Chapter 5

NichePSO, a Multi-Swarm Optimizer

This chapter presents a PSO technique that solves multimodal optimization problems with the concurrent optimization power of multiple swarms. The technique, NichePSO, extends the inherent unimodal nature of the standard PSO approach by growing multiple swarms from an initial particle population. The initial particle swarm is split into smaller swarms as niches are detected. Upon termination of the algorithm, each subswarm represents one of the potential solutions to the problem. Experimental results show that NichePSO can successfully locate all optima on a set of test functions. The influence of control parameters, including the relationship between the swarm size and the number of solutions (niches), as well as the scalability of the algorithm is investigated.

5.1 Introduction

This chapter presents the *niching particle swarm optimization* algorithm, NichePSO. NichePSO is aimed at locating multiple solutions to multimodal problems through the use of multiple, independent subswarms. The *nbest* optimizer presented in the previous chapter sports a drawback: It cannot properly maintain local optima. Overlap in *nbest* particle neighborhoods forces the algorithm to always prefer solutions that have better fitness. Consequently, when a local optimum occurs relatively close to another, better

optimum, the definition of a neighborhood will always prefer better solutions. The NichePSO algorithm overcomes this limitation through the use of multiple subswarms.

The use of subswarms, or *subpopulations*, as part of a population based optimization algorithm, is not a new idea. It has been applied in the GA optimization field, both as a

- diversity improvement technique [6], and as
- a premature convergence avoidance technique [84].

For the purposes of this thesis, subpopulations/subswarms imply

> A *bona-fide* segmentation of a large population of individuals/particles into smaller groupings. Each subswarm can function as a stable, individual swarm entity, evolving on its own, independent of individuals in other swarms.

The use of subswarms has been adopted early on in the development of the PSO. Løvbjerg *et al* introduced a diversity improvement technique that partitions a particle swarm into a number of different subpopulations [60]. Each subpopulation is responsible for maintaining its own best known, or *gbest*, solution. A crossover operator is used to share information about global solutions (see section 2.4.3). The crossover operator may be applied to particles from

- a single swarm, or
- particles originating from different swarms.

The algorithm selects particles on which crossover is to be performed randomly. Performing crossover between particles from the same swarm leads to better solutions and consequently faster convergence *within* a particular swarm. Crossover between particles from different swarms facilitates inter-population communication that eventually leads to a single *global* solution. Inter-swarm sharing of information is not conducive to the formation of subpopulations around different potential solutions. All swarms will eventually gravitate towards a single solution. Performing crossover in the same swarm to maintain a diverse 'local' record of good solutions reminds strongly of the goals of GA crowding techniques. Crowding techniques replace individuals in a population with similar individuals in a next generation (see section 3.3.4). It should however be noted that

the goal of the research presented by Løvbjerg *et al* was not to maintain multiple solutions and to perform niching, but to improve the quality of a global solution. Typically, this approach will be most useful in a deceptive problem domain, where particles may become trapped in suboptimal solutions. By performing crossover on randomly selected particles, particles fooled by a suboptimal solution can be moved closer to a better, or global best, solution.

In the rest of this chapter, the NichePSO algorithm is presented and motivated. A number of newly introduced niching parameters are analyzed and empirical results are presented that motivate the validity of NichePSO as a niching technique.

5.2 The Niching Particle Swarm Optimization Algorithm

The *nbest* optimizer was initially developed as a technique to find and maintain multiple points of intersection in systems of equations (SEs). Based on the reformulation of the fitness function presented in the previous chapter, points of intersection in a SEs will always have *equal*, *optimal* fitness in a search space. When local optima, i.e. points of suboptimal fitness exist, the *nbest* algorithm's neighborhood definition will keep it from locating these solutions. Although the neighborhood formulation introduces a bias towards a local optimum in a search space in the velocity update equation, neighborhoods for individual particles may still overlap. Consequently, if a suboptimal, local solution exists close to a solution that may yield a higher fitness, the neighborhood update will lead to an update, biased towards the better solution (see figure 5.1). NichePSO, through the use of subswarms and two control parameters, δ and μ, overcomes the shortcomings of the *nbest* algorithm.

NichePSO starts by uniformly distributing particles throughout the search space of an optimization problem. The initial swarm of particles is referred to as the *main* swarm. As particles traverse the search space, they invariably move towards positions that have attractive fitness. A potential solution is identified by monitoring the change in a particle's fitness over a number of training iterations. When such a solution is identified, a new subswarm is created by removing from the main swarm the particle

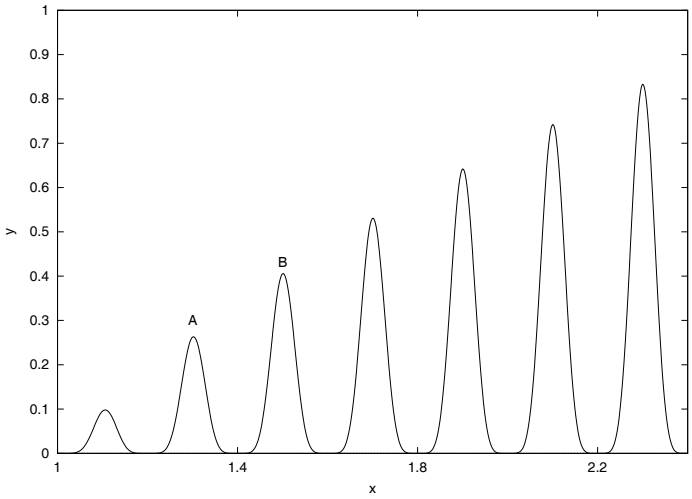


Figure 5.1: $f(x) = \ln(x)\sin^6(5\pi x)$. Note that the function consists of multiple rising peaks located close to each other. A particle close to peak A may be influenced to rather move to peak B based on the influence of its neighborhood.

that detected the potential solution and creating a subswarm from it. The main swarm thus shrinks as subswarms are grown from it. The algorithm is considered to have converged when subswarms no longer improve on the solutions that they represent. The NichePSO algorithm is summarized in figure 5.2.

In the following sections, each step of the algorithm is discussed in detail.

5.2.1 Initialization

The general location of potential solutions in a search space may not always be known in advance. Therefore, it is a good policy to distribute particles uniformly throughout the search space before learning commences. To ensure a uniform distribution, NichePSO uses *Faure*-sequences to generate initial particle positions. An efficient way of calculating Faure-sequences is given in [86]. Other pseudo-random uniform number generators, such as Sobol-sequences [77], may also be used.

1. Initialize the main particle swarm.
2. Train the main swarm particles using one iteration of the *cognition only* model.
3. Update the fitness of each main swarm particle.
4. For each subswarm:
   (a) Train subswarm particles using one iteration of the GCPSO algorithm.
   (b) Update each particle's fitness.
   (c) Update swarm radius
5. If possible, merge subswarms
6. Allow subswarms to absorb any particles from the main swarm that moved into it.
7. Search the main swarm for any particle that meets the partitioning criteria. If any is found, create a new subswarm with this particle and its closest neighbor.
8. Repeat from 2 until stopping criteria are met.

