Chapter 6

Vector-Based PSO Applied to Static Environments

This chapter presents experimental results obtained when the three vector-based PSO (VBPSO) algorithms are applied to static environments. Extensive test results obtained by applying the various versions of the algorithm to a number of benchmark functions are presented. The final version of the VBPSO is analysed by empirically testing the sensitivity of the algorithm to different parameter values, its ability to scale to highly multi-modal functions, and the relationship between the initial swarm size and the algorithm's performance. The chapter is concluded by presenting the results of a comparative study of the performance of three diverse PSO niching algorithms, namely the VBPSO, NichePSO [13] [14] and the species-based PSO [57]. These algorithms are applied to a number of two-dimensional functions with varying characteristics.

6.1 Introduction

Each of the vector-based PSO algorithms was evaluated on a number of benchmark functions. The sequential version, being the first attempt to use vector dot products to identify niches, was only evaluated on a small number of functions. This algorithm finds duplicate results if subswarms in more than one niche converge on the same optimum, and the number of duplicates increases with increasing complexity of benchmark functions. Therefore, the sequential

VBPSO was not investigated further. The two parallel versions, which are improvements of the sequential vector-based PSO, were evaluated on the same functions as the sequential version, as well as a number of additional functions. These functions were carefully selected to present a variety of landscapes, so that the performance of the parallel vector-based PSO algorithms could be evaluated in circumstances where the shapes, sizes and positions of the niches differed considerably.

The remainder of the chapter is organized as follows: Section 6.2 presents and discusses experimental and statistical procedures, gives an overview of a number of selected benchmark functions, and lists general and specific parameter settings for the experiments. Results of the sequential VBPSO are discussed in section 6.3, while section 6.4 presents results of the parallel and enhanced parallel VBPSO. These sections culminate in an identification of the best algorithm, i.e. the enhanced parallel VBPSO. A more in-depth analysis of this algorithm is done in section 6.5. A comparison of nichePSO, the species-based PSO, and the enhanced parallel VBPSO is given in section 6.6.

6.2 Experimental procedure

This section presents procedures that were followed to test the family of vector-based PSO algorithms on a number of benchmark functions. In section 6.2.1 various approaches and general settings are discussed. Section 6.2.2 presents descriptions of statistical procedures while the benchmark functions that were selected to test the VBPSO algorithms are described in section 6.2.3. Section 6.2.4 lists specific settings, namely initial swarm sizes and granularity, for the benchmark functions.

6.2.1 General procedures and settings

Two approaches used to initialize the swarm are described, and general parameter values are given and explained. For each algorithm, the functions were tested with a number of different settings. The settings for these experiments are given and explained below:

Initializing the swarm: To locate all the optima in the search space, it is essential that particles are distributed uniformly throughout the search space before niches are identified. Therefore, a good random number generator is needed to initialize the swarm. This section evaluates two approaches to initialize swarms, namely using Sobol sequences and the random number generator supplied by the C++ compiler. The purpose of this section is to compare performances of the algorithm using these two approaches. This is the first study, to the author's knowledge, that evaluates the influence of random number generators on the performance of PSO niching methods.

Random numbers are commonly generated by means of a system-supplied number generator, for example, *rand*() in C++ and *Rand*() in Excel. Sequences of these numbers are called pseudo-random because the order of these numbers looks unpredictable. Such generators are almost always *linear congruential generators*, which generate a sequence of integers $I_1, I_2, I_3, \ldots$, each between 0 and $m-1$ (e.g. $RAND\_MAX$), by the recurrence relation

$$I_{j+1} = aI_j + c(mod\ m) \tag{6.1}$$

where m is the *modulus*, and a and c are positive integers called the *multiplier* and the *increment* respectively. A fixed sequence of numbers is generated and, depending on m, will eventually repeat itself. Successive random numbers are obtained by successive calls to the *rand*() function. In general, a sequence is initialized by a call to a function supplying a *seed* obtained from an (unpredictable) external feeder like the computer clock. The linear congruential method is very fast and easy to implement. However, it is not free of sequential correlation on successive calls, and many of the available pseudo-random number generators have flaws leading to low-order correlations [76].

Alternatively, quasi-random sequences, also called low-discrepancy sequences, can be used. The low discrepancy property is a measure of uniformity for the distribution of the points, ensuring no large gaps and no clustering of points in the d-dimensional hypercube. The Sobol sequence is one of the most popular quasi-random sequences because of its simplicity of implementation [11] [47]. An efficient implementation proposed by Antonov and Saleev [2] uses Gray codes, a binary numerical system where two successive values differ in only one bit.

The (binary-reflected) Gray code of an integer i is defined as

$$gray(i) = i \oplus \left\lfloor \frac{i}{2} \right\rfloor = (\ldots i_3 i_2 i_1)_2 \oplus (\ldots i_4 i_3 i_2)_2 \tag{6.2}$$

where $\oplus$ indicates the exclusive or operation.

Sobol points over a unit interval are generated using

$$x^n = g_1 v_1 \oplus g_2 v_2 \oplus \ldots \tag{6.3}$$

where $v_1, v_2, \ldots$ is a set of *direction numbers*. Each v_i is a binary fraction that can be written

$$v_i = m_i/2^i$$

where m_i is an odd integer with $m_i \in [0, 2^i]$.

To obtain v_i, a polynomial with coefficients from $[0, 1]$ is chosen and the coefficients are used to define a recurrence for calculating v_i. The recurrence may also be expressed in terms of m_i. To calculate points in one dimension, all m_i are set to 0.

A more detailed explanation of the implementation of Sobol's quasi-random sequence generator is presented in [11] and [47].

Initial swarm sizes: The number of particles that are created during the initialization phase depends on the test function as well as the size of the search space. Since the function landscape is unknown, an initial swarm size must be estimated. Table 6.9 in section 6.2.4 summarizes swarm sizes with which the functions have been initialized for all algorithms. These sizes showed to be sufficient to obtain good results.

Number of iterations: Separate independent experiments were conducted with the number of iterations set to 200 as well as 500, and for Sobol sequences as well as a system-supplied random number generator. Although function evaluations are counted, the number of function evaluations is not used as a stopping condition. Extending the size of some of the subswarms (discussed in section 5.5.1) may cause the swarm to be optimized with more than the number of initial particles. Therefore, the number of function evaluations may differ considerably from run to run. Using a set number of iterations ensures that each particle is updated the same number of times. Since a number of preliminary tests indicated that the algorithms located most optima after 200 iterations, the number of iterations was set to 200. To observe the effect of a larger number of iterations on performance, results were obtained with 500 iterations as well.

Parameter settings: For all three algorithms the random sequences used in the velocity update equation are scaled by constants $c_1, c_2 \in [0, 2]$. As explained in section 5.5.1, both these values were set to 1, and the inertia weight, w, was set to 0.8.

Granularity settings: For the parallel and enhanced parallel PSO algorithms merging takes place when the Euclidian distance between the neighbourhood best positions of two niches becomes less than a merging threshold or *granularity*. The granularity is problem-dependent and has to be set in advance. According to results presented in section 6.5.2 niches merge effectively if the granularity is less than the smallest interniche distance. However, smaller values are preferable to prevent merging taking place too soon. Taking the function landscape and the expected number of optima into account, this value was set to 0.05 for the four one-dimensional functions. Granularity values for the other test functions are summarized in Table 6.9. Since the enhanced parallel VBPSO performed better than the other two algorithms, a study to determine the sensitivity of this algorithm to granularity values was undertaken. Results for various functions and the factors that influence the performance of the algorithm, are discussed in section 6.5.2.

Merging intervals: Niches are merged at intervals of 50 iterations. The intervals were chosen arbitrarily and, as discussed in section 5.5.2 and Appendix A.2, no evidence could be found that it has any effect on performance.

Number of runs: For each of the functions described in section 6.2.3, 50 independent runs were executed for each combination of number of iterations and initialization approach. Results of these experiments are presented in sections 6.3 and 6.4.

6.2.2 Statistical procedures

Since normality of data samples can not be assumed, this study made use of the non-parametric Mann-Whitney U test [87] to determine if there are statistically significant differences between different implementations or instances of the algorithms. For this purpose the Z-value, which is the normal approximation of the Mann-Whitney U test statistic, was computed to evaluate the null-hypothesis, $H_0 : \mu_a = \mu_b$, at significance levels of 0.05 and 0.01. If the null-hypothesis can not be accepted, the alternative hypothesis, $H_1 : \mu_a \neq \mu_b$, is accepted. Here a and b refer to different instances or implementations of the algorithms.

In this study, the Mann-Whitney U test is used to compare the following:

- Experiments using Sobol sequences to those using the built-in random number generator in Borland C++.
- The three versions of the vector-based PSO.

Outcomes of the hypothesis tests are presented in sections 6.3 and 6.4.

6.2.3 Test functions

A number of one-dimensional and two-dimensional functions was selected to test the different algorithms.

The following one-dimensional functions were introduced by Goldberg and Richardson to test the performance of niching genetic algorithms [40]. Beasley *et al.* used the functions to test the sequential niching technique for multimodal optimization [4], while Brits *et al.* used the same functions to test the performance of NichePSO, a parallel niching algorithm [13] [14]. These functions present different situations required to test an algorithm on one-dimensional functions, namely evenly-spaced and non-evenly spaced optima, as well as similar and differing heights of the peaks.

$$F1(x) = \sin^6(5\pi x) \tag{6.4}$$

$$F2(x) = \left(e^{-2\log(2)\cdot(\frac{x-0.1}{0.8})^2}\right) \cdot \sin^6(5\pi x) \tag{6.5}$$

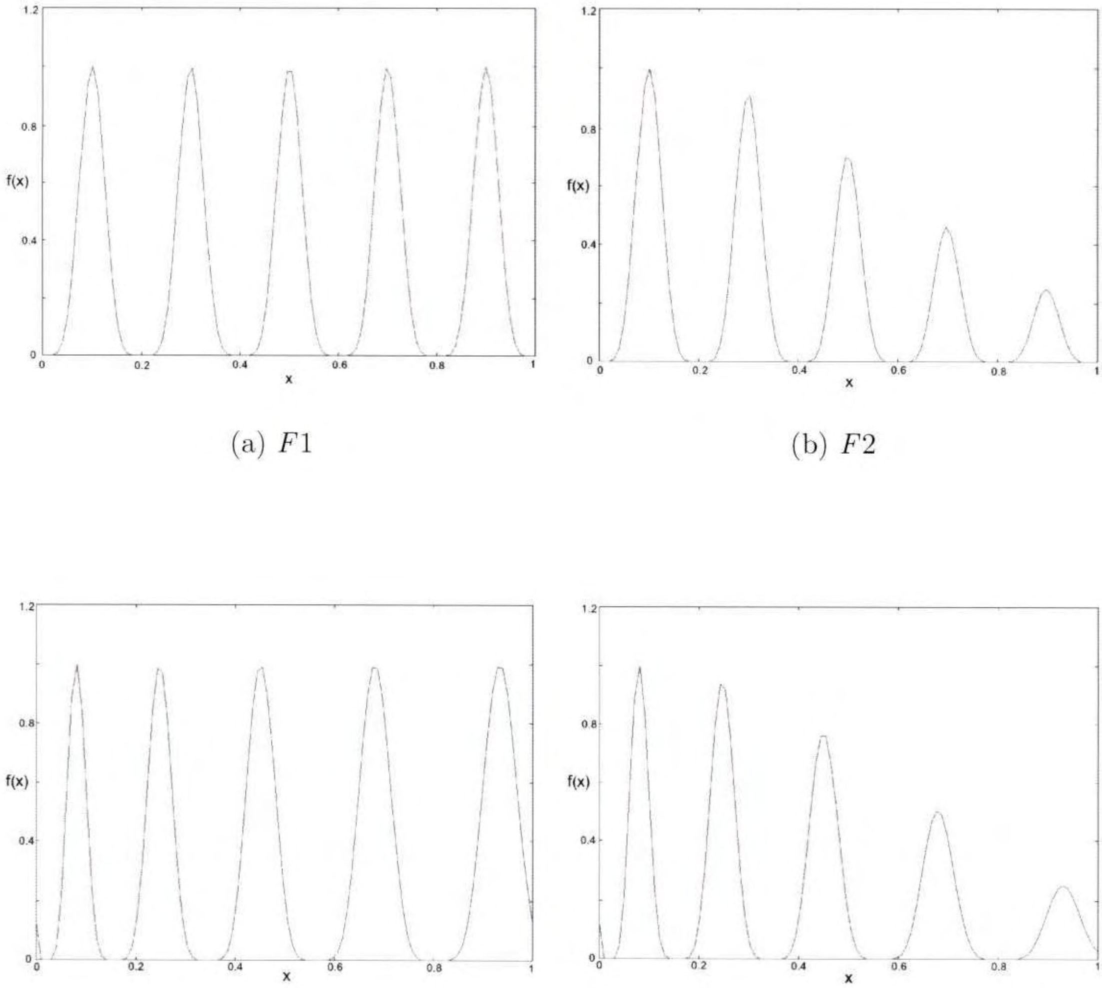
$$F3(x) = \sin^6(5\pi(x^{\frac{3}{4}} - 0.05)) \tag{6.6}$$

$$F4(x) = \left(e^{-2\log(2)\cdot(\frac{x-0.08}{0.854})^2}\right) \cdot \sin^6(5\pi(x^{\frac{3}{4}} - 0.05)) \tag{6.7}$$

Functions $F1$ to $F4$ all have 5 maxima between 0 and 1.0, but the spacing between the maxima and the function values at the peaks differ. For each of these functions, maxima occur at positions as listed in Table 6.1. Function landscapes are illustrated in Figure 6.1.

Table 6.1: Positions of maxima of one-dimensional functions

$F1$	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



(a) $F1$

(b) $F2$

(c) $F3$

(d) $F4$

Figure 6.1: One-dimensional functions

The following two-dimensional functions have been selected to test the vector-based PSO niching algorithms in a demarcated region of the search space. The functions that were selected include well-known benchmark functions that have been used to test other niching algorithms, as well as a number of functions specifically selected to illustrate the robustness of the vector-based algorithms when confronted with function landscapes where the shapes, sizes, and placing of the niches differ significantly.

The modified Himmelblau function: Figure 6.2 illustrates the modified Himmelblau function that is defined as

$$f(x_1, x_2) = 200 - (x_1^2 + x_2 - 11)^2 - (x_1 + x_2^2 - 7)^2 \tag{6.8}$$

The Himmelblau function, defined in two dimensions, is a popular choice for testing niching algorithms, as it has four well-defined optima with similar fitnesses. The positions of the four optima are listed in Table 6.2. Most niching algorithms, including genetic algorithms, yield good results for this function. The function is not repetitive, that is, the landscape where the optima occur, is not repeated elsewhere in the search space. The Himmelblau function was tested in the range $x_1, x_2 \in [-6, 6]$, where the optima occur.

