, but promotes local competition.

In RTS, the selection process is adapted in the following way:

Two individuals, u and v are randomly selected from the pool of individuals in the current generation.

Crossover and mutation operators are performed, yielding two new individuals, u^* and v^*.

The remainder of the current population is searched for individuals that are the most similar to u^* and v^*, and when found, are designated by s_u and s_v.

u^* then competes against s_u for a position in the next generation. The same happens with v^* and s_v.

Harik presented results in [35] proving that RTS successfully locates solutions to multi-modal problems.

3.3.7 Coevolutionary Shared Niching

Goldberg and Wang introduced coevolutionary shared niching (CSN) [33]. CSN locates niches by co-evolving two different populations of individuals in the same search space, in parallel. Let the two parallel populations be designated by A and B, respectively. Population A can be thought of as a normal population of candidate solutions, and it evolves as a normal population of individuals. Individuals in population B are scattered throughout the search space. Each individual in population A associates with itself a member of B that lies the closest to it using a genotypic metric. The fitness calculation of the i^{th} individual in population A, A_i, is then adapted to $f'(A_i) = \frac{f(A_i)}{|B_b|}$, where $f(\cdot)$ is the fitness function; $|B_b|$ designates the cardinality of the set of individuals associated with individual B_b and b is the index of the closest individual in population B to individual i in population A. The fitness of individuals in population B is simply the average fitness of all the individuals associated with it in population A, multiplied by B_b. Goldberg and Wang also developed the *imprint CSN* technique, that allows for the transfer of good performing individuals from the A to the B population.

CSN overcomes the limitation imposed by fixed inter-niche distances assumed in the original fitness sharing algorithm [32] and its derivate, dynamic fitness sharing [66]. The concept of a niche radius is replaced by the association made between individuals from the different populations.

3.3.8 Dynamic Niche Clustering

Dynamic Niche Clustering (DNC) is a fitness sharing based, cluster driven niching technique [28, 29]. It is distinguished from all other niching techniques by the fact that it supports 'fuzzy' clusters, i.e. clusters may overlap. This property allows the algorithm to distinguish between different peaks in a multimodal function that may lie extremely close together. In most other niching techniques, a more general inter-niche radius (such as the σ_{share} parameter in fitness sharing) would prohibit this.

The algorithm works by constructing a *nicheset*, which is a list of niches in a population. The nicheset persists over multiple generations. Initially, each individual in a population is regarded to be in its own niche. Similar niches are identified using Euclidean distance and merged. The population of individuals is then evolved over a pre-determined number of generational steps. Before selection takes place, the following process occurs:

- The midpoint of each niche in the nicheset is updated, using the formula

$$\overline{mid_u} = \overline{mid_u} + \frac{\sum_{i=1}^{n_u}(\mathbf{x}_i - \overline{mid_u}) \cdot f_i}{\sum_{i=1}^{n_u} f_i}$$

  where $\overline{mid_u}$ is the midpoint of niche u, initially set to be equal to the position of the individual from which it was constructed, as described above. n_u is the niche count, or the number of individuals in the niche, f_i is the fitness of individual i in niche u and $\mathbf{x}_i$ is the location of individual i.

- A list of inter-niche distances is calculated and sorted. Niches are then merged using a technique described in [29].

- Similar niches are merged. Each niche is associated with a minimum and maximum niche radius. If the midpoints of two niches lie within the minimum radii of each other, they are merged.

- If any niche has a population size greater than 10% of the total population, random checks are done on the niche population to ensure that all individuals are focusing on the same optima. If this is not the case, such a niche may be split into sub-niches, which will be optimized individually in further generational steps.

Using the above technique, Gan and Warwick also suggested a *niche linkage* extension to model niches of arbitrary shape [30].

3.4 PSO Niching Techniques

While research on GA niching techniques is abundant, niching with PSOs have thus far received little research attention. This section overviews a niching variation of the

objective function stretching optimization technique, as discussed in section 3.4. This approach is, to the author's knowledge, the only existing approach applicable to PSO niching.