Figure 5.2: NichePSO Algorithm

5.2.2 Main Swarm Training

In the *nbest* algorithm, overlapping particle neighborhoods discourage convergence on local optima, such as the ascending maxima shown in figure 5.1. To this end, NichePSO uses a technique that frees a particle from the influence of a neighborhood or global best term in the velocity update equation. When a particle considers only its own 'history and experiences', in the form of a personal best, it can convergence on an optimum that does not have global optimal fitness, as it is not drawn to a position in the search space that has better fitness as a result of the traversal of another particle. This search approach has been previously investigated by Kennedy [47]. It was given in equation (2.20), repeated here for clarity:

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_{1,j}(t)(y_{i,j}(t) - x_{i,j}(t)) \tag{5.1}$$

Kennedy referred to update equation (5.1) as the *cognition only* model, in recognition of the fact that only a conscience factor, in the form of the personal best $\mathbf{y}_i$, is used in the update. No social information, such as the *global best* solution in the *gbest* and *lbest* algorithms, will influence position updates. This arrangement allows each particle to perform a local search.

5.2.3 Identification of Niches

A fundamental question when searching for different niches, is how to identify them. The niching algorithm proposed by Parsopoulos *et al* (see section 3.4) uses a threshold value ϵ, such that when a particle i's fitness at a position $\mathbf{x}_i$ becomes *less* than the threshold, i.e. when

$$f(\mathbf{x}_i) < \epsilon$$

for a minimization problem, the particle is removed from the swarm and labelled as a potential global solution. Immediately thereafter, the objective function's landscape is stretched to avoid additional and unnecessary exploration of this area surrounding the discovered solution. If the isolated particle's fitness is not close to a desired level, the solution can be refined by searching the surrounding function landscape with the addition of more particles. This approach proved to be effective when considering Parsopoulos *et al*'s results. The threshold parameter ϵ is however subject to fine tuning,

and locating good solutions depends strongly on the objective function's landscape and dimensionality.

To avoid the use of this tunable parameter, NichePSO uses a similar approach that monitors changes in the fitness of a particle. If a particle's fitness shows little change over a number of iterations of the learning algorithm, a *subswarm* is created with the particle and its closest topological neighbor. More formally, the standard deviation in particle i's fitness, σ_i, is tracked over a number of iterations, e_σ, where e_σ was set to 3 in the experiments conducted in section 5.3. When $\sigma_i < \delta$, a subswarm may be created with particle i and its closest neighbor. To avoid problem dependence, σ_i is normalized according to the range of the search space, commonly referred to as x_{min} and x_{max} in PSO literature. This approach can find *local* minima, for which $\sigma_i < \delta$ holds. If local minima are undesired, the fitness of a particle can be compared to a threshold to ensure that the solution meets a minimum fitness criterion.

The 'closest neighbor' to particle i's position $\mathbf{x}_i$ is simply the particle c with position $\mathbf{x}_c$, where

$$c = \arg\min_j \{\|\mathbf{x}_i - \mathbf{x}_j\|\}$$

with $1 \leq i, j \leq s$, $i \neq j$ and s is the size of the main swarm. Subswarms are optimized independent from the main swarm, in the same search space. The following sections present measures that are put into place to ensure that search efforts are not duplicated on the same solutions.

5.2.4 Absorption of Particles into a Subswarm

When a particle is still a member of the main swarm, it has no knowledge of subswarms that may have been created during the execution of the NichePSO learning algorithm. It is therefore quite likely that a particle may venture into an area of the search space that is being independently optimized by a subswarm. Such particles are merged with the corresponding subswarm, based on the following suppositions:

- Inclusion of a particle that traverses the search space of an existing subswarm may expand the diversity of the subswarm, thereby more rapidly leading to solutions with better fitness.

- An individual particle moving towards a solution on which a subswarm is working, will make much slower progress than what would have been the case had social information been available to ensure that position updates move towards the particle's *known* favorable solution.

To facilitate merging, particles are absorbed into a subswarm when they move 'into' the subswarm. That is, a particle i will be absorbed into a subswarm S_j when

$$\|\mathbf{x}_i - \hat{\mathbf{y}}_{S_j}\| \leq R_j \tag{5.2}$$

where R_j signifies the radius of subswarm S_j, and is defined as

$$R_j = \max\{\|\hat{\mathbf{y}}_{S_j} - \mathbf{x}_{S_{j,i}}\|\} \tag{5.3}$$

$\mathbf{x}_{S_{j,i}}$ represents all particles in S_j subject to $i \neq g$, $\hat{\mathbf{y}}_{S_j}$ represents the global best particle in S_j. Generally, subswarms have small radii, due to the homogeneous nature of the positions represented by their particles. Therefore, when a particle i moves into the hyper-sphere defined by a subswarm's global best particle and radius, it is unlikely that it would move away from the possible solution maintained by the subswarm. If the absorption step was absent from the algorithm, i will first have to be considered for a subswarm and successfully made part of one, before it can merge with S_j. If no other particles occur in the same portion of the search space, a subswarm containing i will never be created, and the potential solution it represents will never be considered. If i is merged with a particle in a similar situation, but that occurs in a vastly different position in the search space, the algorithm's convergence would be impaired.

5.2.5 Merging Subswarms

A subswarm is created by removing a particle that represents an acceptable candidate solution from the main swarm, as well as a particle that lies closest to it in the search space, and to group these into a subswarm. From this rule, it follows that particles in subswarms all represent similar solutions. This can lead to subswarms with radii that are very small, and even radii approximating zero. Consequently, when a particle approaches a potential solution, it may not necessarily be absorbed into a subswarm that

is already optimizing the particular solution. If the particle has an acceptable fitness, another subswarm will be created on its position in the search space. If two solutions are very similar, a single subswarm will be created to optimize both solutions. Eventually, only one of these solutions will be found. This introduces a dilemma, as multiple swarms will attempt to optimize the same solution. To alleviate this, subswarms may be merged when the hyper-space defined by their particle positions and radii intersect in the search space. When swarms are merged, the newly created swarm benefits from the extensive social information present in the parent swarms. Accordingly, superfluous local traversal of the search space is avoided. Formally, two subswarms S_{j1} and S_{j2} *intersect*, when

$$\|\hat{\mathbf{y}}_{S_{j1}} - \hat{\mathbf{y}}_{S_{j2}}\| < (R_{j1} + R_{j2}) \tag{5.4}$$

When $R_j = 0$ holds for subswarm S_j, all particles in S_j represent the same candidate solution. If this condition holds for both swarms under consideration, equation (5.4) fails to detect the presence of multiple subswarms in the same niche. Consequently, when two swarms, S_{j1} and S_{j2} do not satisfy equation (5.4), because $R_{j1} = R_{j2} = 0$[1], they can be merged when

$$\|\hat{\mathbf{y}}_{S_{j1}} - \hat{\mathbf{y}}_{S_{j2}}\| < \mu \tag{5.5}$$

As with δ, μ can be an appreciably small number, such as 10^{-3}, to ensure that two swarms are sufficiently similar. To avoid having to tune μ over the range of the search space under consideration, $\|\hat{\mathbf{y}}_{S_{j1}} - \hat{\mathbf{y}}_{S_{j2}}\|$ is normalized to the interval $[0, 1]$. S_{j1} and S_{j2} are merged by creating a new subswarm consisting of all S_{j1} and S_{j2}'s particles. The influence of different values of μ, and an upper bound on it, are discussed in section 5.4.1.