Table 6.2: Optima of the Himmelblau function

Locations of optima		Fitness
x_1	x_2	$f(x_1, x_2)$
3	2	200
3.58	-1.85	200
-2.81	3.13	200
-3.78	-3.28	200

The Griewank function: The Griewank function is illustrated in Figure 6.3 in the range $x_1, x_2 \in [-5.0, 5.0]$ and in a larger range. The function is described by

$$f(\mathbf{x}) = -\left(\left(\frac{1}{4000}\sum_{i=1}^{n} x_i^2\right) - \left(\prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}})\right) + 1\right) \tag{6.9}$$

The function is also massively multimodal with one global optimum at $[0, 0]^n$, where n indicates the number of dimensions. Griewank has an infinite number of optima of

Table 6.3: Optima of the Griewank function

Locations of optima		Fitness
x_1	x_2	$f(x_1, x_2)$
3.14	4.44	-0.0074
3.14	-4.44	-0.0074
0	0	0
-3.14	4.44	-0.0074
-3.14	-4.44	-0.0074

decreasing fitness as the distance from the global optimum increases. However, the positions of the optima are not spaced as evenly as in the case of the Rastrigin function. The Griewank function is tested in the range $x_1, x_2 \in [-5.0, 5.0]$. A two-dimensional version of the Griewank function has 5 optima in this range. The positions of the optima are listed in Table 6.3.

The Rastrigin function: Figure 6.4 shows the Rastrigin function in the range $x_1, x_2 \in [-1.25, 1.25]$ and in a larger range to illustrate its massive multimodality. The function is defined as

$$f(\mathbf{x}) = -\left(\sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]\right) \tag{6.10}$$

The Rastrigin function is also very popular for testing niching algorithms, as it is massively multimodal. The function has a global optimum at $[0,0]^n$ where n indicates the number of dimensions, as well as an infinite number of optima radiating out from the global optimum. The fitness of these optima decreases as the distance from the global optimum increases. The function is also eminently suitable for testing scalability, as optima are situated at regularly spaced intervals within the problem space. To test for a finite number of optima, a search space has to be defined. For this experiment the Rastrigin function was tested in the range $x_1, x_2 \in [-1.25, 1.25]$, where the two-dimensional version has 9 optima. The positions of these optima are listed in Table 6.4.

The Ackley function: The Ackley function is defined as

$$f(\mathbf{x}) = 20 + e - 20e^{-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2x_i)} \tag{6.11}$$

Table 6.4: Optima of the Rastrigin function

Locations of optima		Fitness
x_1	x_2	$f(x_1, x_2)$
-1.0	-1.0	-2.0
-1.0	0.0	-1.0
-1.0	1.0	-2.0
0.0	-1.0	-1.0
0.0	0.0	0.0
0.0	1.0	-1.0
1.0	-1.0	-2.0
1.0	0.0	-1.0
1.0	1.0	-2.0

Table 6.5: Optima of the Ackley function

Locations of optima		Fitness
x_1	x_2	$f(x_1, x_2)$
-1.0	-1.0	-2.0
-1.0	0.0	-1.0
-1.0	1.0	-2.0
0.0	-1.0	-1.0
0.0	0.0	0.0
0.0	1.0	-1.0
1.0	-1.0	-2.0
1.0	0.0	-1.0
1.0	1.0	-2.0

Figure 6.5 shows the function landscape in the range where it is tested, and in a larger range to illustrate the multimodal character of the function. The function also has an infinite number of optima surrounding a central global optimum. Fitnesses of the surrounding optima decrease sharply. In addition, the diameters of the niches in the function landscape become smaller as the distances from the central global optimum increase, making it quite difficult to detect niches some distance away from the center. For this experiment the Ackley function is tested in the range $x_1, x_2 \in [-1.6, 1.6]$ where the function has 9 optima. The positions of these optima are listed in Table 6.5.

Table 6.6: Optima of the Ursem F1 function

Locations of optima		Fitness
x_1	x_2	$f(x_1, x_2)$
-1.44	0	3.25
1.70	0	4.82

The Ursem F1 function: The Ursem F1 function is defined as

$$f(x_1, x_2) = -(sin(2x_1 - 0.5\pi) + 3cos(x_2) + 0.5x_1) \tag{6.12}$$

The function has a repetitive character and well-defined optima occur indefinitely, as illustrated for two ranges in Figure 6.6. However, it has no central optimum, but the fitnesses decrease linearly while the shapes of the niches remain the same. When the problem space is demarcated to contain two optima with different fitnesses, Ursem F1 is a simple two-dimensional function that should give good results for any niching algorithm. The Ursem F1 function is tested in the range $x_1 \in [-2.5, 3.0]$ and $x_2 \in [-2.0, 2.0]$, a region containing two of the optima. Table 6.6 lists the positions of these optima.

The Ursem F3 function: The Ursem F3 function is defined as

$$\begin{aligned} f(x_1, x_2) = &- (sin(2.2\pi x_1 - 0.5x_1)) \cdot \left(\frac{2 - |x_2|}{2}\right) \cdot \left(\frac{3 - |x_1|}{2}\right) \\ &- (sin(0.5\pi x_2^2 + 0.5\pi)) \cdot \left(\frac{2 - |x_2|}{2}\right) \cdot \left(\frac{2 - |x_1|}{2}\right) \end{aligned} \tag{6.13}$$

The landscape of the function contains a number of flattened peaks. Groups of four peaks with varying heights give the landscape the appearance of a mountain range. Figure 6.7 illustrates the function for small and large ranges. Groups of four peaks are repeated at specific intervals. The algorithms are tested with one such group in the range $x_1, x_2 \in [-2.0, 2.0]$. The positions of the optima in this group are listed in Table 6.7.

Table 6.7: Optima of the Ursem F3 function

Locations of optima		Fitness
x_1	x_2	$f(x_1, x_2)$
-1.72	0	1.5
-0.74	0	2.5
0.25	0	2.5
1.23	0	1.5

Table 6.8: Optima of the Six Hump Camel function

Locations of optima		Fitness
x_1	x_2	$f(x_1, x_2)$
-0.80	1.70	0.22
-0.71	0.09	1.03
-0.57	-1.61	-2.10
0.57	1.61	-2.10
0.71	-0.09	1.03
0.80	-1.70	0.22

The six hump camel function: The six hump camel function is illustrated in Figure 6.8 and defined as

$$f(x_1, x_2) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right) x_1^2 + x_1 x_2 + (-4 + 4x_2^2)x_2^2 \tag{6.14}$$

Like the Himmelblau function, the six hump camel function is defined in two dimensions and is not repetitive. It has six rounded peaks of which the heights differ considerably. The niche radii also differ, making it a good test function to estimate the robustness of niching algorithms. The range $x_1 \in [1.1, 1.1]$ and $x_2 \in [-1.9, 1.9]$ contains six optima and the function is tested in this region. Table 6.8 lists the positions of these optima.

6.2.4 Initial swarm sizes and granularity

Table 6.9 lists the initial swarm sizes and granularity settings used for all experiments for which results are presented in sections 6.3 and 6.4.

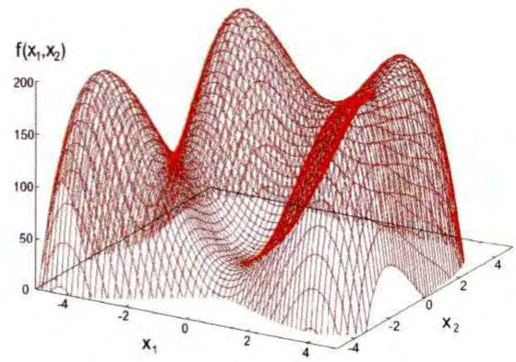
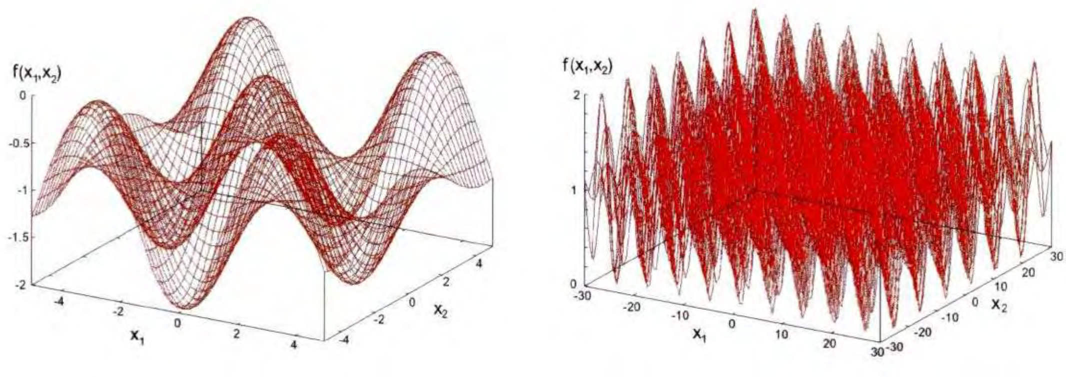
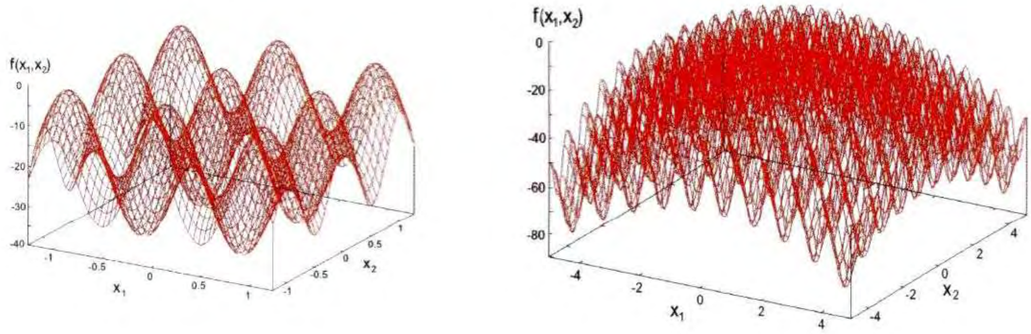


Figure 6.2: The Himmelblau function showing maxima.



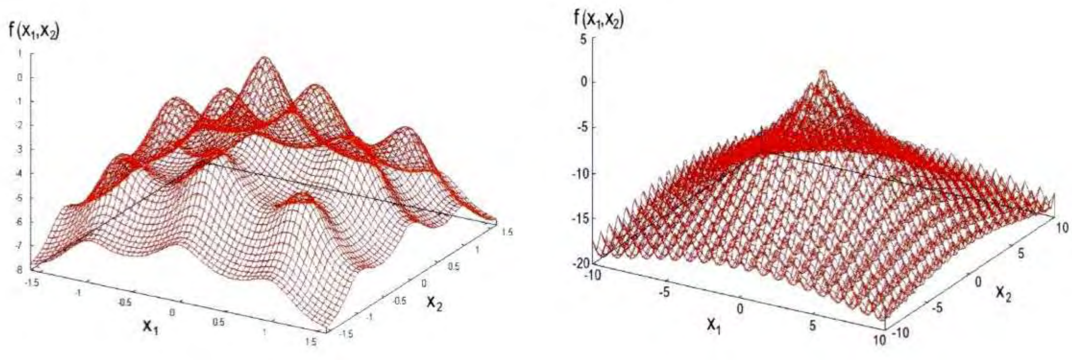
(a) (b)

Figure 6.3: The Griewank function.



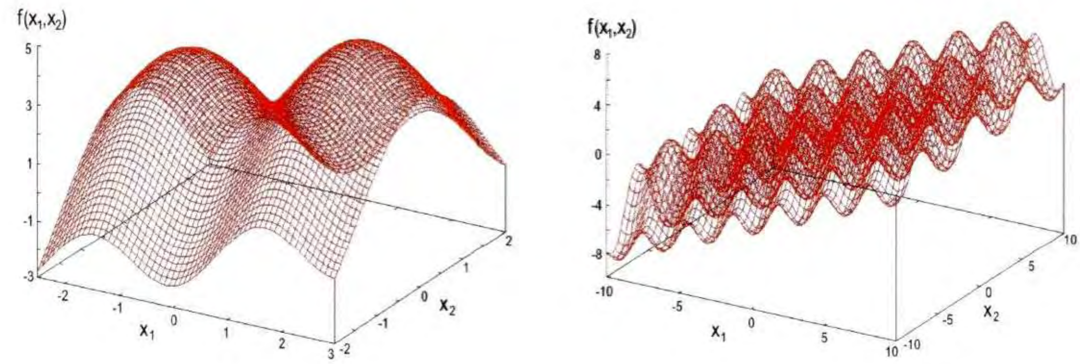
(a) (b)

Figure 6.4: The Rastrigin function.



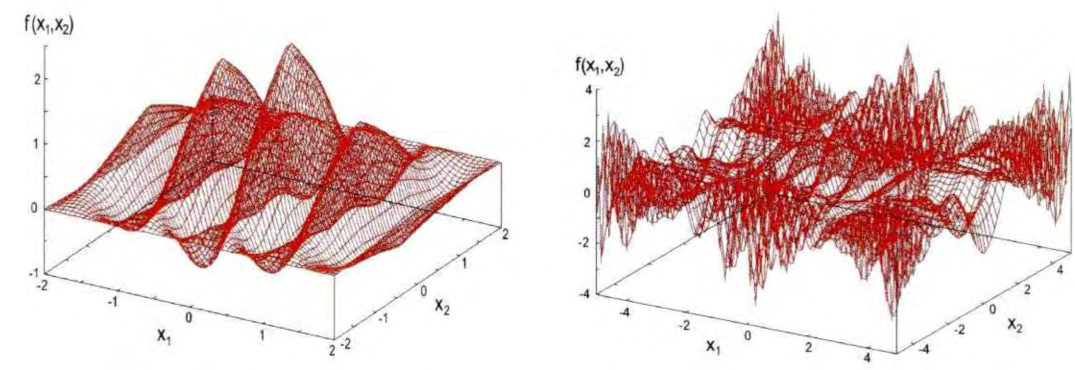
(a) (b)

Figure 6.5: The Ackley function.



(a) (b)

Figure 6.6: The Ursem F1 function.



(a) (b)

Figure 6.7: The Ursem F3 function.

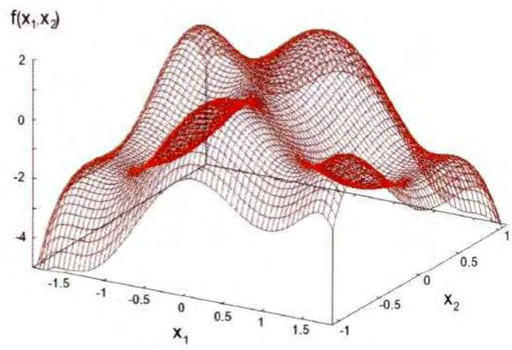


Figure 6.8: The six hump camel function.

Table 6.9: Initial swarm sizes and granularity for test functions

Function	Swarm size	Granularity
$F1$	20	0.05
$F2$	20	0.05
$F3$	20	0.05
$F4$	20	0.05
Himmelblau	30	0.5
Griewank	40	0.5
Rastrigin	60	0.1
Ackley	60	0.3
Ursem F1	30	0.5
Ursem F3	40	0.3
Six hump camel	50	0.3

6.3 Results of the sequential vector-based PSO

To analyze the performance, and investigate the effect of particle initialization on performance of the sequential VBPSO, Tables 6.10, 6.11, and 6.12 summarize the average results of 50 experiments for functions $F1$ to $F4$ and the Himmelblau function. Results have been obtained using Sobol sequences as well as a system-supplied random number generator, and for 200 as well as 500 iterations. Therefore, each function yielded four independent sets of results.