Objective Function Stretching for Locating Multiple Global Minima

Objective function stretching, introduced in section 2.7.1, was applied as a sequential niching technique by Parsopoulos and Vrahatis [72]. The technique is partially repeated here for completeness and discussed in more detail to clarify its niching ability.

The stretching technique adapts the landscape of an optimization problem's fitness function to remove local minima. When a local solution is detected during the evolutionary learning process, the *stretching* operator is applied to remove the detected solution from the fitness landscape. Subsequent iterations of the PSO algorithm can then focus on locating solutions in other parts of the search space, assured that the detected local optima will not again lead to premature convergence.

The niching variation of the stretching technique detects potential solutions by comparing candidate solutions to a threshold value ϵ. Parsopoulos and Vrahatis suggest values such as 0.01 and 0.001 [72]. (Note that it was implicitly assumed that optimization problems were of low dimension. Their test results were only given on one and two-dimensional problems. Higher dimensional problems will most likely require larger values.) When a potential solution $\mathbf{x}^*$ is detected with this technique, the particle at $\mathbf{x}^*$ is *isolated* from the rest of the swarm. The stretching operator is then applied at $\mathbf{x}^*$, marking this position and its vicinity as undesirable by increasing the fitness. The "application of the stretching functions" means that the fitness calculation of remaining particles are adapted. If the position vector of the detected solution is given by $\mathbf{x}^*$, the fitness calculation, $f(\mathbf{x})$, for all remaining particle positions $\mathbf{x}$, is redefined to be $H(\mathbf{x})$, where:

$$H(\mathbf{x}) = G(\mathbf{x}) + \gamma_2 \frac{\text{sign}(f(\mathbf{x}) - f(\mathbf{x}^*)) + 1)}{2\tanh(\mu(G(\mathbf{x}) - G(\mathbf{x}*)))} \tag{3.2}$$

and

$$G(\mathbf{x}) = f(\mathbf{x}) + \gamma_1 \frac{\|\mathbf{x} - \mathbf{x}^*\|(\text{sign}(f(\mathbf{x}) - f(\mathbf{x}^*)) + 1)}{2} \tag{3.3}$$

The interpretation of the sign function is the same as in section 2.7.1. The transformation represented by $G(\mathbf{x})$ in equation (3.3) removes all local minima located above the

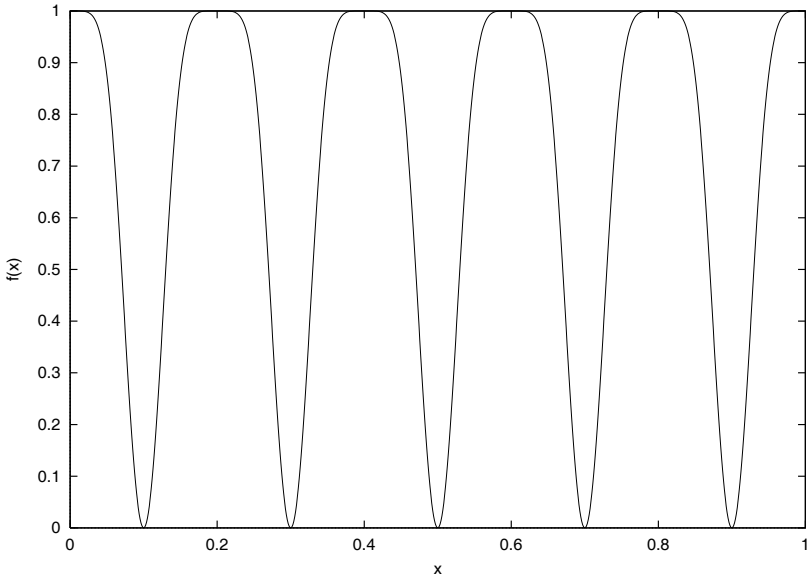

Figure 3.2: Function $F_s(x) = 1.0 - \sin^6(5\pi x)$ with 5 minima of equal fitness.

detected solution $\mathbf{x}^*$. The transformation in equation (3.2) assigns higher objective function values to positions close to $\mathbf{x}^*$. The objective function landscape below $\mathbf{x}^*$ remains unchanged. Local optima with worse fitness than at position $\mathbf{x}^*$ is thus removed from the search space.