5.2.6 The GCPSO Algorithm

The GCPSO algorithm was presented and discussed in section 2.7.1. The subswarm creation technique presented in section 5.2.5 always yields swarms that initially consist of two particles. Training such a small swarm with the *gbest* algorithm, especially when

[1]Since position updates in PSO is a stochastic process, it is practically safer to consider the situation where $R_{j1} \approx 0$ and $R_{j2} \approx 0$.

its particles are topologically highly similar, may lead to swarm stagnation, forcing the subswarm to convergence on a suboptimal solution. GCPSO puts measures in place that ensure that a swarm does not stagnate. For a definition of what is meant by swarm stagnation, as well as a rigorous analysis of why GCPSO is necessary, the reader is referred to section 5.4.3.

5.2.7 Stopping Criteria

When each individual subswarm has located a solution and stably maintained it for a number of training iterations, the NichePSO may be considered to have converged. The following stopping criteria are implemented:

- Each swarm must converge on a unique solution. Typically, a subswarm is considered to have converged when its global best solution's fitness is either above or below a threshold value, depending on whether the fitness function describes a maximization or minimization problem. Fitness threshold criteria cannot however detect acceptable solutions in a multimodal fitness function where local *and* global maxima exist. Local maxima are never considered to be acceptable solutions, as their fitness do not necessarily adhere to possibly strict threshold values. Any algorithm that therefore depends solely on threshold values will fail to converge. Therefore, the change in particle positions are tracked over a number of iterations. If discernible change occurs in their positions, such as may be detected by considering their variance over a small number of training iterations, the subswarm may be considered to have converged.
- The algorithm is stopped after a maximum number of training iterations.

5.3 Experimental Results

This section presents experimental results obtained on a set of well-known multimodal functions. These functions have been extensively used in the testing of a number of GA niching techniques [4, 32, 37, 61]. Test functions are defined in section 5.3.1, and experimental results and a discussion thereof follows in section 5.3.2.

5.3.1 Test Functions

NichePSO is tested on a number of multimodal functions, where the goal is to identify all optima. These functions were originally introduced by Goldberg and Richardson to test fitness sharing [32] and have also been used by Beasley *et al* to evaluate their sequential niching algorithm [4]. Figure 5.3 illustrates functions F1 to F4, defined as:

$$F1(x) = \sin^6(5\pi x) \tag{5.6}$$

$$F2(x) = \left(e^{-2\log(2)\times\left(\frac{x-0.1}{0.8}\right)^2}\right) \times \sin^6(5\pi x) \tag{5.7}$$

$$F3(x) = \sin^6(5\pi(x^{3/4} - 0.05)) \tag{5.8}$$

$$F4(x) = \left(e^{-2\log(2)\times\left(\frac{x-0.08}{0.854}\right)^2}\right) \times \sin^6(5\pi(x^{3/4} - 0.05)) \tag{5.9}$$

Functions $F1$ and $F3$ both have 5 maxima with a function value of 1.0. In $F1$, maxima are evenly spaced, while in $F3$ maxima are unevenly spaced. In $F2$ and $F4$, local and global peaks exist at the same x-positions as in $F1$ and $F3$, but their fitness magnitudes decrease exponentially. Functions $F1$ to $F4$ are investigated in the range $x \in [0, 1]$. For each of the functions, maxima occur at the following x positions:

$F1$	0.1	0.3	0.5	0.7	0.9
$F2$	0.1	0.3	0.5	0.7	0.9
$F3$	0.08	0.25	0.45	0.68	0.93
$F4$	0.08	0.25	0.45	0.68	0.93

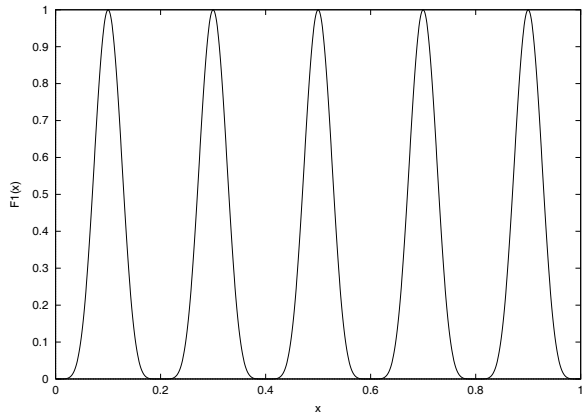
Function $F5$, the modified Himmelblau function (see figure 5.4), is defined as

$$F5(x, y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2 \tag{5.10}$$

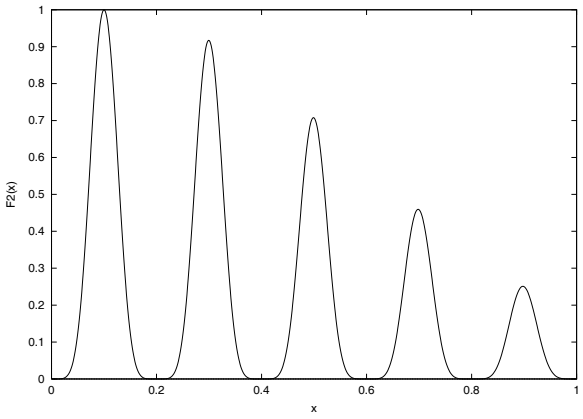
$F5$ has 4 equal maxima with $F5(x, y) = 200$. Maxima are located at $(-2.81, 3.13)$, $(3.0, 2.0)$, $(3.58, -1.85)$ and $(-3.78, -3.28)$.

5.3.2 Results

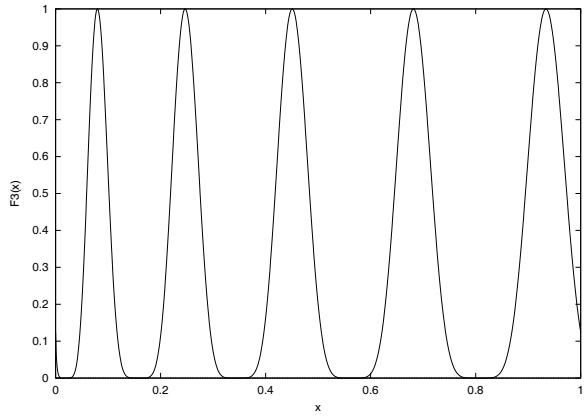
For each of the 5 test functions, 30 simulations were done with the NichePSO algorithm with $c_1 = c_2 = 1.2$. The inertia weight w was scaled linearly from 0.7 to 0.1 over a