The tables list the average number of function evaluations and the average number of solutions located per configuration over 50 runs, as well as the average number of duplicate solutions. The seventh column provides the average of the derivatives of the objective function, $f'(x)$, at positions where optima are located. Derivatives were used in these experiments to indicate the quality of the solutions, since the offline error (the method of choice), calculated to 6 decimal positions, yielded a value of 0 in all cases. The success rate indicates the number of optima located as a percentage of all possible optima over 50 runs.

The averages of the function evaluations, number of solutions and number of duplicate solutions are reported together with the standard error.

Table 6.13 shows statistical data generated by the Mann-Whitney U test for the sequential vector-based PSO. The success rate of experiments using Sobol sequences (sample 1) are compared to that of experiments using the system-supplied random number generator (sample 2). The following results are listed for the one-dimensional functions $F1$ to $F4$ and the two-dimensional Himmelblau function, for 200 and 500 iterations:

- the sums of the rankings of sample 1 (ΣR_1) and sample 2 (ΣR_2).
- Z, the normal approximation of the Mann-Whitney U test statistic [87].
- the outcomes of the hypothesis tests.

Discussion

Results produced by the sequential vector-based PSO algorithm on functions $F1$ to $F4$ and the Himmelblau function showed an average success rate of above 98% when Sobol sequences were used to generate initial particle positions and above 90% for a distribution by means of a system-supplied random number generator. For all functions, the average success rate was higher when Sobol sequences were used. However, the Mann-Whitney U test (Table 6.19) only shows a statistically significant difference between the two approaches for the sequential VBPSO with 200 iterations at a significance level of 0.05.

Results also show that, for each one-dimensional functions as well as the Himmelblau function, the average number of function evaluations recorded for experiments using Sobol sequences and those using a system-supplied random number generator differ very slightly for the same number of iterations. No trend could be observed. Averages of the derivatives of the four sets of results for each objective function were in the same order of magnitude. However, averages of derivatives for functions $F1$ and $F3$ as well as the Himmelblau function, were much smaller than the averages of derivatives for functions $F2$ and $F4$. For functions $F1$ and $F3$ distances between optima are similar, while this is not the case for functions $F2$ and $F4$. For the Himmelblau function, distances between optima differed only slightly. Therefore, the quality of the solutions depends to a large extent on the function landscape.

Although the sequential vector-based PSO performed well in the sense that it could locate all the optima in most of the runs, it produced a number of duplicate solutions. Functions $F1$ to $F4$ that have five optima each in the search space, produced an average of 1.4 duplicates per run. The Himmelblau function yielded an average of 7.8 duplicates per run while the search space only contains four optima. The conclusion was reached that the sequential VBPSO creates too many duplicates and was therefore excluded from further analysis.

Table 6.10: Sequential vector-based PSO results for one-dimensional functions $F1$ and $F2$

	Random number generator	Iterations	Average function evaluations	Average ♯ solutions	Average ♯ duplicates	Average derivative	Success rate
$F1$	System-supplied	200	5234 ± 126.2	4.76 ± 0.06	1.38 ± 0.2	3.01E-14 $\pm$ 2.25E-15	95.2%
		500	12374 ± 342.1	4.64 ± 0.07	1.18 ± 0.2	2.80E-14 $\pm$ 3.86E-15	92.8%
	Sobol sequences	200	4702 ± 119.5	5 ± 0	1.12 ± 0.21	2.91E-14 $\pm$ 2.26E-15	100%
		500	11479 ± 310.6	5 ± 0	0.9 ± 0.19	2.84E-14 $\pm$ 2.55E-15	100%
$F2$	System-supplied	200	5185 ± 125.3	4.66 ± 0.07	1.9 ± 0.26	0.11 ± 0.08	93.2%
		500	12786 ± 321.4	4.68 ± 0.1	1.78 ± 0.25	0.07 ± 0.12	93.6%
	Sobol sequences	200	4755 ± 108.7	4.96 ± 0.03	1.36 ± 0.21	0.07 ± 0.07	99.2%
		500	11811 ± 253.5	4.98 ± 0.07	1.36 ± 0.2	0.11 ± 0.07	99.6%

Table 6.11: Sequential vector-based PSO results for one-dimensional functions $F3$ and $F4$

	Random number generator	Iterations	Average function evaluations	Average ♯ solutions	Average ♯ duplicates	Average derivation	Success rate
$F3$	System-supplied	200	5144 ± 111.9	4.5 ± 0.1	1.58 ± 0.2	1.5E-04 ± 5.47E-05	90%
		500	12877 ± 290.2	4.62 ± 0.09	1.5 ± 0.22	1.4E-04 ± 5.33E-05	92.4%
	Sobol sequences	200	4617 ± 94.84	4.96 ± 0.03	0.94 ± 0.16	1.4E-04 ± 5.33E-05	99.2%
		500	11770 ± 243.5	4.94 ± 0.03	1.24 ± 0.19	1.9E-04 ± 6.51E-05	98.8%
$F4$	System-supplied	200	5014 ± 107.4	4.72 ± 0.06	1.74 ± 0.22	0.21 ± 0.022	94.4%
		500	12575 ± 315.8	4.6 ± 0.08	1.8 ± 0.25	0.2 ± 0.018	92%
	Sobol sequences	200	4755 ± 118.6	4.94 ± 0.03	1.4 ± 0.23	0.21 ± 0.21	98.8%
		500	11609 ± 262.9	4.98 ± 0.02	1.22 ± 0.19	0.21 ± 0.02	99.6%

Table 6.12: Sequential vector-based PSO results for the Himmelblau function

Random number generator	Iterations	Average evaluations	Average ♯ solutions	Average ♯ duplicates	Average $f'(x)$	Average $f'(y)$	Success rate
System-supplied	200	8764 ± 151	3.98 ± 0.02	7.78 ± 0.38	5.4E-04 $\pm$ 1.2E-03	9.1E-04 $\pm$ 2.2E-03	99.5%
	500	21438 ± 374	3.96 ± 0.03	7.5 ± 0.41	1.4E-04 $\pm$ 5.1E-04	3.1E-04 $\pm$ 5.1E-04	99%
Sobol sequences	200	8910 ± 219	4 ± 0	7.88 ± 0.53	9.8E-03 $\pm$ 0.03	7.2E-03 $\pm$ 0.02	100%
	500	21891 ± 473	3.98 ± 0.02	7.96 ± 0.47	1E-04 $\pm$ 8.55E-06	1.7E-04 $\pm$ 1.89E-05	99.5%

Table 6.13: Results of the Mann-Whitney U test. Success rates of the sequential VBPSO using a system-supplied random number generator are compared to those using Sobol sequences.

Function	Iterations	ΣR_1	ΣR_2	Significance level	Z	H_0/H_1
$F1$	200	2.83E+03	2.23E+03	0.05 0.01	-2.068	H_1 H_0
	500	2.78E+03	2.28E+03	0.05 0.01	-1.72	H_0 H_0
$F2$	200	2.88E+03	2.17E+03	0.05 0.01	-2.42	H_1 H_0
	500	2.78E+03	2.27E+03	0.05 0.01	-1.73	H_0 H_0
$F3$	200	3.00E+03	2.05E+03	0.05 0.01	-3.30	H_1 H_0
	500	2.88E+03	2.17E+03	0.05 0.01	-2.42	H_0 H_0
$F4$	200	2.80E+03	2.25E+03	0.05 0.01	-1.90	H_1 H_0
	500	2.95E+03	2.10E+03	0.05 0.01	-2.94	H_0 H_0
Himmelblau	200	2.80E+03	2.25E+03	0.05 0.01	-1.90	H_1 H_0
	500	2.95E+03	2.10E+03	0.05 0.01	-2.94	H_0 H_0

6.4 Results for the parallel and enhanced parallel vector-based PSO

This section presents results for the parallel and enhanced parallel vector-based PSO tested on all the benchmark functions described in section 6.2.3. Section 6.4.1 presents results where Sobol sequences as well as a system-supplied random number generator were used to initialize the swarm, and for 200 as well as 500 iterations. These results are compared to determine the best approach for initializing the swarm and the preferred number of iterations. Since both parallel algorithms incorporate a merging process, observations of the course of merging for both algorithms on all the functions are reported in section 6.4.2. To determine the best parallel algorithm, section 6.4.3 presents a comparison of the success rates of the algorithms.

6.4.1 Test results

Tables 6.14 to 6.17 present results of the performance of the parallel and enhanced parallel vector-based PSO algorithms for the one-dimensional functions $F1$ to $F4$. Tables 6.20 to 6.26 report results of the parallel algorithms for the following two-dimensional functions: Himmelblau, Griewank, Rastrigin, Ackley, Ursem $F1$, Ursem $F3$, and the six hump camel function.

All tables list the the average number of function evaluations and the average number of solutions over 50 runs together with the standard error, the average of the derivatives of the objective function, $f'(x_i)$, at positions where optima are located, and the success rate. The success rate is the total number of optima as a percentage of the total number of possible optima.

The outcome of the Mann-Whitney U test to determine statistically significant differences between experiments using Sobol sequences and a system-supplied random number generator are given in Tables 6.18, 6.19, 6.27, and 6.28. These outcomes are presented for both the parallel and enhanced parallel vector-based PSO algorithms.

Discussion

From the reported results and statistical data a number of observations were made. However, owing to the stochastic nature of PSO, results obtained from a number of independent runs of such algorithms are not deterministic. Therefore, small differences cannot form the basis of general conclusions on the performance of different algorithms.

One purpose of these experiments was to ascertain the performance of the vector-based algorithms on functions with varying landscapes with regard to the shapes, sizes, and placing of the niches. Results reported showed that the vector-based PSO algorithms performed well on functions where the shapes and distribution of the niches are relatively symmetrical, like the Himmelblau, Rastrigin, Griewank and Ursem F1 functions. The Ackley, Ursem F3 and six hump camel functions do not exhibit these characteristics, and performance degradation could be expected. However, although slightly worse, the average success rate of all functions was still above 98% for the parallel vector-based PSO and above 99% for the enhanced parallel version. Therefore, it can be concluded that both versions of the VBPSO that were tested in this section, are robust niching algorithms that performs well even in adverse circumstances.

The experiments were also designed to assess the influence of different approaches to initialization of particles. According to Table 6.35, algorithms using Sobol sequences as a random number generator had better success rates than those using system-supplied random number generators in all instances. Tables 6.27 and 6.28 report results of the Mann-Whitney U test comparing these two approaches to initialize particles. Rankings for algorithms using Sobol sequences (ΣR_1) were better than or equal to rankings for algorithms using system-supplied random number generators (ΣR_2). However, the Himmelblau, Griewank, Rastrigin and Ursem F1 functions showed no significant difference between algorithms using the two random number generators, that is, the null hypothesis was supported. Using the parallel vector-based PSO and 200 iterations, the alternative hypothesis was supported for the Ursem F3, Ackley and six hump camel functions at a significance level of 0.05, and for the six hump camel function at a significance level of 0.01. In the case of the enhanced parallel vector-based PSO, the alternative hypothesis was only supported for the 500 iterations version of the Ackley function at a significance level of 0.05. The function landscapes of these functions are less symmetrical than those of the other functions. For such function landscapes, an even distribution of particles becomes more important, which is provided by Sobol sequences.

Experiments conducted with 500 iterations showed smaller derivatives than experiments with 200 iterations. Although the number of iterations had no noticeable effect on the success rate, better quality solutions were obtained with more iterations.

From the above observations it can be concluded that algorithms using Sobol sequences to distribute particles evenly throughout the search space, perform better than those using a system-supplied random number generator. A higher number of iterations ensure better quality solutions. For this study, 500 iterations yielded an adequate level of accuracy. Hence, algorithms using Sobol sequences and executing 500 iterations will be used for all further work.

Table 6.14: Parallel vector-based PSO results for one-dimensional functions $F1$ and $F2$

	Random number generator	Iterations	Average function evaluations	Average ♯ solutions	Average derivation	Success rate
$F1$	System-supplied	200	5234 ± 129	4.86 ± 0.05	6.09E-07 $\pm$ 1.34E-06	97.2%
		500	13028 ± 335	4.74 ± 0.08	2.88E-14 $\pm$ 2.34E-15	94.8%
	Sobol sequences	200	4763 ± 82	5 ± 0	5.33E-06 $\pm$ 1.33E-05	100%
		500	12253 ± 260	5 ± 0	2.88E-14 $\pm$ 2.32E-15	100%
$F2$	System-supplied	200	5327 ± 120	4.74 ± 0.07	0.11 ± 0.07	94.8%
		500	12776 ± 318	4.88 ± 0.05	9.5E-02 $\pm$ 0.07	97.6%
	Sobol sequences	200	5181 ± 121	5 ± 0	0.10 ± 0.07	100%
		500	12696 ± 258	4.98 ± 0.02	9.7E-02 $\pm$ 0.07	99.6%

Table 6.15: Parallel vector-based PSO results for one-dimensional functions $F3$ and $F4$

	Random number generator	Iterations	Average function evaluations	Average ♯ solutions	Average derivation	Success rate
$F3$	System-supplied	200	5290 ± 112	4.94 ± 0.03	1.3E-02 ± 0.04	98.8%
		500	13249 ± 316	4.84 ± 0.05	4.8E-04 ± 7.0E-04	96.8%
	Sobol sequences	200	5100 ± 94	5 ± 0	2E-03 ± 5E-03	100%
		500	12273 ± 265	4.98 ± 0.02	1.6E-04 ± 5.59E-05	99.6%
$F4$	System-supplied	200	5400 ± 140	4.8 ± 0.06	0.2 ± 0.02	96%
		500	13078 ± 287	4.84 ± 0.05	0.21 ± 0.02	96.8%
	Sobol sequences	200	5181 ± 120	4.94 ± 0.3	0.2 ± 0.02	98.8%
		500	12314 ± 293	4.94 ± 0.03	4.5E-02 ± 0.02	98.8%

Table 6.16: Enhanced parallel vector-based PSO results for one-dimensional functions $F1$ and $F2$

	Random number generator	Iterations	Average function evaluations	Average ♯ solutions	Average derivation	Success rate
$F1$	System-supplied	200	5880 ± 138	4.8 ± 0.06	2.84E-14 ± 2.32E-15	96%
		500	13769 ± 347	4.48 ± 0.05	2.77E-14 ± 2.87E-15	96.8%
	Sobol sequences	200	5490 ± 137	5 ± 0	2.78E-14 ± 2.85E-15	100%
		500	13153 ± 281	5 ± 0	2.87E-14 ± 2.33E-15	100%
$F2$	System-supplied	200	5905 ± 163	4.84 ± 0.07	0.11 ± 0.07	96.8%
		500	14071 ± 323	4.76 ± 0.06	8.7E-02 ± 0.08	95.2%
	Sobol sequences	200	5348 ± 122	4.98 ± 0.02	9.7E-02 ± 0.07	99.6%
		500	12733 ± 242	5 ± 0	0.10 ± 0.07	100%

Table 6.17: Enhanced parallel vector-based PSO results for one-dimensional functions $F3$ and $F4$