Van den Bergh investigated the efficacy of objective function stretching as global optimization technique. It was found that the technique may alter the search space by introducing false minima [87]. This observation in part warrants a more thorough investigation of the applicability of stretching as a niching technique. The alteration of the search space by the stretching operator for niching purposes is discussed next.

In this thesis the objective function stretching technique was applied to locate all the minima of the function

$$F_s(x) = 1.0 - \sin^6(5\pi x) \tag{3.4}$$

in the domain $x \in [0, 1]$ (see figure (3.2)). The objective function f is defined to be $f(x) = F_s(x)$. A PSO was trained on the objective function f, with parameter settings $\gamma_1 = 10^4$, $\gamma_2 = 1.0$, $\mu = 10^{-10}$ and $\epsilon = 10^{-5}$. When a minimum was detected at a particle position x when $f(x) < \epsilon$, the particle was isolated and the fitness function redefined to $f(x) = H(x)$. When $x \approx 0.9$, the fitness function landscape was altered, as is shown in

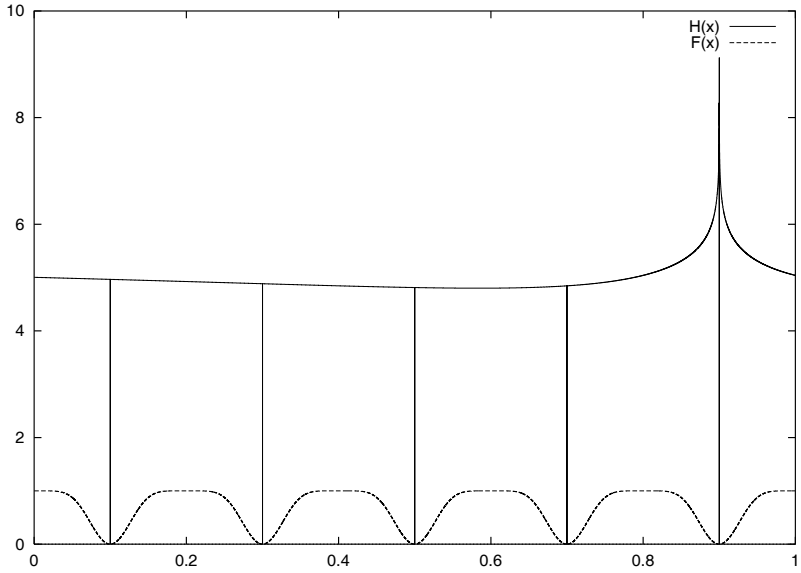

Figure 3.3: The modified objective function landscape, after the stretching functions were applied.

figure (3.3). The effect of the respective transformation functions, $H(x)$ and $G(x)$ can be pertinently seen. The effect of $G(x)$, i.e. removing all minima above the position indicated by x, clearly lifts out the positions of the remaining minima. $H(x)$ ensures that the fitness landscape around the potential solution x is marked as undesirable. The following problems are however introduced:

- If multiple acceptable minima are located close to each other, the effect of $G(x)$ may cause these alternative solutions never to be detected. The steep fitness function slope, regardless of the 'trenches' on remaining minima (see below), will keep particles from traversing towards this area of the search space. Also, if x is removed from the swarm, $\hat{\mathbf{y}}$ will be redefined to a different location that would be likely to discourage movement towards the position of x.

- The adaptation of $f(x)$ to remove all potential minima above x, introduces 'trenches' in the fitness function around remaining solutions. Although this transformation makes detecting these remaining minima visually simple, the optimization process is less likely to detect them. If $\epsilon = 10^{-5}$, as was given above, a solution

will be detected around $x \approx 0.9$ when any particle has a position in the range $[0.89999, 0.90001]$. The width of the 'trench' around 0.9 is then 0.00002. The probability of the evolutionary search process locating this position can then be calculated by dividing the width of the trench by the total width of the search space. This yields an extremely small probability of locating the minima under consideration and other minima.

- The transformation of $f(x)$ with the given values for γ_1, γ_2 and μ, introduced a new local minimum at $x \approx 0.6$. Because of the small likelihood of locating the actual minima, the optimization process is more likely to regard this position as a minimum. Tuning parameters γ_1, γ_2 and μ may remove the introduced minimum.