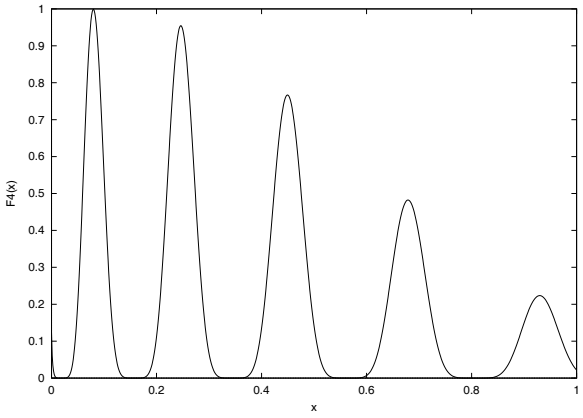
(a) Function F1



(b) Function F2



(c) Function F3



(d) Function F4

Figure 5.3: NichePSO Test Functions

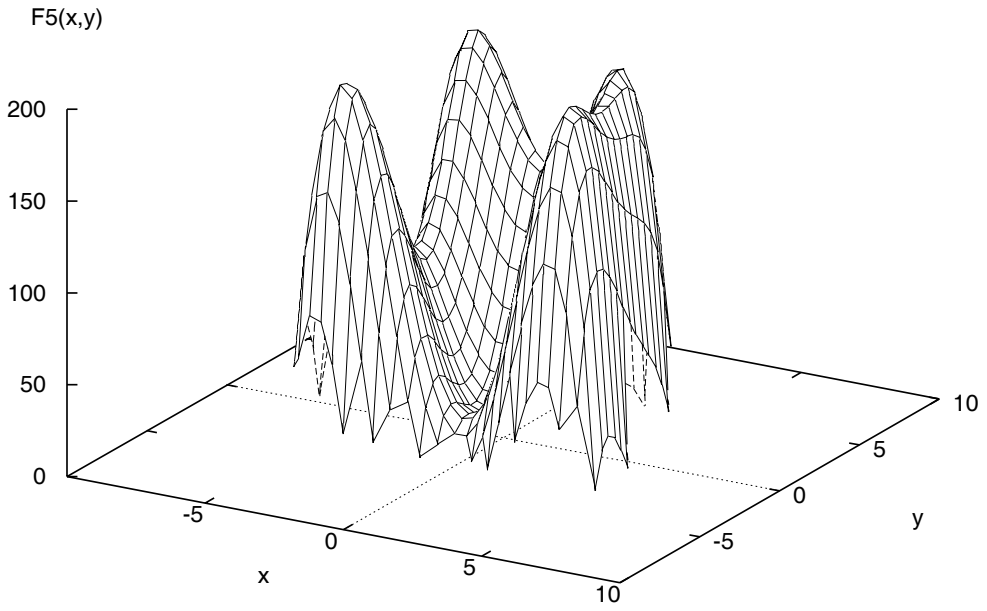


Figure 5.4: The Himmelblau function

maximum of 2000 iterations of the NichePSO algorithm. A single NichePSO iteration is defined as performing steps 2 to 8 in figure 5.2 once. A gradually decreasing inertia weight lets particles make slower velocity and position updates, at the same time ensuring that they follow convergent trajectories [87]. Table 5.1 reports NichePSO parameter settings, as well as limits on the search space ranges of each test function. $|S|$ denotes the initial number of particles in the main swarm before any niche subswarms were created; μ is the subswarm merging threshold, and δ is the subswarm creation threshold. For functions $F1$ to $F4$, a particle consists simply of a potential x value. For function $F5$, a particle represents an (x, y) position. NichePSO is evaluated according to

- *Accuracy:* Thus how close the discovered optima are to the actual solutions; and
- *Success consistency:* The proportion of the experiments that found all optima.

Function	δ	μ	$\|\mathbf{S}\|$	x_{min}	$x_{max} = v_{max}$
$F1$	0.0001	0.001	30	0.0	1.0
$F2$	0.0001	0.001	30	0.0	1.0
$F3$	0.0001	0.001	30	0.0	1.0
$F4$	0.0001	0.001	30	0.0	1.0
$F5$	0.0001	0.01	20	-5.0	5.0

Table 5.1: NichePSO Parameter Settings

The parameter values for δ and μ presented in table 5.1 have been experimentally found to be effective.

Table 5.2 reports the mean and standard deviation in fitness of all particles in all subswarms. Fitness here is simply defined as the function value $f(x)$ for each of the test problems. Only global optima are considered (suboptimal solutions in $F2$ and $F4$ are not taken into account). *%Converged* signifies the percentage of experiments that successfully located all the maxima. NichePSO successfully located all global maxima of all the functions tested. For functions $F2$ and $F4$, NichePSO located the global maximum in all cases, but did not find all local maxima for all simulations. This explains the relatively large difference in fitness between functions $F1$ and $F3$, and functions $F2$ and $F4$.

Table 5.2: Performance Results

Function	**Fitness**	**Deviation**	**NichePSO % Converged**
$F1$	$7.68E-05$	$2.20E-04$	100%
$F2$	$9.12E-02$	$6.43E-02$	93%
$F3$	$5.95E-06$	$4.86E-05$	100%
$F4$	$8.07E-02$	$6.68E-02$	93%
$F5$	$4.78E-06$	$1.03E-05$	100%

5.4 Analysis

Unimodal optimization techniques, such as the standard PSO and GAs, fail to locate multiple solutions to multimodal problems because of their inherent unimodal optimization nature. These algorithms need to be extended to facilitate niching and speciation abstractions. NichePSO is no exception – although the essence of the original PSO is retained, a number of extensions were made. The motivations and reasoning behind these extensions are presented, justified and investigated in this section. The following issues are considered:

- The algorithm's sensitivity to the niching parameters, μ and δ,
- the performance of GCPSO compared to *gbest*, and
- the relationship between the initial swarm size and the number of solutions in a multimodal fitness function.

Finally, the scalability of NichePSO on highly multimodal functions are considered.

5.4.1 Sensitivity to Changes in μ

Each subswarm created by the NichePSO algorithm can be seen as a hyper-sphere in the search space. The hyper-sphere's radius is determined by the Euclidean distance between the swarm's global best position and the particle in the swarm that lies furthest from it. Two subswarms are merged when the two conceptual hyper-spheres that they represent overlap. When all particles in a swarm have converged on a single solution, a swarm will have an effective radius of zero. In such a situation, equation (5.4) fails to allow similar

Solutions involved	**Distance between solutions**
$\|\mathbf{A} - \mathbf{B}\|$	0.714
$\|\mathbf{B} - \mathbf{C}\|$	0.622
$\|\mathbf{C} - \mathbf{D}\|$	0.675
$\|\mathbf{A} - \mathbf{D}\|$	0.493

Table 5.3: Normalized Inter-solution Distances for Function $F5$

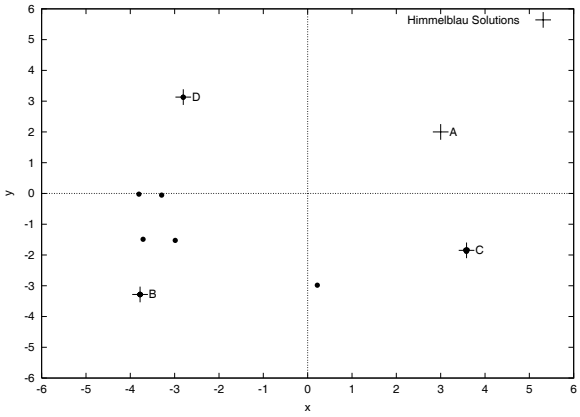
swarms to be merged. Therefore, the use of the μ parameter was introduced in equation (5.5), to allow virtually identical swarms to be merged when they occupy positions that are approximately similar. Large values of μ allow swarms that settle on different solutions, to merge. If two swarms that correctly represent different solutions are merged, the newly created swarm eventually converges on only one of the possible solutions, because of the subswarm optimization technique: The GCPSO algorithm searches for a single solution.