	Random number generator	Iterations	Average function evaluations	Average ♯ solutions	Average derivation	Success rate
$F3$	System-supplied	200	5609 ± 137	4.8 ± 0.06	2.7E-04 $\pm$ 2.3E-04	96%
		500	13225 ± 310	4.7 ± 0.08	3.8E-04 $\pm$ 7.34E-05	94%
	Sobol sequences	200	5369 ± 109	5 ± 0	3.8E-04 $\pm$ 5.4E-04	100%
		500	12955 ± 291	5 ± 0	3.9E-04 $\pm$ 7.47E-05	100%
$F4$	System-supplied	200	5707 ± 140	4.74 ± 0.07	0.21 ± 0.02	94.8%
		500	13651 ± 318	4.6 ± 0.09	0.21 ± 0.02	92%
	Sobol sequences	200	5382 ± 115	4.92 ± 0.04	0.20 ± 0.02	98.4%
		500	12992 ± 258	4.96 ± 0.03	0.20 ± 0.02	99.2%

Table 6.18: Summary of ranks based on the Mann-Whitney U test for comparing success rates of algorithms using a system-supplied random number generator to those using Sobol sequences

Function	Iterations	Parallel VBPSO		Enhanced parallel VBPSO	
		ΣR_1	ΣR_2	ΣR_1	ΣR_1
$F1$	200	2.70E+03	2.35E+03	2.75E+03	2.30E+03
	500	2.78E+03	2.28E+03	2.73E+03	2.33E+03
$F2$	200	2.78E+03	2.27E+03	2.65E+03	2.40E+03
	500	2.78E+03	2.27E+03	2.83E+03	2.23E+03
$F3$	200	2.60E+03	2.45E+03	2.75E+03	2.30E+03
	500	2.70E+03	2.35E+03	2.83E+03	2.23E+03
$F4$	200	2.68E+03	2.37E+03	2.73E+03	2.32E+03
	500	2.68E+03	2.37E+03	2.88E+03	2.17E+03

Table 6.19: Results of the Mann-Whitney U test. Success rates of the parallel and enhanced parallel VBPSO algorithms using a system-supplied random number generator are compared to those using Sobol sequences

Function	Iterations	Significance level	Parallel VBPSO		Enhanced parallel VBPSO	
			Z	H_0/H_1	Z	H_0/H_1
$F1$	200	0.05	-1.21	H_0	-1.55	H_0
		0.01		H_0		H_0
	500	0.05	-1.72	H_0	-1.38	H_0
		0.01		H_0		H_0
$F2$	200	0.05	-1.73	H_0	-8.69E-01	H_0
		0.01		H_0		H_0
	500	0.05	-1.73	H_0	-2.068	H_0
		0.01		H_0		H_0
$F3$	200	0.05	-5.17E-01	H_0	-1.55	H_0
		0.01		H_0		H_0
	500	0.05	-1.21	H_0	-2.068	H_0
		0.01		H_0		H_0
$F4$	200	0.05	-1.04	H_0	-1.39	H_0
		0.01		H_0		H_0
	500	0.05	-1.04	H_0	-2.43	H_0
		0.01		H_0		H_0

Table 6.20: Parallel and enhanced parallel vector-based PSO results for the Himmelblau function

	Random number generator	Iterations	Average evaluations	Average ♯ solutions	Average $f'(x)$	Average $f'(y)$	Success rate
Parallel VBPSO	System-supplied	200	9358 ± 238	4 ± 0	2.6e-04 ± 1.4E-04	2.6E-04 ± 1E-04	100%
		500	22062 ± 432	3.96 ± 0.03	1.3E-04 ± 4.73E-05	0.02 ± 0.04	99%
	Sobol sequences	200	9328 ± 170	4 ± 0	1.3E-04 ± 2.31E-05	1.8E-04 ± 2.17E-05	100%
		500	24074 ± 389	4 ± 0	1.1E-04 ± 8.67E-05	1.6E-04 ± 1.91E-05	100%
Enhanced parallel VBPSO	System-supplied	200	10709 ± 242	3.98 ± 0.02	1.3E-04 ± 1E-04	1.9E-04 ± 1.9E-04	99.5%
		500	24483 ± 486	3.98 ± 0.02	1.86E-05 ± 2.27E-06	1.57E-05 ± 2.07E-05	99.5%
	Sobol sequences	200	11222 ± 233	4 ± 0	7.64E-05 ± 4.22E-05	5.27E-05 ± 2.53E-05	100%
		500	25310 ± 513	4 ± 0	0.018 ± 0.04	0.013 ± 0.02	100%

Table 6.21: Parallel and enhanced parallel vector-based PSO results for Griewank function

	Random number generator	Iterations	Average evaluations	Average ♯ solutions	Average $f'(x)$	Average $f'(y)$	Success rate
Parallel VBPSO	System-supplied	200	11819 ± 180	5 ± 0	3.23E-05 ± 5.53E-05	1.7E-03 ± 3.9E-03	100%
		500	28561 ± 447	5 ± 0	9.00E-05 ± 2E-04	1.79E-06 ± 1.27E-07	100%
	Sobol sequences	200	11654 ± 170	5 ± 0	2.26E-06 ± 2.82E-07	2.20E-06 ± 5.13E-07	100%
		500	29462 ± 521	5 ± 0	2.12E-06 ± 1.48E-07	1.80E-06 ± 1.26E-07	100%
Enhanced parallel VBPSO	System-supplied	200	13590 ± 229	4.98 ± 0.02	2.29E-06 ± 2.18E-07	2.01E-06 ± 2.08E-07	99.6%
		500	31512 ± 506	4.96 ± 0.03	2.10E-06 ± 1.50E-07	1.79E-06 ± 1.27E-07	99.2%
	Sobol sequences	200	13853 ± 235	5 ± 0	3.22E-06 ± 1.75E-06	2.31E-06 ± 8.74E-07	100%
		500	31678 ± 593	5 ± 0	2.11E-06 ± 1.49E-07	1.79E-06 ± 1.27E-07	100%

Table 6.22: Parallel and enhanced parallel vector-based PSO results for Rastrigin function

	Random number generator	Iterations	Average evaluations	Average ♯ solutions	Average $f'(x)$	Average $f'(y)$	Success rate
Parallel VBPSO	System-supplied	200	17166 ± 296	8.98 ± 0.02	1.2E-04 $\pm$ 2.85E-05	1.4E-04 $\pm$ 8.17E-05	99.78%
		500	43313 ± 683	8.98 ± 0.02	0.11 ± 0.34	0.016 ± 0.47	99.78%
	Sobol sequences	200	17554 ± 281	9 ± 0	0.034$\pm$ 8.5E-03	8.7E-04 $\pm$ 2.3E-03	100%
		500	43644 ± 570	9 ± 0	9.58E-05 $\pm$ 9.58E-06	9.58E-05 $\pm$ 9.58E-06	100%
Enhanced parallel VBPSO	System-supplied	200	20800 ± 296	8.88 ± 0.05	1.3E-04 $\pm$ 3.16E-05	1.4E-04 $\pm$ 5.76E-05	98.67%
		500	48234 ± 604	8.88 ± 0.05	9.49E-05 $\pm$ 9.63E-06	9.53E-05 $\pm$ 9.62E-06	98.67%
	Sobol sequences	200	20973 ± 251	9 ± 0	1.2E-04 $\pm$ 1.71E-05	1.2E-04 $\pm$ 2.50E-05	100%
		500	48687 ± 634	9 ± 0	9.58E-05 $\pm$ 9.59E-06	9.58E-05 $\pm$ 9.59E-06	100%

Table 6.23: Parallel and enhanced parallel vector-based PSO results for Ackley function

	Random number generator	Iterations	Average evaluations	Average ♯ solutions	Average $f'(x)$	Average $f'(y)$	Success rate
Parallel VBPSO	System-supplied	200	20427 ± 241	8.44 ± 0.09	0.23 ± 0.09	0.22 ± 0.1	93.78%
		500	49787 ± 782	8.52 ± 0.1	0.21 ± 0.09	0.22 ± 0.09	94.67%
	Sobol sequences	200	20309 ± 272	8.76 ± 0.07	0.22 ± 0.1	0.2 ± 0.09	97.33%
		500	52383 ± 752	8.6 ± 0.08	0.21 ± 0.09	0.22 ± 0.09	95.56%
Enhanced parallel VBPSO	System-supplied	200	21594 ± 233	8.84 ± 0.06	0.19 ± 0.09	0.22 ± 0.09	96.67%
		500	51212 ± 642	8.68 ± 0.08	0.2 ± 0.09	0.2 ± 0.09	96.44%
	Sobol sequences	200	21964 ± 324	9 ± 0	0.2 ± 0.09	0.19 ± 0.09	100%
		500	51824 ± 653	9 ± 0	0.2 ± 0.09	0.2 ± 0.09	100%

Table 6.24: Parallel and enhanced parallel vector-based PSO results for Ursem $F1$ function

	Random number generator	Iterations	Average evaluations	Average ♯ solutions	Average $f'(x)$	Average $f'(y)$	Success rate
Parallel VBPSO	System-supplied	200	9734 ± 212	2 ± 0	1.45E-05 ± 2.25E-07	4.36E-07 ± 2.11E-07	100%
		500	22766 ± 551	2 ± 0	1.42E-05 ± 7.02E-08	2.22E-08 ± 2.31E-09	100%
	Sobol sequences	200	9379 ± 212	2 ± 0	1.43E-05 ± 1.68E-07	1.46E-07 ± 5.75E-08	100%
		500	23339 ± 437	2 ± 0	1.42E-05 ± 7.02E-08	2.33E-08 ± 2.27E-09	100%
Enhanced parallel VBPSO	System-supplied	200	10348 ± 215	2 ± 0	1.6E-06 ± 1.97E-07	4.5E-07 ± 1.94E-07	100%
		500	24161 ± 554	2 ± 0	1.27E-06 ± 7.02E-08	0 ± 0	100%
	Sobol sequences	200	10473 ± 231	2 ± 0	1.47E-06 ± 1.74E-07	3.9E-07 ± 4.29E-07	100%
		500	25225 ± 577	2 ± 0	1.27E-06 ± 1.27E-06	0 ± 0	100%

Table 6.25: Parallel and enhanced parallel vector-based PSO results for Ursem $F3$ function

	Random number generator	Iterations	Average evaluations	Average ♯ solutions	Average $f'(x)$	Average $f'(y)$	Success rate
Parallel VBPSO	System-supplied	200	13947 ± 210	3.86 ± 0.08	0.51 ± 0.04	0.76 ± 0.04	98.5%
		500	33752 ± 494	3.98 ± 0.02	0.51 ± 0.04	0.76 ± 0.04	99.5%
	Sobol sequences	200	14162 ± 232	4 ± 0	0.51 ± 0.04	0.76 ± 0.04	100%
		500	35955 ± 394	3.98 ± 0.02	0.51 ± 0.04	0.76 ± 0.04	99.5%
Enhanced parallel VBPSO	System-supplied	200	16168 ± 244	4 ± 0	0.51 ± 0.04	0.76 ± 0.04	100%
		500	37698 ± 600	3.98 ± 0.02	0.51 ± 0.04	0.76 ± 0.04	99.5%
	Sobol sequences	200	16474 ± 255	4 ± 0	0.51 ± 0.04	0.76 ± 0.04	100%
		500	38935 ± 578	4 ± 0	0.51 ± 0.04	0.76 ± 0.04	100%

Table 6.26: Parallel and enhanced parallel vector-based PSO results for the six hump camel function

	Random number generator	Iterations	Average evaluations	Average ♯ solutions	Average $f'(x)$	Average $f'(y)$	Success rate
Parallel VBPSO	System-supplied	200	16470 ± 247	5.12 ± 0.1	3.56E-05 ± 4.32E-06	8.98E-06 ± 1.88E-06	85.33%
		500	40371 ± 584	5.5 ± 0.08	1.2E-03 ± 2.8E-03	6.94E-06 ± 1.08E-07	91.67%
	Sobol sequences	200	16377 ± 254	5.6 ± 0.07	6.5E-03 ± 0.02	3.7E-04 ± 8.4E-04	93.33%
		500	41649 ± 640	5.56 ± 0.08	3.54E-05 ± 3.83E-06	6.97E-06 ± 1.09E-07	92.67%
Enhanced parallel VBPSO	System-supplied	200	18690 ± 234	5.68 ± 0.07	2.3E-03 ± 5.4E-03	2.3E-04 ± 5.1E-04	94.67%
		500	41418 ± 568	5.76 ± 0.06	2.41E-06 ± 2.81E-07	6.47E-06 ± 3.55E-07	96%
	Sobol sequences	200	18450 ± 234	5.86 ± 0.05	5.4E-04 ± 1.3E-03	1.3E-03 ± 3.2E-03	97.67%
		500	42825 ± 580	5.96 ± 0.03	2.42E-06 ± 2.78E-07	6.4E-06 ± 3.59E-07	99.33%

Table 6.27: Summary of ranks based on the Mann-Whitney U test for comparing success rates of algorithms using a system-supplied random number generator to those using Sobol sequences

Function	Iterations	Parallel VBPSO		Enhanced parallel VBPSO	
		ΣR_1	ΣR_2	ΣR_1	ΣR_2
Himmelblau	200	2.53E+03	2.53E+03	2.55E+03	2.50E+03
	500	2.58E+03	2.48E+03	2.55E+03	2.50E+03
Griewank	200	2.53E+03	2.53E+03	2.55E+03	2.50E+03
	500	2.53E+03	2.53E+03	2.55E+03	2.50E+03
Rastrigin	200	2.55E+03	2.50E+03	2.68E+03	2.38E+03
	500	2.55E+03	2.50E+03	2.68E+03	2.38E+03
Ursem $F1$	200	2.53E+03	2.53E+03	2.53E+03	2.53E+03
	500	2.53E+03	2.53E+03	2.53E+03	2.53E+03
Ursem $F3$	200	2.60E+03	2.45E+03	2.53E+03	2.53E+03
	500	2.53E+03	2.53E+03	2.55E+03	2.50E+03
Ackley	200	2.86E+03	2.19E+03	2.70E+03	2.35E+03
	500	2.58E+03	2.47E+03	2.88E+03	2.18E+03
Six hump camel	200	2.98E+03	2.07E+03	2.73E+03	2.32E+03
	500	2.58E+03	2.47E+03	2.78E+03	2.28E+03

Table 6.28: Results of the Mann-Whitney U test. Success rates of the parallel and enhanced parallel VBPSO algorithms using a system-supplied random number generator are compared to those using Sobol sequences

Function	Iterations	Significance level	Parallel VBPSO		Enhanced parallel VBPSO	
			Z	H_0/H_1	Z	H_0/H_1
Himmelblau	200	0.05	0.00	H_0	-1.72E-01	H_0
		0.01		H_0		H_0
	500	0.05	-3.45E-01	H_0	-1.72E-01	H_0
		0.01		H_0		H_0
Griewank	200	0.05	0.00	H_0	-1.72E-01	H_0
		0.01		H_0		H_0
	500	0.05	0.00	H_0	-3.45E-01	H_0
		0.01		H_0		H_0
Rastrigin	200	0.05	-1.72E-01	H_0	-1.034	H_0
		0.01		H_0		H_0
	500	0.05	-1.72E-01	H_0	-1.034	H_0
		0.01		H_0		H_0
Ursem $F1$	200	0.05	0.00	H_0	0.00	H_0
		0.01		H_0		H_0
	500	0.05	0.00	H_0	0.00	H_0
		0.01		H_0		H_0
Ursem $F3$	200	0.05	-5.17E-01	H_1	0.00	H_0
		0.01		H_0		H_0
	500	0.05	0.00	H_0	-1.72E-01	H_0
		0.01		H_0		H_0
Ackley	200	0.05	-2.31	H_1	-1.21	H_0
		0.01		H_0		H_0
	500	0.05	-4.00E-01	H_0	-2.41	H_1
		0.01		H_0		H_0
Six hump camel	200	0.05	-3.14	H_1	-1.40	H_0
		0.01		H_1		H_0
	500	0.05	-4.10E-01	H_0	-1.72	H_0
		0.01		H_0		H_0