The above issues question the usability of objective function stretching as an effective niching technique. The results reported by Parsopoulos and Vrahatis in [72] could not be replicated.

3.5 Application of GA Niching Techniques to PSO

Given the wealth of GA based niching techniques, a natural step would be to consider the adaptation of these techniques to particle swarm optimizers. This section discusses this possibility.

By default, the PSO algorithm uses a phenotypic similarity metric in the form of a fitness function. In unimodal optimization, this approach is acceptable, since the assumption can be made that when two particles exhibit similar fitness, they are definitely approaching the same solution. The particles will also occupy similar positions in the search space. In multimodal optimization, the fitness function is still crucial in the assessment of the *quality* of solutions found by particles, but it is not capable of giving an indication of particle *similarity*, based on phenotypic behavior. Niches with similar fitness may occur in different positions in the search space. A metric such as Euclidean distance can give an acceptable indication of particle similarity, but it is not capable of doing this while considering particle quality.

GA inspired crowding techniques promote the formation of niches by maintaining sets of similar individuals. Similarity between individuals in different generations is a result

of the replacement strategy used. By maintaining individuals that are similar, multiple niche locations can be identified and maintained over several generations. Maintaining particles around a potential solution in the PSO algorithm is done quite differently. Particles are not linked over different generations or iterations of the algorithm in the same way. With PSOs, this explicit maintenance of locations around a potential solution is largely taken over by the cognition memory of its personal best solution. The PSO algorithm's rapid global search nature is achieved by the propagation of knowledge regarding global good solutions and each particle's ability to remember its personal best solution. Sharing information about a single solution focuses all search efforts on the best solution found by the swarm at any given time during the optimization process.

Removing the global best information yields particles that perform a local search, biased by the best solution found by each. This type of particle is somewhat similar to individuals in GA populations. All GA techniques that depend on the independent nature of individuals, as described above, cannot simply be applied to PSOs. Attempting to remove all references to global information effectively truncates the value of the PSO search to that of a random search evolutionary program.

Because of this fact, two new niching techniques, suited specifically to the nature of the PSO, are presented in the next chapters.

3.6 Application of Niching Techniques to Real-World Problems

Several areas, where the location of multiple solutions in a search space are beneficial, can be identified. This section gives a short overview of a number of well-known techniques.

Carroll investigated the application of a multitude of different GA techniques to the optimization of chemical oxygen-iodine lasers [10]. Chemical lasers are produced through a series of chemical reactions between gases. In particular, Carroll compared Goldberg's sharing technique [32] with a non-sharing GA. He found that sharing helped to more rapidly find an optimal power input. The interested reader is referred to [10] for a more in-depth treatment of this particular application.

Hwang and Cho successfully applied the fitness sharing technique to evolve diverse

circuit architectures [43]. Fitness sharing allowed them to design an embedded device that dynamically reconfigures its circuit architecture when necessary. The system can generate multiple architectures concurrently.

Kim and Cho used deterministic crowding (see section 3.3.5) to evolve a checkers player [53]. Their system evolved a neural network to play the game. They found that by selecting multiple neural networks from different optimal solutions found by the different niches identified by DC, game-play was improved.

To evaluate the effectiveness of the proposed PSO based niching techniques, multimodal function optimization problems are considered, as well as solving systems of unconstrained equations with multiple solutions.

3.7 Conclusion

Niching is an optimization technique inspired by natural evolution. Niching algorithms allow traditionally unimodal optimization techniques, such as GAs and PSOs, to be extended to locate multiple solutions in a search space. This chapter reviewed well-known GA-based niching techniques, as well as more recent attempts, including a PSO based algorithm. The possibility of extending GA niching techniques to PSOs was investigated. It was found that the techniques inspired by the generational model of GAs are not easily extended to the particle swarm model. The PSO model is based on a set of different assumptions and particles in a swarm are more free to traverse the search space, than individuals in a GA population. Finally, a number of existing real-world niching applications were presented.

In the next chapters, niching algorithms, designed to make use of the specifics of the PSO model, are presented.