Figures 5.5(a), 5.5(b) and 5.5(c) illustrate the effect that different μ values have on the convergence capabilities of NichePSO, tested on function $F5$. Each particle in the swarm is represented by a '•'. The position of each solution is indicated by a '+', and labelled by a letter of the alphabet.

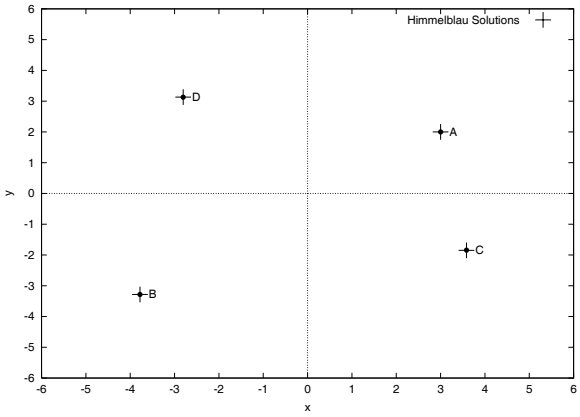
The 'goodness' of a particle's position can be determined by its proximity to the indicated solutions. Table 5.3 presents the normalized distances between the known solutions for function $F5$. The symbols correspond to those used in figures 5.5(a), 5.5(b) and 5.5(c). For $\mu = 0.5$, the NichePSO algorithm did not find all maxima for function $F5$ (see figure 5.5(a)). From table 5.3, the normalized distance between solutions A and D is 0.493. Swarms that represent solutions A and D were therefore merged, as their inter-niche solution distance was less than the threshold value μ. For μ-values less than 0.5, the algorithm successfully located all solutions (see for example figure 5.5(b)). For extremely small μ-values, NichePSO still successfully located all solutions, but not all swarms that were positioned on the same solutions were merged (see solution D in figure 5.5(c)). Swarms congregated around the solutions, but due to the 'strict' merging threshold, they could only be merged when virtually identical.

Figures 5.6(a) and 5.6(b) plot the mean number of solutions found by NichePSO for $F5$, $F1$ and $F3$ respectively, for different μ values. For both functions $F1$ and $F3$, NichePSO located all solutions when $\mu \leq 0.1$ (see figure 5.6(b)).

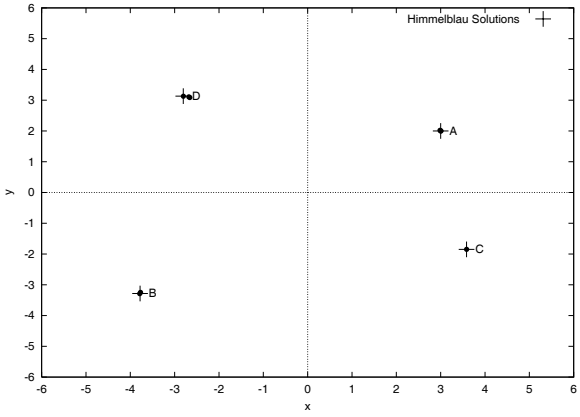
From figure 5.6, an upper bound on μ can be derived: μ should not be greater than the lowest inter-niche distance. The upper bound is similar to the assumptions made about the inter-niche distance, $2\sigma_{sh}$, in Goldberg's fitness sharing technique [32], and the niche radius r in Beasley *et al*'s sequential niching technique [4].



(a) $\mu = 0.5$

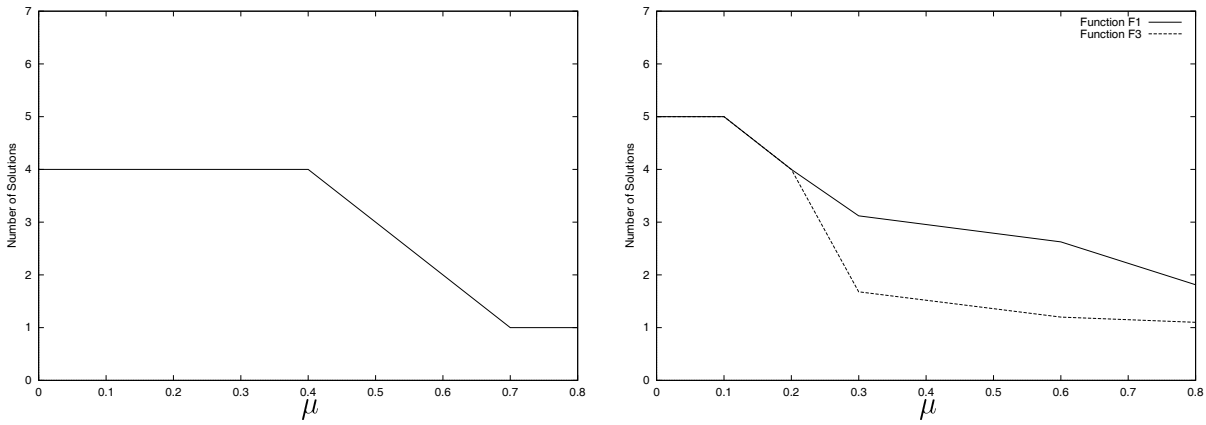


(b) $\mu = 0.4$



(c) $\mu = 0.0001$

Figure 5.5: Effect of changes in μ



(a) Number of solutions vs. μ for function $F5$

(b) Number of solutions vs. μ for functions $F1$ and $F3$

Figure 5.6: Number of solutions vs. μ for functions F1 and F3

5.4.2 Sensitivity to Changes in δ

To identify new potential solutions, the NichePSO algorithm monitors changes in the particles of the main swarm. If any particle in the main swarm exhibits very little change in its position over a number of iterations of the algorithm, the particle has approached an optimum position. The optimum may be a local or global optimum. An effective measure to detect small changes in a particle i's position is to monitor the standard deviation σ_i in particle i's fitness over a number of training iterations, e_σ. When particle i's variance in fitness becomes less than a threshold value δ, a new subswarm is created using particle i and its closest neighbor. A particle exhibits this behavior only when it is approaching a solution and has a low velocity, or when it is oscillating around a potential solution.

Different δ values were tested for functions $F1$, $F3$ and $F5$. Figure 5.7 plots the mean number of fitness function evaluations over 30 simulations for each test function against different δ settings. The number of fitness function evaluations are considered to illustrate that some δ values increase the time required for the algorithm to converge. A simulation was considered to have converged when all its subswarms had a fitness

less than 10^{-4}. For relatively large δ values ($\delta > 0.1$), NichePSO initially easily created subswarms with any particle that remotely exhibited stagnating behavior. In this context, 'stagnating behavior' indicates that a particle slowed down, and that it occupied similar positions in the search space over consecutive algorithm iterations. For small δ values ($\delta < 0.1$), particles were required to be more stationary before being considered for a subswarm. A different interpretation is that particles had to be very sure of a solution, before a subswarm was created. As indicated in figure 5.7, smaller δ values effected a slight increase in the number of fitness function evaluations required before the algorithm converged. From figure 5.7 only broad general trends can be seen. Each function appears to have an optimal δ value. For the test functions, these values are reported in table 5.4.