6.4.2 Tracking merging of niches

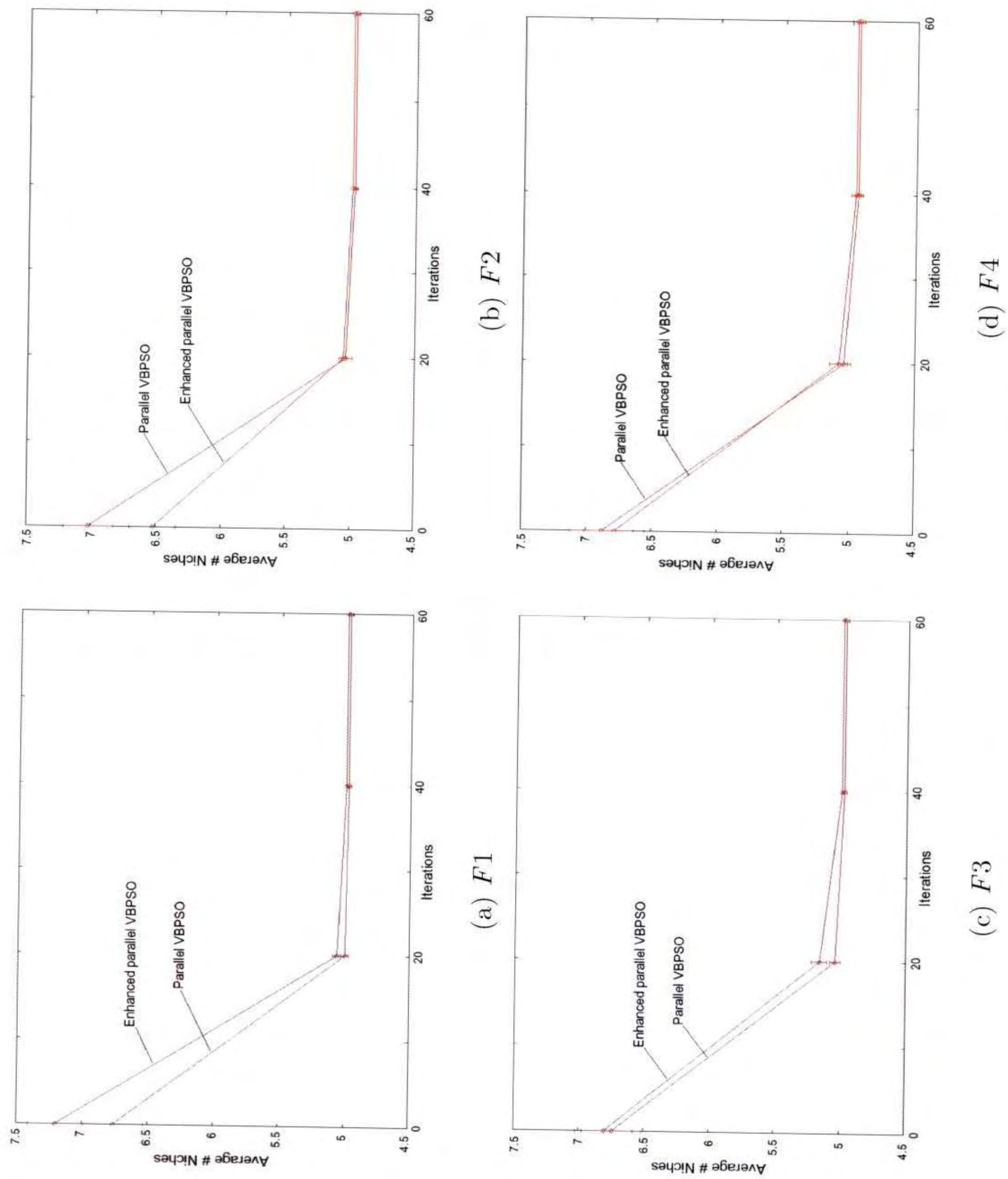
The parallel and enhanced parallel VBPSO incorporate a merging procedure to merge subswarms that converge on the same optimum. To observe the merging process, the number of niches was recorded after establishing the initial niches, as well as after every following 20 iterations. Table 6.29 summarizes these results for functions $F1$ to $F4$. Tables 6.30, 6.31, 6.32 and 6.33 present results of the niche merging process for seven two-dimensional functions, for the parallel VBPSO as well as the enhanced parallel vector-based PSO. According to the decision reached in the previous section, Sobol sequences were used to distribute particles uniformly over the search space, and the number of iterations was set to 500. Figures 6.9, 6.10 and 6.11 show graphs of the number of niches plotted against the number of iterations. For each function, graphs of the parallel and the enhanced parallel vector-based PSO are displayed on the same axes.

Results show that the final number of niches were established early during a run. For the one-dimensional functions, all niches have merged after 40 iterations, while the number of iterations differ for the two-dimensional functions. Variations in the number of initial niches can be ascribed to the stochastic nature of the PSO algorithm, as the parallel and enhanced parallel VBPSO versions identify niches in exactly the same way. Variations in merging time are in line with variations in the number of initial niches.

Tracking the number and positions of niches while vector-based PSO is converging, yielded a number of interesting observations. For functions $F1$, $F2$ and $F4$, niches merged faster when the enhanced parallel VBPSO was used, while for function $F3$, niches merged faster when the parallel version of the algorithm was used. For all two-dimensional functions, niches merged faster when the enhanced parallel VBPSO was used. For the Griewank, Ursem F1 and Rastrigin functions, the difference was negligible. The difference between the merging behaviour of the two algorithms was more pronounced in the case of the Himmelblau, Ackley, Ursem F3 and six hump camel functions where niches are more asymmetrically shaped. Therefore, the strategy used by the enhanced parallel VBPSO to contain particles in niches during optimization, resulted in faster merging, as well as a higher success rate. Two of the functions that used the parallel version of VBPSO, namely Ursem F3 and the six hump camel function, showed niches converging during the later stages of a run. These results showed that the parallel VBPSO may lose niches after all subswarms have converged to the required number of niches, while the enhanced parallel VBPSO did not.

Table 6.29: Merging of niches for one-dimensional functions $F1$ to $F4$

Iterations	Average number of niches							
	Parallel VBPSO				Enhanced parallel VBPSO			
	F1	F2	F3	F4	F1	F2	F3	F4
0	6.76 ± 0.23	7.02 ± 0.19	6.74 ± 0.23	6.88 ± 0.24	7.2 ± 0.21	6.52 ± 0.18	6.8 ± 0.22	6.78 ± 0.23
20	5 ± 0.03	5.04 ± 0.05	5.04 ± 0.02	5.04 ± 0.05	5.06 ± 0.03	5.06 ± 0.03	5.16 ± 0.06	5.08 ± 0.07
40	4.98 ± 0.02	4.98 ± 0.02	4.98 ± 0.02	4.94 ± 0.03	5 ± 0	5 ± 0	5 ± 0	4.96 ± 0.04
60	4.98 ± 0.02	4.98 ± 0.02	4.98 ± 0.02	4.94 ± 0.03	5 ± 0	5 ± 0	5 ± 0	4.96 ± 0.04



(a) $F1$ (b) $F2$ (c) $F3$ (d) $F4$

Figure 6.9: Merging of one-dimensional functions

Table 6.30: Merging of niches using the parallel VBPSO - part 1

Iterations	Average number of niches			
	Himmelblau	Griewank	Rastrigin	Ackley
0	13.68 ± 0.38	14.76 ± 0.46	23.58 ± 0.54	31.06 ± 0.63
20	9.26 ± 0.46	10.5 ± 0.26	15.94 ± 0.35	12.26 ± 0.23
40	5.44 ± 0.17	6.28 ± 0.16	10.42 ± 0.15	9.1 ± 0.11
60	4.42 ± 0.11	5.7 ± 0.12	9.36 ± 0.09	8.84 ± 0.08
80	4.16 ± 0.07	5.54 ± 0.1	9.26 ± 0.08	8.72 ± 0.08
100	4.14 ± 0.07	5.54 ± 0.1	9.12 ± 0.05	8.72 ± 0.08
120	4.08 ± 0.05	5.48 ± 0.1	9.02 ± 0.02	8.68 ± 0.07
140	4.06 ± 0.03	5.48 ± 0.1	9.02 ± 0.02	8.68 ± 0.07
160	4.06 ± 0.03	5.38 ± 0.1	9.02 ± 0.02	8.68 ± 0.07
180	4.04 ± 0.03	5.2 ± 0.07	9 ± 0	8.68 ± 0.07
200	4.04 ± 0.03	5.16 ± 0.07	9 ± 0	8.68 ± 0.07
220	4 ± 0	5.06 ± 0.04	9 ± 0	8.68 ± 0.07
240	4 ± 0	5.02 ± 0.02	9 ± 0	8.66 ± 0.07
260	4 ± 0	5 ± 0	9 ± 0	8.66 ± 0.07
280	4 ± 0	5 ± 0	9 ± 0	8.66 ± 0.07
300	4 ± 0	5 ± 0	9 ± 0	8.66 ± 0.07
320	4 ± 0	5 ± 0	9 ± 0	8.66 ± 0.07
340	4 ± 0	5 ± 0	9 ± 0	8.66 ± 0.07
360	4 ± 0	5 ± 0	9 ± 0	8.66 ± 0.07
380	4 ± 0	5 ± 0	9 ± 0	8.66 ± 0.07
400	4 ± 0	5 ± 0	9 ± 0	8.66 ± 0.07
420	4 ± 0	5 ± 0	9 ± 0	8.66 ± 0.07
440	4 ± 0	5 ± 0	9 ± 0	8.66 ± 0.07
460	4 ± 0	5 ± 0	9 ± 0	8.66 ± 0.07
480	4 ± 0	5 ± 0	9 ± 0	8.66 ± 0.07
500	4 ± 0	5 ± 0	9 ± 0	8.66 ± 0.07

Table 6.31: Merging of niches using the parallel VBPSO - part 2

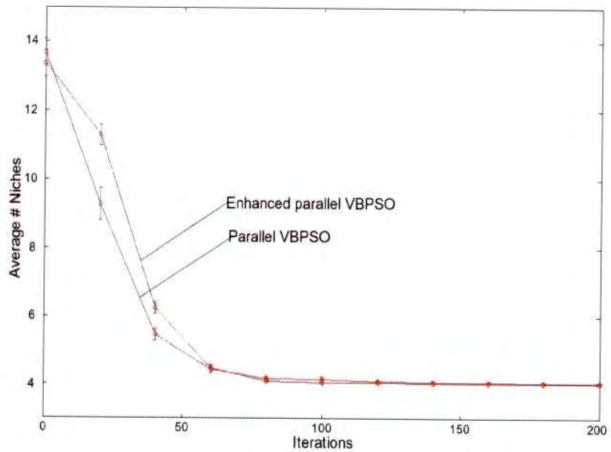
Iterations	Average number of niches		
	Ursem $F1$	Ursem $F3$	Six hump camel
0	11.24 ± 0.45	21.86 ± 0.37	24.18 ± 0.56
20	2.98 ± 0.15	11.26 ± 0.25	9.6 ± 0.18
40	2.1 ± 0.04	7.06 ± 0.22	6.3 ± 0.11
60	2 ± 0	5.52 ± 0.16	5.76 ± 0.08
80	2 ± 0	4.78 ± 0.12	5.64 ± 0.08
100	2 ± 0	4.38 ± 0.11	5.6 ± 0.08
120	2 ± 0	4.24 ± 0.07	5.6 ± 0.08
140	2 ± 0	4.16 ± 0.06	5.6 ± 0.08
160	2 ± 0	4.1 ± 0.05	5.6 ± 0.08
180	2 ± 0	4.1 ± 0.05	5.58 ± 0.08
200	2 ± 0	4.06 ± 0.03	5.58 ± 0.08
220	2 ± 0	4.04 ± 0.03	5.58 ± 0.08
240	2 ± 0	4.02 ± 0.02	5.58 ± 0.08
260	2 ± 0	4.02 ± 0.02	5.58 ± 0.08
280	2 ± 0	4.02 ± 0.02	5.58 ± 0.08
300	2 ± 0	4 ± 0	5.58 ± 0.08
320	2 ± 0	4 ± 0	5.58 ± 0.08
340	2 ± 0	4 ± 0	5.58 ± 0.08
360	2 ± 0	4 ± 0	5.58 ± 0.08
380	2 ± 0	4 ± 0	5.58 ± 0.08
400	2 ± 0	4 ± 0	5.58 ± 0.08
420	2 ± 0	3.98 ± 0.02	5.58 ± 0.08
440	2 ± 0	3.98 ± 0.02	5.58 ± 0.08
460	2 ± 0	3.98 ± 0.02	5.58 ± 0.08
480	2 ± 0	3.98 ± 0.02	5.56 ± 0.08
500	2 ± 0	3.98 ± 0.02	5.56 ± 0.08

Table 6.32: Merging of niches using the enhanced parallel VBPSO - part 1

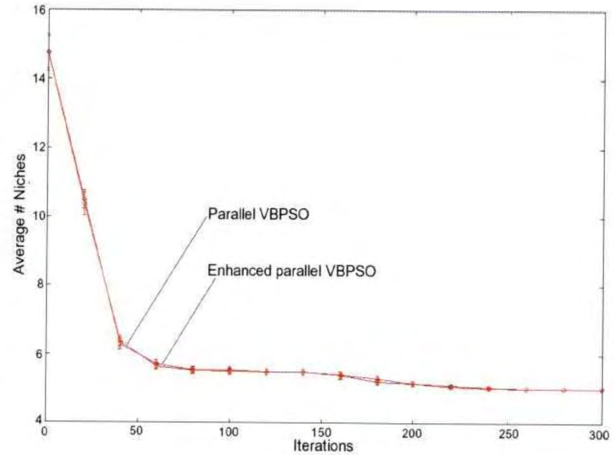
Iterations	Average number of niches			
	Himmelblau	Griewank	Rastrigin	Ackley
0	13.36 ± 0.39	14.74 ± 0.53	25.68 ± 0.52	27.56 ± 0.56
20	11.28 ± 0.3	10.36 ± 0.33	14.52 ± 0.27	11.76 ± 0.2
40	6.22 ± 0.16	6.36 ± 0.16	9.5 ± 0.1	9.14 ± 0.05
60	4.46 ± 0.09	5.64 ± 0.11	9.02 ± 0.02	9.06 ± 0.03
80	4.08 ± 0.04	5.52 ± 0.1	9 ± 0	$9. \pm 0$
100	4.04 ± 0.03	5.5 ± 0.1	9 ± 0	9 ± 0
120	4.04 ± 0.03	5.48 ± 0.1	9 ± 0	9 ± 0
140	4.02 ± 0.02	5.48 ± 0.1	9 ± 0	9 ± 0
160	4.02 ± 0.02	5.4 ± 0.1	9 ± 0	9 ± 0
180	4.02 ± 0.02	5.3 ± 0.08	9 ± 0	9 ± 0
200	4.02 ± 0.02	5.16 ± 0.07	9 ± 0	9 ± 0
220	4.02 ± 0.02	5.1 ± 0.05	9 ± 0	9 ± 0
240	4.02 ± 0.02	5.04 ± 0.04	9 ± 0	9 ± 0
260	4.02 ± 0.02	5 ± 0	9 ± 0	9 ± 0
280	4.02 ± 0.02	5 ± 0	9 ± 0	9 ± 0
300	4 ± 0	5 ± 0	9 ± 0	9 ± 0
320	4 ± 0	5 ± 0	9 ± 0	9 ± 0
340	4 ± 0	5 ± 0	9 ± 0	9 ± 0
360	4 ± 0	5 ± 0	9 ± 0	9 ± 0
380	4 ± 0	5 ± 0	9 ± 0	9 ± 0
400	4 ± 0	5 ± 0	9 ± 0	9 ± 0
420	4 ± 0	5 ± 0	9 ± 0	9 ± 0
440	4 ± 0	5 ± 0	9 ± 0	9 ± 0
460	4 ± 0	5 ± 0	9 ± 0	9 ± 0
480	4 ± 0	5 ± 0	9 ± 0	9 ± 0
500	4 ± 0	5 ± 0	9 ± 0	9 ± 0