Figure 5.7 illustrates that NichePSO is not dependent on a finely tuned δ. Fervent subswarm creation with a 'high' δ value will be negated by the merging of similar swarms. Very small δ values ($\delta < 0.01$) leads to a minor performance penalty, but when compared to NichePSO's performance on higher δ values, the cost is low. Without exception, NichePSO successfully located *all* solutions to the test functions for all δ values used.

5.4.3 The Subswarm Optimization Technique

The NichePSO algorithm uses the GCPSO technique (refer to section 2.7.1) as subswarm optimization technique. This section compares two implementations of NichePSO: one using *gbest* and the other using GCPSO.

When considering the particle position update given in equation (2.12), it is clear that when, for a particle i, its position $\mathbf{x}_i(t)$ at time step t becomes close to its personal best position $\mathbf{y}_i(t)$ and the global best position $\hat{\mathbf{y}}(t)$, the velocity update for the next

Test Function	**Optimal δ**
F1	0.01
F3	0.1
F5	0.2

Table 5.4: Optimal δ values

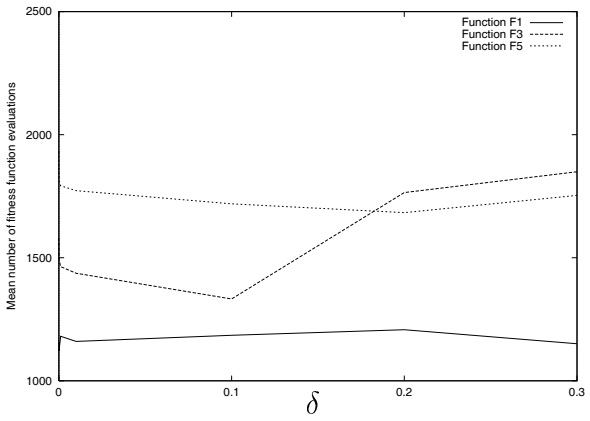


Figure 5.7: Mean number of fitness function evaluations required for different δ values

iteration of the algorithm, $\mathbf{v}_i(t+1)$ depends only on previous velocity values and the inertia weight w. A small $\mathbf{v}_i(t+1)$ dictates negligible change in a particle's position. The particle will therefore stagnate on its current position; $\hat{\mathbf{y}}(t)$ does not necessarily represent an optimum, but only the best solution found thus far by all particles in the search space. When a particle swarm consists of only two particles, as is frequently the case when subswarms are created with NichePSO, it may occur that these swarms stagnate almost immediately. The situation can be symbolically explained as follows:

With two particles, p and q, in a newly created subswarm, one of these particles, say p, immediately represents the global best position of the swarm. It also holds that the personal best position $\mathbf{y}_p$ of p is equal to the swarm's global best. This is an obvious assumption when considering that the global best position is identified as a personal best position of one of the particles in the swarm. With $\mathbf{x}_p = \mathbf{y}_p = \hat{\mathbf{y}}$, particle p's initial velocity update is then effectively reduced to

$$\mathbf{v}_p(t+1) \quad = \quad w\mathbf{v}_p(t) \tag{5.11}$$

Particle p's initial traversal of the search space therefore solely depends on its velocity vector $\mathbf{v}_p(0)$ and the value of the inertia weight w. When a subswarm is created, each particle in the new swarm not only retains its position vector, as this was the basis for its selection, but also its velocity vector. This ensures that the particle continues on its path to a local optimum. If the particle was already close to a potential solution, the magnitude of its velocity vector would be a value close to zero. Particle p will

therefore not easily move around in search space. Particle q was chosen to be the second particle in the subswarm, because it was the closest particle to p when the subswarm was created. This implies, as stated above, that $\mathbf{y}_p$ will always be considered over $\mathbf{y}_q$ for the swarm's initial global position, and consequently, that $\mathbf{x}_q$ will move towards $\mathbf{x}_p$. When $\mathbf{y}_p = \mathbf{y}_q = \hat{\mathbf{y}}$, the following conditions will occur:

- $\mathbf{x}_p \approx \mathbf{x}_q$, and
- $\mathbf{v}_p$ and $\mathbf{v}_q$ will approach zero.

Under these circumstances, no further learning takes place, and exploration of the search space is minimal. Again, it should be noted that the assumption that $\hat{\mathbf{y}}$ represents a global solution, cannot be made. To detect and avoid the described situation, the GCPSO algorithm was used. GCPSO uses adapted velocity and position update equations for the global best particle in a swarm (in this case particle p), that allows efficient local traversal of the search space. GCPSO avoids stagnation by moving the global best particle, until an optimum has been located.

Table 5.5 presents experimental results that compare the performance of GCPSO with *gbest* as subswarm optimization technique. The $\%Convergence$ column expresses an average success rate for finding *all* solutions of a test function over 30 simulations. It is clear that GCPSO was better suited towards maintaining niches than *gbest*. Experimental results obtained showed that for all functions, when using *gbest*, it frequently occurred that subswarms were formed that consisted of only two particles. Such swarms quickly stagnated on suboptimal locations. When a subswarm did not move, the probability that a particle in the main swarm would pass over it and be absorbed into the subswarm, was severely reduced. A subswarm consisting of only two particles that have stagnated on a suboptimal solution, is of no use. Results reported in table 5.5 ignored two particle swarms and reports only whether an experiment did manage to locate the actual solutions.

When several local and global optima exist in close proximity to each other in a function, GCPSO tends to be biased towards the global optima. This behavior is dictated by equations (2.16) and (2.15). GCPSO's addition of a random factor to the swarm's best particle position may place the particle closer to a global solution, hence forcing the

Table 5.5: % Convergence of experiments for GCPSO and *gbest*

Test Problem	**% Convergence: GCPSO**	**% Convergence: *gbest***
$F1$	100%	76%
$F2$	93%	66%
$F3$	100%	83%
$F4$	93%	86%
$F5$	100%	86%

subswarm to move towards the global solution that may already be well-represented by other subswarms. *gbest* does not exhibit this behavior, since the best particle's position is not modified.

5.4.4 Relationship between $|S|$ and the Number of Solutions

This section investigates the relationship between the swarm size $|S|$ and the number of optima, a, in a multimodal function.

Figures 5.8 and 5.9 present experimental results that compare the number of solutions found and the number of fitness function evaluations required for different swarm sizes. Reported results are means over 30 simulations for functions $F1$, $F3$ and $F5$. Trivially,

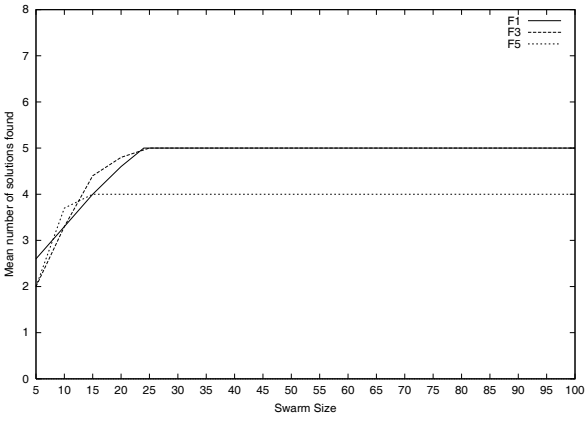


Figure 5.8: Relationship between different swarm sizes and the number of solutions located.