Table 6.33: Merging of niches using the enhanced parallel VBPSO - part 2

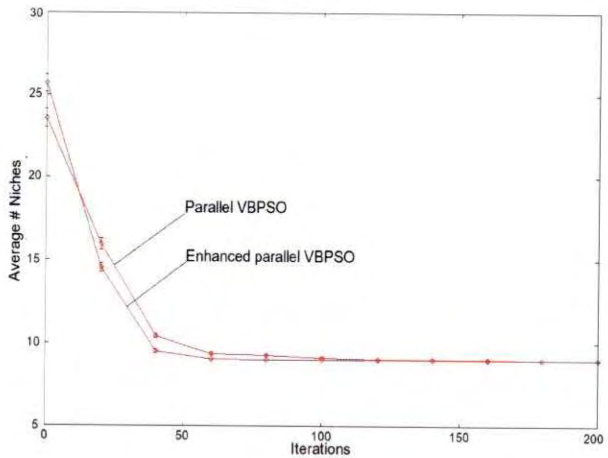
Iterations	Average number of niches		
	Ursem $F1$	Ursem $F3$	Six hump camel
0	11.54 ± 0.53	22.42 ± 0.42	22.56 ± 0.52
20	3.16 ± 0.27	11.16 ± 0.25	12.06 ± 0.31
40	2.06 ± 0.04	6.1 ± 0.17	7.22 ± 0.13
60	2 ± 0	4.44 ± 0.11	6.04 ± 0.05
80	2 ± 0	4.1 ± 0.04	5.96 ± 0.03
100	2 ± 0	4.08 ± 0.04	5.96 ± 0.03
120	2 ± 0	4.02 ± 0.02	5.96 ± 0.03
140	2 ± 0	4 ± 0	5.96 ± 0.03
160	2 ± 0	4 ± 0	5.96 ± 0.03
180	2 ± 0	4 ± 0	5.96 ± 0.03
200	2 ± 0	4 ± 0	5.96 ± 0.03
220	2 ± 0	4 ± 0	5.96 ± 0.03
240	2 ± 0	4 ± 0	5.96 ± 0.03
260	2 ± 0	4 ± 0	5.96 ± 0.03
280	2 ± 0	4 ± 0	5.96 ± 0.03
300	2 ± 0	4 ± 0	5.96 ± 0.03
320	2 ± 0	4 ± 0	5.96 ± 0.03
340	2 ± 0	4 ± 0	5.96 ± 0.03
360	2 ± 0	4 ± 0	5.96 ± 0.03
380	2 ± 0	4 ± 0	5.96 ± 0.03
400	2 ± 0	4 ± 0	5.96 ± 0.03
420	2 ± 0	4 ± 0	5.96 ± 0.03
440	2 ± 0	4 ± 0	5.96 ± 0.03
460	2 ± 0	4 ± 0	5.96 ± 0.03
480	2 ± 0	4 ± 0	5.96 ± 0.03
500	2 ± 0	4 ± 0	5.96 ± 0.03



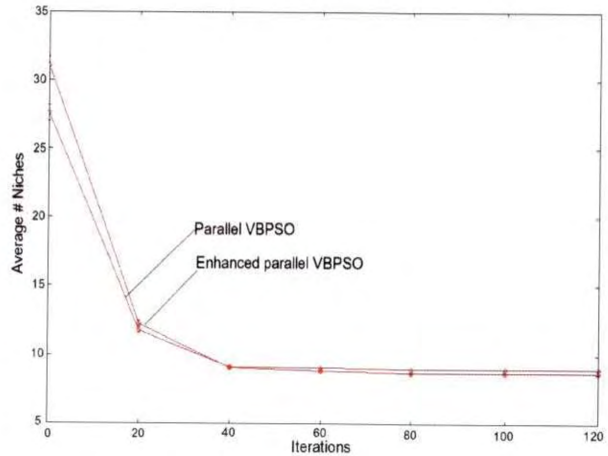
(a) Himmelblau



(b) Griewank

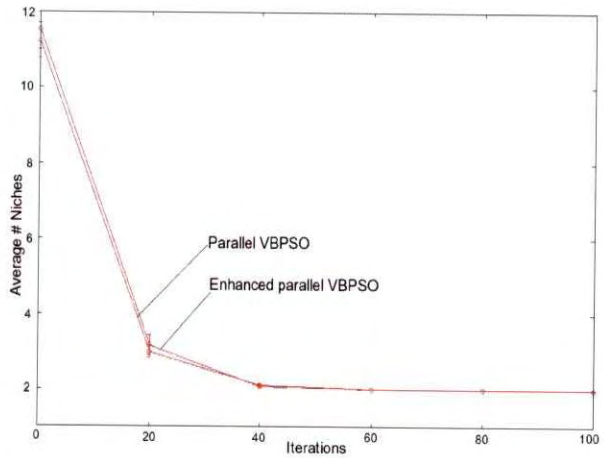


(c) Rastrigin

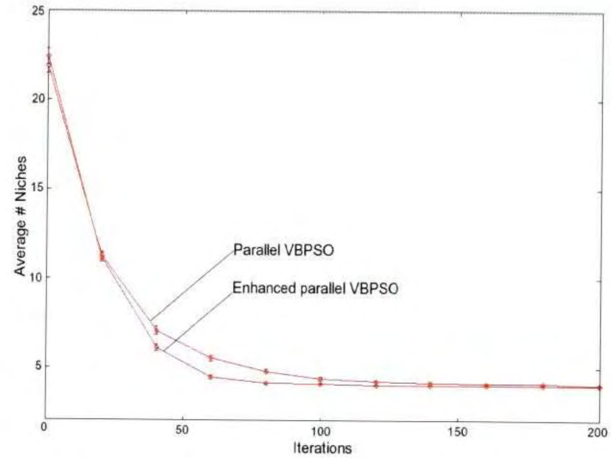


(d) Ackley

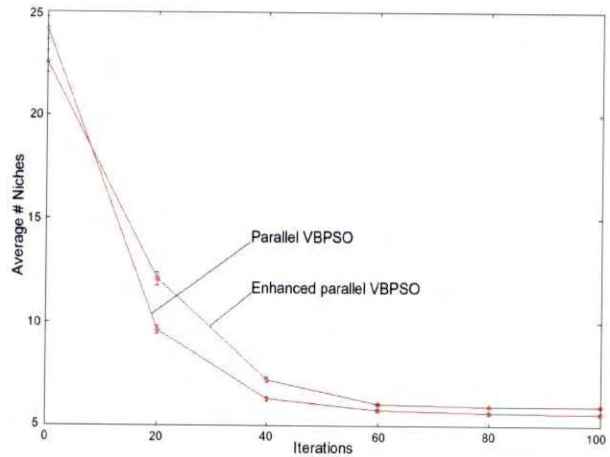
Figure 6.10: Merging of two-dimensional functions - part 1



(e) Ursem $F1$



(f) Ursem $F3$



(g) Six Hump Camel

Figure 6.11: Merging of two-dimensional functions - part 2

6.4.3 Comparing the parallel and enhanced parallel algorithms

This section compares the performance of the parallel vector-based PSO and the enhanced parallel vector-based PSO. Test results of the various functions were presented in section 6.3.1. To compare the algorithms, these results are summarized in Tables 6.34 and 6.35.

The parallel vector-based PSO was compared to the enhanced vector-based PSO for all the functions tested in section 6.4. Table 6.34 shows exactly the same outcomes for parallel and enhanced parallel VBPSO. Although individual differences occur, Table 6.35 shows that the enhanced parallel VBPSO performed better than the parallel VBPSO. Results of the Mann-Whitney U test reported in Table 6.36 show no significant difference between the performances of the algorithms for functions $F1$ to $F4$. Table 6.37 shows a significant difference between the algorithms only in the case of the Ackley and six-hump camel functions. These functions have a number of small niches that may easily merge with larger niches if particles leave the niche. The enhanced parallel VBPSO contains a strategy to prevent such occurrences and consequently performs better for such functions.

The development of the family of vector-based particle swarm optimizers culminates in the enhanced parallel vector-based PSO using Sobol sequences as a random number generator. Hence, the latter is used for all further work and will be referred to as the vector-based PSO (VBPSO).

Table 6.34: Average % optima located for functions $F1$ to $F4$

Function	Parallel VBPSO		Enhanced parallel VBPSO	
	Average ♯ solutions	Success rate	Average ♯ solutions	Success rate
$F1$	5 ± 0	100%	5 ± 0	100%
$F2$	4.98 ± 0.02	99.6%	5 ± 0	100%
$F3$	4.98 ± 0.02	99.6%	5 ± 0	100%
$F4$	4.94 ± 0.03	98.8%	4.96 ± 0.03	99.2%
Average	4.98	99.5%	4.99	99.8%

Table 6.35: Summary of VBPSO success rates for two-dimensional functions

Function	Parallel VBPSO	Enhanced parallel VBPSO
Himmelblau	100%	100%
Griewank	100%	100%
Rastrigin	100%	100%
Ackley	96.45%	100%
Ursem F1	100%	100%
Ursem F3	99.75%	100%
Six hump camel	93%	98.5%
Average	98.46%	99.79%

Table 6.36: Comparing algorithms for functions $F1$ to $F4$

Function	Significance level	Enhanced parallel vs parallel VBPSO			
		Rank 1	Rank 2	Z	H_0/H_1
$F1$	0.05	2.53E+03	2.53E+03	0.00	H_0
	0.01				H_0
$F2$	0.05	2.55E+03	2.50E+03	-1.72E-01	H_0
	0.01				H_0
$F3$	0.05	2.55E+03	2.50E+03	-1.72E-01	H_0
	0.01				H_0
$F4$	0.05	2.55E+03	2.50E+03	-1.72E-01	H_0
	0.01				H_0

Table 6.37: Comparing the enhanced parallel VBPSO to the parallel VBPSO

Function	Significance	Enhanced parallel VBPSO versus parallel VBPSO			
		ΣR_1	ΣR_2	Z	H_0/H_1
Himmelblau	0.05	2.53E+03	2.53E+03	0.00	H_0
	0.01				H_0
Griewank	0.05	2.53E+03	2.53E+03	0.00	H_0
	0.01				H_0
Rastrigin	0.05	2.53E+03	2.53E+03	0.00	H_0
	0.01				H_0
Ursem $F1$	0.05	2.53E+03	2.53E+03	0.00	H_0
	0.01				H_0
Ursem $F3$	0.05	2.55E+03	2.50E+03	-1.72E-01	H_0
	0.01				H_0
Ackley	0.05	2.98E+03	2.08E+03	-3.10E+00	H_1
	0.01				H_1
Six hump camel	0.05	3.00E+03	2.50E+03	-3.28E+00	H_1
	0.01				H_1

6.5 Analysis of the vector-based particle swarm optimizer

This section presents further results of experiments with the vector-based particle swarm optimizer. Only the enhanced version of the algorithm, using Sobol sequences to distribute initial particles evenly across the search space and executing 500 iterations, has been used, since it showed to produce the best results. This algorithm will now be referred to as the vector-based particle swarm optimizer (VBPSO).

Three aspects of the VBPSO have been investigated:

- The behaviour of the VBPSO on more complicated function landscapes.
- Sensitivity of the vector-based PSO to changes in the granularity parameter.
- Scalability of the vector-based PSO and the relationship between the initial swarm size and the number of solutions.

6.5.1 Analysis of the VBPSO on additional functions

This subsection presents results of experiments with the VBPSO on a number of additional functions in demarcated regions of the search space. These functions were specifically selected to test the behaviour of the algorithm on more complicated function landscapes. Features of each function are discussed and results for specific swarm sizes and granularity settings are presented.

Taking the size of the search space, the number of optima that can be expected, and settings for previous experiments into account, granularity settings and initial swarm sizes were estimated for these experiments. These settings were not claimed to be optimal. The sensitivity of the vector-based PSO to granularity is discussed in section 6.5.2, and the scalability of the algorithm is discussed in section 6.5.3.

The Styblinksi-Tang function:

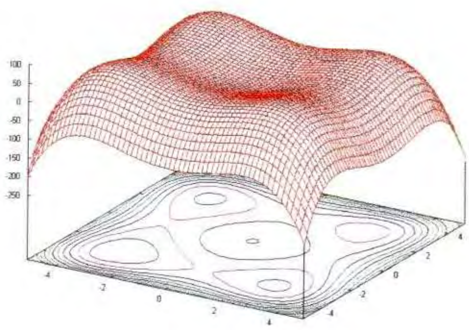


Figure 6.12: The Styblinski-Tang function showing maxima

The Styblinski-Tang function is defined as

$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{2}(x_i^4 - 16x_i^2 + 5x_i) \tag{6.15}$$

and illustrated in Figure 6.12 in the range $x_1, x_2 \in [-5.0, 5.0]$. The function is defined in two dimensions with one global minimum and three local optima of which the fitness differ slightly. Niches are flat and broad, and the function has been selected to test the ability of the algorithm to optimize such niches. Positions of optima and function values at these positions are presented in Table 6.38.

The function was tested with an initial swarm size of 30 and a granularity of 0.5. Results are presented in Table 6.40. Results show a 100% success rate for 50 independent runs of the algorithm, while the derivatives indicate high quality solutions. Therefore, the function was not tested with larger swarm sizes.

Table 6.38: Optima of the Styblinksi-Tang function

Locations of optima		Fitness
x_1	x_2	$f(x_1, x_2)$
-2.9	-2.9	-78.33
2.75	-2.9	-64.2
-2.9	2.75	-64.2
2.75	2.75	-50.06

The Bird function:

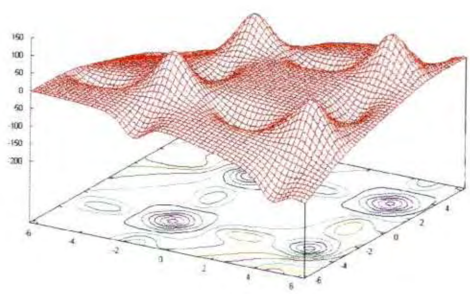


Figure 6.13: The Bird function showing maxima

The Bird function is defined as

$$f(\mathbf{x_1}, \mathbf{x_2}) = \sin(x_1)e^{(1-\cos(x_2))^2} + \cos(x_2)e^{(1-\sin(x_1))^2} + (x_1 - x_2)^2 \tag{6.16}$$

and illustrated in Figure 6.13 in the range $x_1, x_2 \in [-2\pi, 2\pi]$.