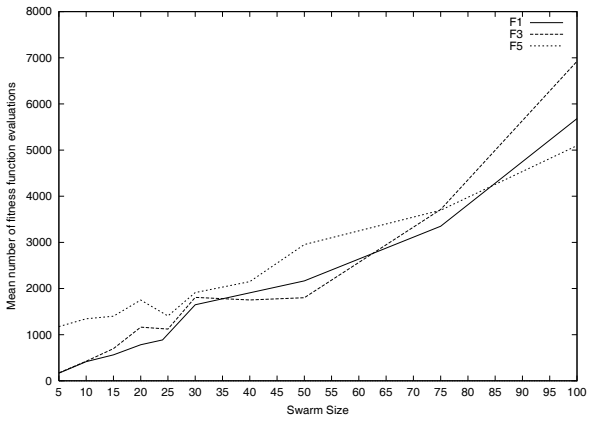


Figure 5.9: The mean number of fitness function evaluations required for different swarm sizes.

NichePSO failed to locate all solutions when $|S| < a$. When $|S| < 2a$, NichePSO also did not locate all the solutions. Since the subswarm creation technique needed two particles to create a subswarm, intuitively, $2a$ would have been expected to be a sufficient swarm size. This was however not the case. This situation can be clarified when considering the distribution of particles, and the fact that velocity vectors were initialized randomly. No 'directional-bias' is introduced by forcing velocity vectors to lead a particle into a specific direction, towards a solution. If possible solutions are not known in advance, this would not be possible. A particle could therefore be initialized close to a solution, but an initial velocity value may cause it to move away from the possible solution towards another solution, where it could eventually settle. For function $F1$, when $|S| \geq 25$, NichePSO successfully located and stably maintained all solutions. For all the tested functions, a swarm of size $|S| \geq a^2$ managed to locate all solutions. It serves to confirm the suspected relationship between the number of solutions in a multi-modal problem and the swarm size. The relationship $|S| \geq a^2$ prescribes acceptable swarm sizes for the test problems considered in this section. Results in section 5.4.5 show it is however not a general condition. Equation (5.15) in section 5.4.5 presents a more general formulation of the relationship between the number of solutions and swarm size.

Figure 5.9 shows the mean number of fitness function evaluations required for the different swarm sizes used above. As expected, the number of fitness function evaluations

steadily increases as the swarm size grows. The trend illustrates that the use of larger swarms does not necessarily benefit the optimization process:

- Convergence speed is not improved.
- Smaller swarms, subject to the condition identified above, yield the same results with lower complexity.

5.4.5 Scalability of NichePSO

The results obtained in section 5.3 showed NichePSO to effectively solve multimodal optimization problems. This section presents empirical results that investigate the scalability on NichePSO to massively multimodal domains. NichePSO was tested on the following two multimodal functions:

Griewank function:

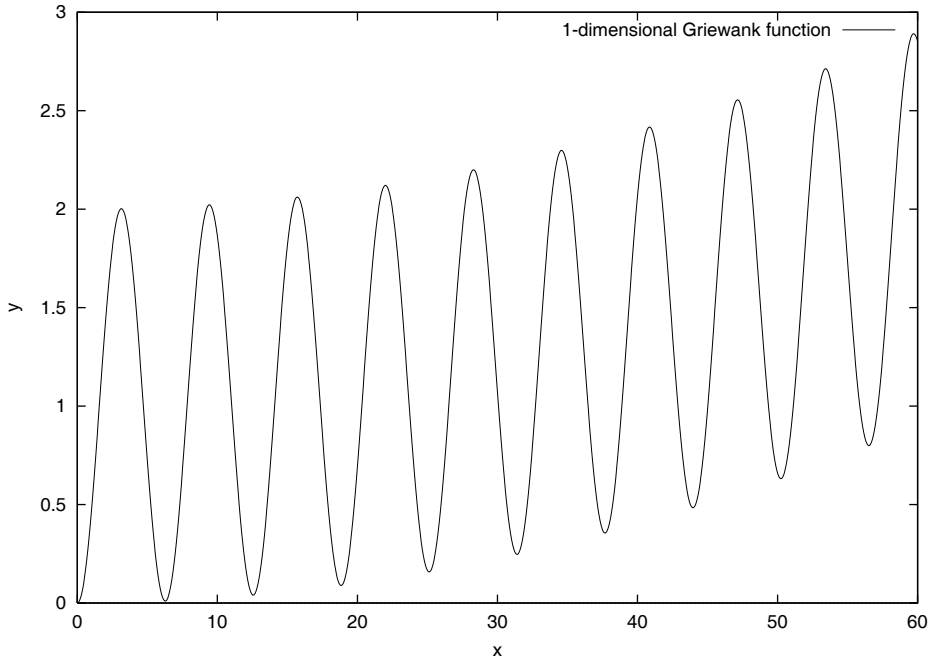
$$f(\mathbf{x}) = \left(\frac{1}{4000}\sum_{i=1}^{n} x_i^2\right) - \left(\prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)\right) + 1 \tag{5.12}$$

Rastrigin function:

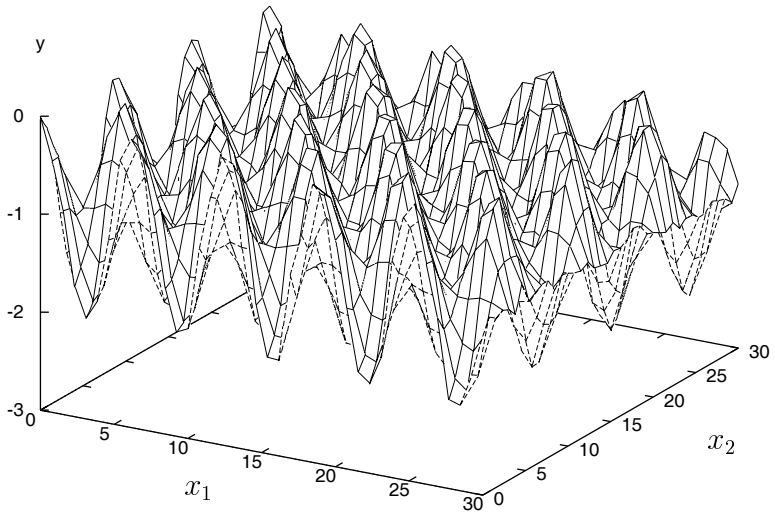
$$f(\mathbf{x}) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos\left(2\pi x_i\right) + 10\right] \tag{5.13}$$

These functions are massively multimodal. Both contain a single global minimum at the origin of the n-dimensional real-valued space in which they are defined. For each of the functions, the number of minima increase exponentially, as can be seen from the one and two dimensional plots given in figures 5.10 and 5.11. Figures 5.10(b) and 5.11(b) are drawn inverted to more clearly illustrate the multimodal nature of the function surfaces. The goal of the experiments were to ascertain whether increased dimensionality and large numbers of optima degraded the performance of NichePSO.

For each of the test functions, 10 experiments were performed with the NichePSO algorithm. Initial swarms sizes used were as listed in tables 5.6 and 5.7. For all experiments, the inertia weight w was scaled linearly from 0.7 to 0.1 over a maximum of 2000 training iterations. The acceleration coefficients were set to $c_1 = c_2 = 1.2$. NichePSO

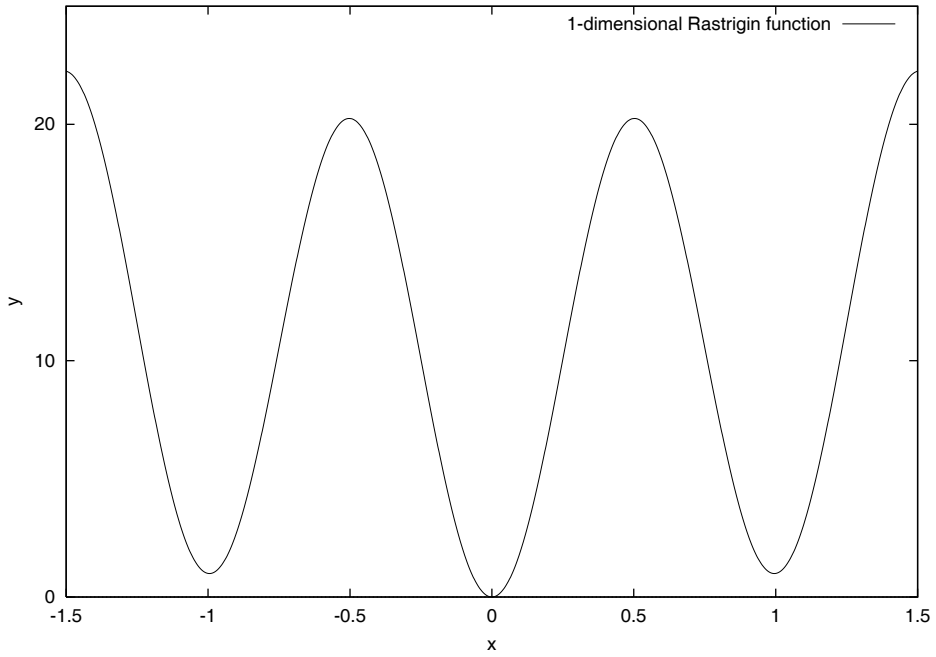


(a) One-dimensional Griewank function

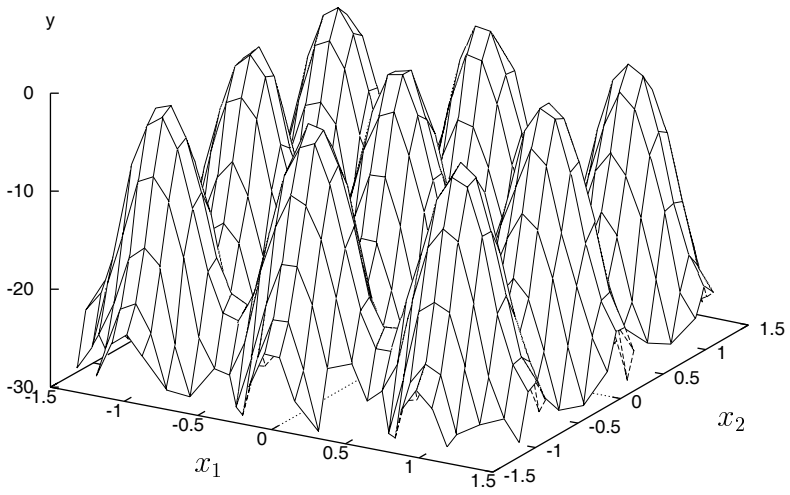


(b) Two-dimensional Griewank function

Figure 5.10: Griewank function



(a) One-dimensional Rastrigin function



(b) Two-dimensional Rastrigin function

Figure 5.11: Rastrigin function

Dimensions (n)	Number of solutions (a)	Swarm Size ($\|S\|$)	% Accuracy
1	5	20	100.00%
2	25	100	100.00%
3	625	2500	94.75%

Table 5.6: Performance on the Griewank function

Dimensions (n)	Number of solutions (a)	Swarm Size ($\|S\|$)	% Accuracy
1	3	9	100.00%
2	9	36	100.00%
3	27	108	97.45%
4	81	324	97.08%
5	243	972	92.00%

Table 5.7: Performance on the Rastrigin function

parameters were set as $\mu = 0.001$ and $\delta = 0.1$. The Griewank function was investigated in the range $[-28, 28]^n$, and the Rastrigin function in the range $[-\frac{3}{2}, \frac{3}{2}]^n$. Tables 5.6 and 5.7 present performance results of NichePSO on the two test functions.

The following observations can be made:

- Given the sharp increase in the number of optima, NichePSO gave consistent performance, with slight degradation as the number of dimensions increased. It should be taken into account that a linear increase in the number of dimensions is coupled with an *exponential* increase in the number of solutions.

- Using the exponential relationship $|S| = a^2$ suggested in section 5.4.4, is computationally very expensive. As an example, the 5-dimensional Rastrigin function with 243 solutions would require a swarm of $59,049$ particles! As shown in tables 5.6 and 5.7, the relationship between the number of solutions (a) and the swarm size ($|S|$) was kept at

$$|S| = 4a \tag{5.14}$$

  This relationship is computationally more tractable. It also shows that $|S| = a^2$ does not represent a lower bound on the relationship between swarm size and num-

ber of solutions. Consequently, the relationship between swarm size and number of solutions would more likely be expressed as

$$|S| = c \cdot a^q \tag{5.15}$$

where c is a constant, and $1 \leq q \leq 2$. Further experimentation would be required to empirically estimate ideal values for c and q.

5.5 Conclusion

Swarm intelligence algorithms such as particle swarm optimizers, present a real and viable alternative to existing numerical optimization techniques. Population based optimization techniques can rapidly search large and convoluted search spaces and less likely to be sensitive to suboptimal solutions. The standard *gbest* and *lbest* PSO approaches share information about a best solution found by the swarm or a neighborhood of particles. Sharing this information introduces a bias in the swarm's search, forcing it to converge on a single solution. When the influence of a current best solution is removed, each particle traverses the search space individually, using no expierential knowledge of its peers.

With NichePSO, a subswarm is created when a possible solution is detected at a particle's location. The subswarm is then responsible for traversing the search space in the vicinity of the potential solution to find an optimal location. This pseudo-memetic approach is called NichePSO. Experimental results obtained on a set of multimodal functions showed that NichePSO successfully located and maintained multiple optimal solutions. Several parameter optimization issues, related to NichePSO, were addressed. Suggestions were made as to potential values for tunable parameters. A scalability study carried out on highly multimodal functions, commonly used in the study of PSO algorithms, were used to demonstrate that NichePSO scales acceptably on high dimensional problems.