The Bird function has four well-defined minima as well as two very small depressions in the landscape. This function has been selected to test the ability of the algorithm to identify large (major solutions) as well as small niches (minor solutions) and locate the corresponding optima. Table 6.39 shows the positions and fitness of the optima. Table 6.41 presents results of the experiments, calculated separately for major and minor optima. The granularity was set to an estimated value of 30° or $\pi/6$.

To test the effect of different swarm sizes, experiments were run with initial swarm sizes of 30 and 50 particles. Results show that the algorithm had a 100% success rate for the four large niches for swarm sizes of 30 as well as 50 particles, while the minor optima were only located 76% of the time for an initial swarm size of 30 and 88% of the time when the swarm size was extended to 50 particles. These results confirm that prominent and well-defined optima are located easier, while the success rate for small niches increases for larger initial swarms sizes.

Table 6.39: Optima of the Bird function

Solution type	Locations of optima		Fitness
	x_1	x_2	$f(x_1, x_2)$
Major	-90.65	-179.35	-106.77
	269.35	180.65	-106.77
	-88.04	178.04	-87.31
	266.72	-176.72	-48.41
Minor	-321.88	-308.12	1.49
	38.12	51.88	1.49

Table 6.40: VBPSO results for Styblinksi-Tang function

♯ Particles	Granularity	Average ♯ evaluations	Average ♯ solutions	Average $f'(x)$	Average $f'(y)$	Success rate
30	0.5	24417 ± 461	4 ± 0	3.83E-06 ± 4.07E-07	3.83E-06 ± 4.07E-07	100%

Table 6.41: VBPSO results for the Bird function

♯ Particles	Granularity	Average ♯ evaluations	Optima	Average ♯ solutions	Average $f'(x)$	Average $f'(y)$	Success rate
30	$\pi/6$	29933 ± 499	Major	4 ± 0	1.44E-06 ± 1.46E-07	1.31E-06 ± 1.23E-07	100%
			Minor	1.52 ± 0.09	1.77E-08 ± 1.49E-09	1.94E-08 ± 1.77E-09	76%
50	$\pi/6$	47357 ± 644	Major	4 ± 0	1.30E-06 ± 1.19E-07	1.32E-06 ± 1.21E-07	100%
			Minor	1.76 ± 0.07	1.87E-08 ± 1.39E-09	2.68E-08 ± 1.27E-08	88%

The generalized Schwefel function 2.26:

The function is defined as:

$$f(\mathbf{x}) = -\sum_{i=1}^{2}(x_i \sin(\sqrt{|x_i|})) \tag{6.17}$$

Figure 6.14 illustrates the generalized Schwefel function 2.26 in the range $x_1, x_2 \in [-65, 100]$.

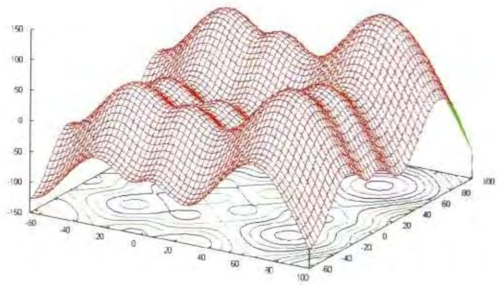


Figure 6.14: The generalized Schwefel function 2.26 showing maxima

The generalized Schwefel function 2.26 has nine optima of different heights in the given range. The four most prominent niches are situated near to the four corners of the defined search space, and are referred to as major solutions for the purposes of this experiment. The other five optima are referred to as minor solutions, as one of these is very small, while each of the other four is situated very near to a major solution, and would therefore be difficult to locate. Table 6.42 lists the positions and fitnesses of the optima.

Table 6.44 presents the results of the experiments. As in the case of the Bird function, the major solutions have a success rate of 100%, while the minor solutions are found 82.8% and 91.6% of the time for 50 and 80 particles respectively. For this function, partial derivatives could not be calculated, since the function, that contains an absolute value, is not continuous. However, all optimal positions were similar to the positions listed in Table 6.42 if compared based on two decimal positions.

Table 6.42: Optima of the generalized Schwefel function 2.26

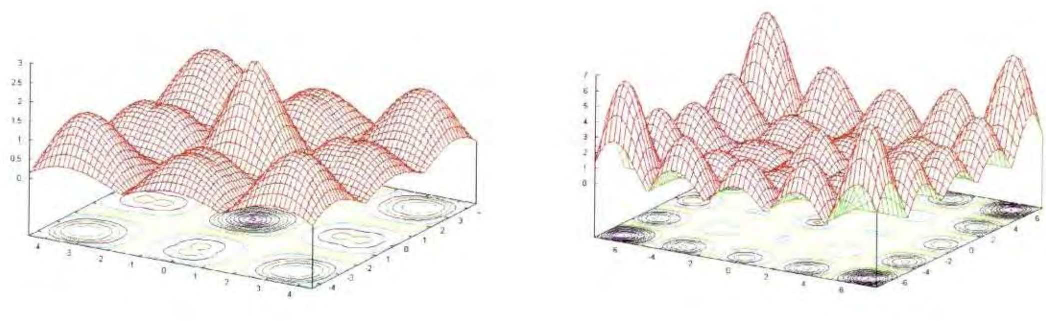
Solution type	Locations of optima		Fitness
	x_1	x_2	$f(x_1, x_2)$
Major	65.55	65.55	-127.7
	65.55	25.88	-87.72
	-25.88	65.55	-87.72
	-25.88	-25.88	-48.17
Minor	5.24	65.55	-67.58
	65.55	5.24	-67.58
	-25.88	5.24	-28.03
	5.24	-25.88	-28.03
	5.24	5.24	-7.89

The tabular holder function:

The function is defined as:

$$f(\mathbf{x_1}, \mathbf{x_2}) = -\left|\cos(x_1)\cos(x_2)e^{|1-((x_1^2+x_2^2)^{0.5})/\pi|}\right| \tag{6.18}$$

Figure 6.15 illustrates the tabular holder function in the range $x_1, x_2 \in [-4.5, 4.5]$ and in the range $x_1, x_2 \in [-7.5, 7.5]$.



(a) (b)

Figure 6.15: The tabular holder function

The tabular holder function has optima of different shapes and sizes, depending on the search space. Some of these peaks have two optima very near to one another as illustrated in Figure 6.15(a). Experiments have been run in the range $x_1, x_2 \in [-4.5, 4.5]$ to test the ability of the algorithm to locate optima situated very near to one another. For the purpose of this experiment, larger niches are referred to as major solutions, while niches that overlap to such an extent that they have the appearance of peaks with two optima, are referred to as minor solutions. This distinction is made because of the difficulty of locating optima that are very near to one another. Table 6.43 lists these optimal positions and their corresponding fitnesses. Table 6.45 presents the average number of major and minor solutions located with an initial swarm size of 60 particles. Because the function is not continuous, partial derivatives could not be obtained, but as in the case of the Schwefel function, an average offline error of 0 was found for all results reported to two decimals. As expected, the success rate of he algorithm was higher for the major solutions, namely 98%, while the minor solutions yielded a success rate of 81.75%.

Table 6.43: Optima of the tabular holder function in the range $x_1, x_2 \in [-4.5, 4.5]$

Range	Solution type	Locations of optima		Fitness
		x_1	x_2	$f(x_1, x_2)$
$x_1, x_2 \in [-4.5, 4.5]$	Major	-3.36	-3.36	-1.59
		-3.36	3.36	-1.59
		0	0	-2.72
		3.36	-3.36	-1.59
		3.36	3.36	-1.59
	Minor	-3.45	0	-1.05
		-2.83	0	-1.05
		0	-3.45	-1.05
		0	-2.83	-1.05
		0	3.45	-1.05
		0	2.83	-1.05
		3.45	0	-1.05
		2.83	0	-1.05

Table 6.44: VBPSO results for generalized Schwefel function 2.26

♯ Particles	Granularity	Average ♯ evaluations	Average ♯ major solutions	Success rate	Average ♯ minor solutions	Success rate
50	5	51613 ± 762	4 ± 0	100%	4.14 ± 0.12	82.8%
80	5	73098 ± 865	4 ± 0	100%	4.58 ± 0.08	91.6%

Table 6.45: VBPSO results for tabular holder function

♯ Particles	Granularity	Average ♯ evaluations	Average ♯ major solutions	Success rate	Average ♯ minor solutions	Success rate
60	0.3	48762 ± 695	4.9 ± 0.04	98%	6.54 ± 0.14	81.75%

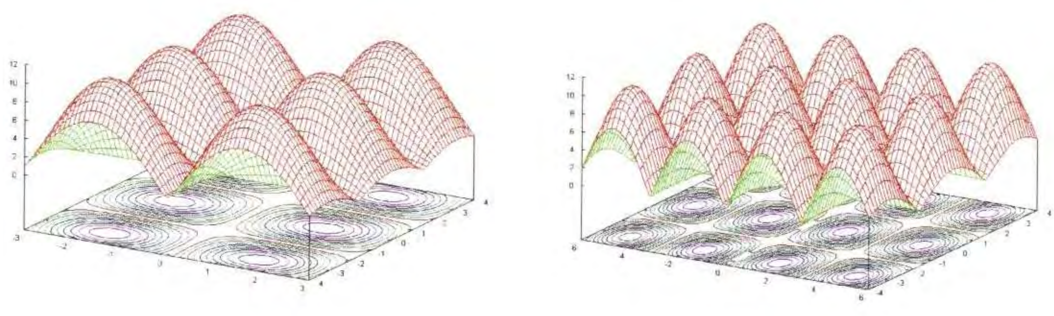
The tube holder function:

The function is defined as

$$f(\mathbf{x_1}, \mathbf{x_2}) = -\left|\sin(x_1)\cos(x_2)e^{|\cos((x_1^2+x_2^2)/200)|}\right| \tag{6.19}$$

Figure 6.16 illustrates the Tube holder function in the range $x_1 \in [-3, 3]$ and $x_2 \in [-4, 4]$, and in the range $x_1 \in [-6, 6]$ and $x_2 \in [-4, 4]$. Optima with slightly differing fitnesses are distributed evenly across the search space. The function has been selected to assess the ability of the vector-based PSO to locate optima when the size of the search space is increased. The algorithm was tested for search spaces containing 6 and 12 optima each as illustrated in Figure 6.16(a) and 6.16(b). Locations and fitnesses of these optima in the range $x_1 \in [-3, 3]$ and $x_2 \in [-4, 4]$ is listed in Table 6.46, while optimal positions and fitnesses in the range $x_1 \in [-6, 6]$ and $x_2 \in [-4, 4]$ are listed in Table 6.47. In addition, the algorithm was also tested for a search space containing 20 optima where $x_1 \in [-6, 6]$ and $x_2 \in [-8, 8]$.

Table 6.48 presents results of the experiments for each of the search spaces. An average success rate of more than 99% is reported in each case. To conclude, these experiments show that the vector-based PSO scales successfully to higher multi-modality when testing the Tube holder function.



(a) (b)

Figure 6.16: The tube holder function.

Table 6.46: Optima of the tube holder function in the range [-3,4] and [3,4]

Solution type	Locations of optima		Fitness
	x_1	x_2	$f(x_1, x_2)$
	-1.57	-3.14	-10.85
	-1.57	0	-10.87
	-1.57	3.14	-10.85
	1.57	-3.14	-10.85
	1.57	0	-10.87
	1.57	3.14	-10.85

Table 6.47: Optima of the tube holder function in the range [-3,4] and [3,4]

Solution type	Locations of optima		Fitness
	x_1	x_2	$f(x_1, x_2)$
	-1.57	-3.14	-10.85
	-1.57	0	-10.87
	-1.57	3.14	-10.85
	1.57	-3.14	-10.85
	1.57	0	-10.87
	1.57	3.14	-10.85
	-4.71	-3.14	-10.73
	-4.71	0	-10.81
	-4.71	3.14	-10.73
	4.71	-3.14	-10.73
	4.71	0	-10.81
	4.71	3.14	-10.73

Table 6.48: VBPSO results for tube holder function

Range (♯ optima)	♯ Particles	Granularity	Average ♯ evaluations	Average ♯ solutions	Average $f'(x)$	Average $f'(y)$	Success rate
$x_1 \in [-3, 3]$, $x_2 \in [-4, 4]$ (6)	30	0.5	23509 ± 472	5.98 ± 0.02	0.0038 ± 0.0003	0.0070 ± 0.0007	99.67%
$x_1 \in [-6, 6]$, $x_2 \in [-4, 4]$ (12)	50	0.5	42204 ± 606	11.96 ± 0.03	0.020 ± 0.002	0.012 ± 0.002	99.67%
$x_1 \in [-6, 6]$, $x_2 \in [-8, 8]$ (20)	80	0.5	72142 ± 678	19.94 ± 0.03	0.030 ± 0.004	0.041 ± 0.005	99.7%

6.5.2 Sensitivity of the vector-based PSO to granularity

The previous sections showed that an additional parameter, called the *granularity*, has to be introduced into the VBPSO if it becomes necessary to merge niches when a number of subswarms are optimized in parallel. One of the strengths of the VBSPO is the ability to optimize irregularly shaped niches. In such cases not all particles are included when niches are formed. The remaining particles form false or extra niches on the boundaries of existing niches. The subswarms occupying these niches will eventually merge with the subswarm having the fittest neighbourhood best value. In the experiments done so far, a value for the granularity was chosen by taking into account the size of the search space as well as the number of optima that can be expected. Intuitively it can be argued that interniche distances indicate an upper bound on the size of the granularity.

To test the influence of different granularities for a range of functions, three two-dimensional functions were used, namely the Himmelblau, Griewank and Rastrigin functions. Sets of 50 runs were conducted with the granularity set to different values. As a starting point, granularity values were used that yielded good results in previous experiments: 0.5 for Himmelblau and Griewank, and 0.1 for Rastrigin.

Results for the three functions are summarized in the following sections.

The Himmelblau function

The Himmelblau function was tested for granularity values ranging from 0.1 to 10.0. The average number of optima located over 50 runs for various values of g is summarized in Table 6.49. Figure 6.17 illustrates the variation of the number of optima located with increase in the granularity. In order to clarify the role of interniche distances in granularity settings, all interniche distances in the Himmelblau function landscape are listed in Table 6.50. Figure 6.18 presents a graphical representation of these distances.

Discussion

The graph in Figure 6.17 shows a sharp decline in the number of optima found for granularity values of between 3 and 4, 6 and 7, and between 8 and 9. These values correspond roughly to the interniche distances as given in Table 6.50 and Figure 6.18. From these results it can be inferred that the VBPSO should find all four optima of the Himmelblau function given that the granularity has a value less than the smallest interniche distance. Formulated differently, the

Table 6.49: Testing granularity for the Himmelblau function

Granularity	Average # solutions	Granularity	Average # solutions
0.1	4 ± 0	5.0	3 ± 0
0.5	4 ± 0	5.5	2.88 ± 0.05
1.0	4 ± 0	6.0	2.72 ± 0.06
1.5	3.96 ± 0.03	6.5	1.88 ± 0.06
2.0	4 ± 0	7.0	1.88 ± 0.05
2.5	4 ± 0	7.5	1.72 ± 0.06
3.0	4 ± 0	8.0	1.58 ± 0.07
3.5	3.92 ± 0.04	8.5	1.4 ± 0.07
4.0	3 ± 0	9.0	1 ± 0
4.5	2.98 ± 0.02	10.0	1 ± 0

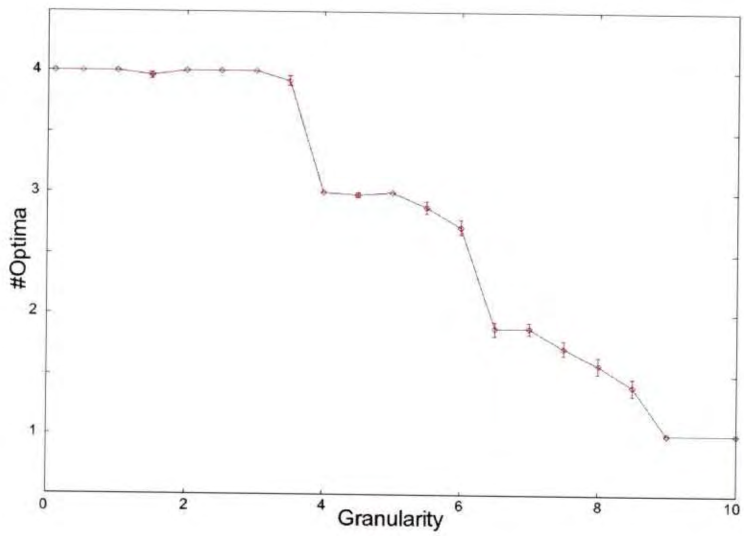


Figure 6.17: Number of optima vs. granularity for the Himmelblau function

VBPSO should find all optima where the interniche distances are larger than the granularity. However, if a function landscape and the interniche distances are unknown, it can be concluded that, for a specific granularity choice, the VBPSO should locate all optima where the smallest Euclidian distance between any two optima is less than the granularity.

Table 6.50: Interniche distances for the Himmelblau function

x_1	x_2	x_1	x_2	Interniche distance
-3.78	-3.28	-2.81	3.13	6.49
-3.78	-3.28	3.58	-1.85	7.50
-3.78	-3.28	3	2	8.59
-2.81	3.13	3.58	-1.85	8.10
-2.81	3.13	3	2	5.91
3.58	-1.85	3	2	3.89

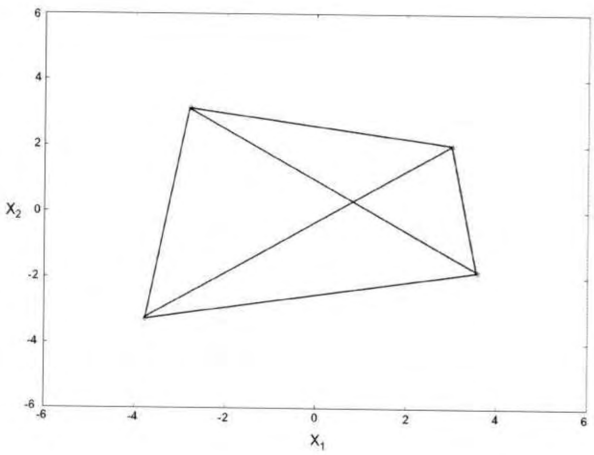


Figure 6.18: Interniche distances for the Himmelblau function

The Griewank function

The Griewank function was tested for granularity values ranging from 0.5 to 7.5. According to experiments reported in section 6.4, the Griewank function was tested with an estimated granularity value of 0.5 which yielded good results. Therefore, the granularity value was increased, starting at 0.5. The average number of optima located over 50 runs for a range of values of the granularity is summarized in Table 6.51. A graphical representation is given in Figure 6.19, which indicates the variation of the number of optima located with increase in the granularity.

Table 6.51: Testing granularity for the Griewank function

Granularity	Average # solutions	Granularity	Average # solutions
0.5	5 ± 0	6.5	1.58 ± 0.07
1.0	5 ± 0	7.0	1.46 ± 0.07
1.5	4.98 ± 0.02	7.5	1.44 ± 0.07
2.0	5 ± 0	8.0	1.52 ± 0.07
2.5	4.96 ± 0.03	8.5	1.48 ± 0.07
3.0	4.92 ± 0.04	9.0	1.6 ± 0.07
3.5	4.76 ± 0.06	9.5	1.56 ± 0.07
4.0	4.48 ± 0.07	10.0	1.64 ± 0.07
4.5	4.44 ± 0.1	10.5	1.57 ± 0.07
5.0	4.18 ± 0.1	11.0	1.5 ± 0.07
5.5	1.62 ± 0.08	11.5	1.42 ± 0.07
6.0	1.48 ± 0.07	12.0	1.38 ± 0.07

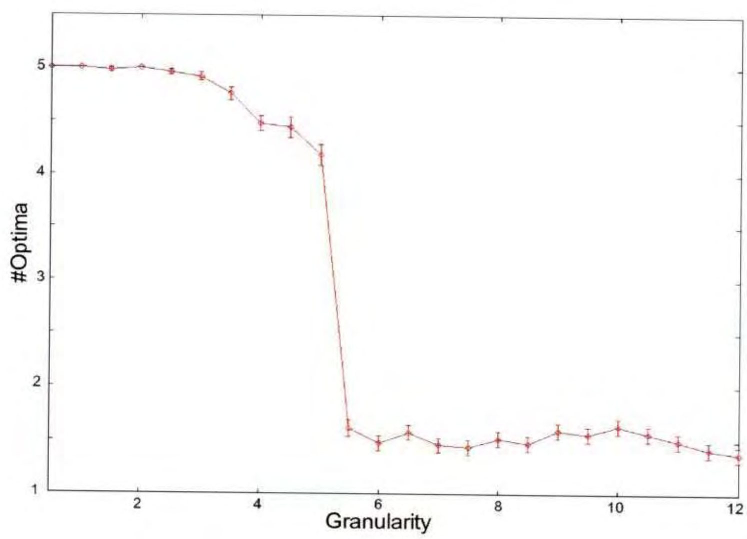


Figure 6.19: Number of optima vs. granularity for the Griewank function

Table 6.52: Interniche distances for the Griewank function

x_1	x_2	x_1	x_2	Interniche distance
0	0	-3.14	-4.44	5.44
0	0	-3.14	4.44	5.44
0	0	3.14	-4.44	5.44
0	0	3.14	4.44	5.44
-3.14	-4.44	-3.14	4.44	8.88
-3.14	-4.44	3.14	-4.44	6.28
-3.14	-4.44	3.14	4.44	10.87
3.14	-4.44	-3.14	4.44	10.87
3.14	-4.44	3.14	4.44	8.88
-3.14	4.44	3.14	4.44	6.28

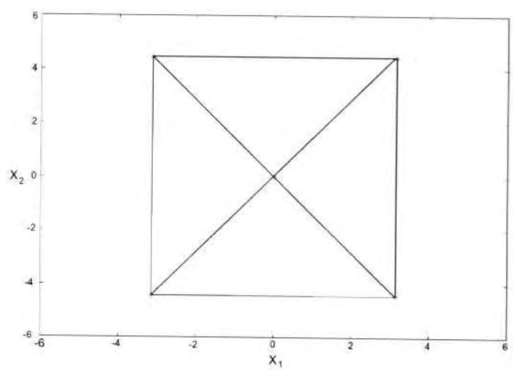


Figure 6.20: Interniche distances for the Griewank function

Discussion

Figure 6.19 shows a slight decline in the number of optima starting at a granularity value of 3, and a sharp decline for a granularity value of just more than 5. Table 6.52 and Figure 6.20 show that the interniche distances between all 5 optima and their nearest neighbours are similar to one another. For larger granularity values the number of optima found stabilizes around 1.3. When g becomes large, the algorithm found one or in some cases two optima. The above figures show that, taking the stochastic nature of PSO into account, the VBPSO locates all five optima most of the time if the granularity is less than the smallest interniche distance.

The Rastrigin function

The Rastrigin function was tested for granularity values ranging from 0.1 to 1.6. The search space is defined for $x_1 \in [-1.25, 1.25]$ and $x_2 \in [-1.25, 1.25]$, where the Rastrigin function has nine optima. The average number of optima found over 50 runs for a range of values of the granularity is summarized in Table 6.53. The variation of the number of optima found with increased granularity is illustrated in Figure 6.21.

Table 6.53: Testing granularity for the Rastrigin function

Granularity	Average # solutions	Granularity	Average # solutions
0.1	9 ± 0	0.9	7.62 ± 0.15
0.2	9 ± 0	1.0	4.14 ± 0.14
0.3	8.96 ± 0.03	1.1	3.24 ± 0.11
0.4	9 ± 0	1.2	3.32 ± 0.13
0.5	8.96 ± 0.03	1.3	3.52 ± 0.12
0.6	8.86 ± 0.05	1.4	2.88 ± 0.14
0.7	8.38 ± 0.08	1.5	1 ± 0
0.8	8.27 ± 0.10	1.6	1 ± 0

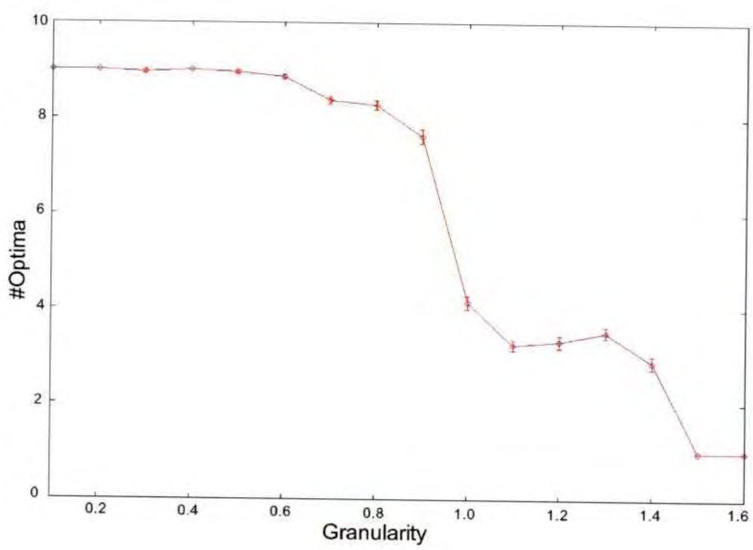


Figure 6.21: Number of optima vs. granularity for the Rastrigin function

Table 6.54: Interniche distances for the Rastrigin function.

x_1	x_2	x_1	x_2	Interniche distance
0	0	0	1	1
0	0	0	-1	1
0	0	1	0	1
0	0	-1	0	1
0	1	1	1	1
0	1	-1	1	1
0	-1	1	-1	1
0	-1	-1	-1	1
1	0	1	1	1
1	0	1	-1	1
-1	0	-1	1	1
-1	0	-1	-1	1
0	0	1	1	1.41
0	0	1	-1	1.41
0	0	-1	1	1.41
0	0	-1	-1	1.41
0	1	1	0	1.41
0	1	-1	0	1.41
0	-1	1	0	1.41
0	-1	-1	0	1.41

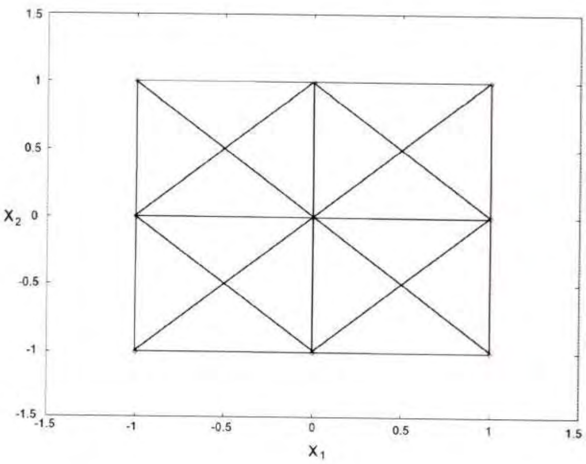


Figure 6.22: Interniche distances for the Rastrigin function

Discussion

For granularity values ranging from 0.1 to 0.9, the VBPSO finds most of the optima of the Rastrigin function, degrading slightly for higher values of the granularity. Again, the interniche distances as listed in Table 6.54 and illustrated in Figure 6.22 indicate upper boundaries for the granularity. Between granularity values of 0.9 and 1 the number of optima drops sharply, meaning that a number of adjacent niches have merged. Niches start merging when the granularity value approaches an interniche distance, which is 1 in this case. Another drop in the number of optima located occurs just before the granularity becomes 1.5. A number of interniche distances are equal to 1.41, which explains the decrease in the number of optima located. The same outcome is experienced as in the case of the previous two functions, namely that the granularity influences performance of the VBPSO algorithm in terms of the number of solutions that can be located.

Introducing a new parameter, the granularity, may be perceived as violating the principle of parsimony. However, for the VBPSO algorithm some value is required to facilitate subswarm merging. Experiments in this section showed that the granularity can be chosen from a wide range of values, provided it is smaller than the smallest interniche distance. Stated differently, the value of the granularity determines the smallest distance between niches with a reasonable expectation of being located.

6.5.3 Scalability of the vector-based PSO

Results obtained in previous sections showed that the vector-based PSO could potentially solve multi-modal optimization problems. However, a study of a niching algorithm will be incomplete if its ability to scale to massively multimodal domains is not investigated.

In the context of multimodal function optimization the concept of scalability relates to the capability of an algorithm to locate all or most of the optima when the size of the search space and the number of optima that can be found in such a space, increase. Modality can be increased by changing the boundaries of the search space, keeping the dimensionality fixed, or by increasing the dimensionality, or both. Previous experiments where swarm sizes were estimated, suggest that an increase in the number of optima requires a bigger swarm. However, success rates of algorithms locating multiple optima also depend on function landscapes. If search spaces are increased in such a way that the new landscape contains niches with shapes and sizes that differ considerably from that of the original landscape, results will not be meaningful.

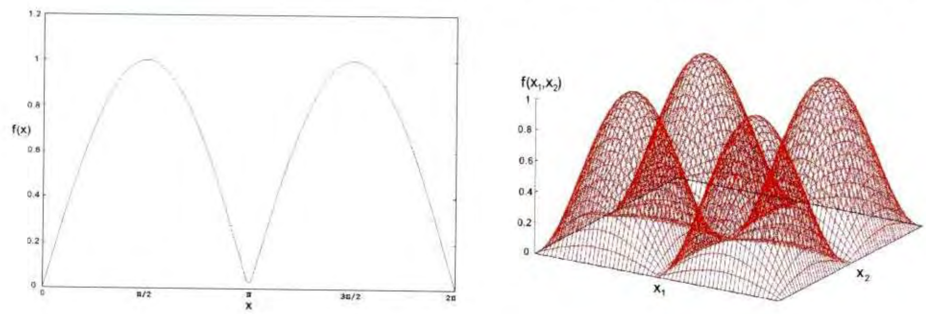
In this study, scalability is investigated by relating the required swarm size to the number of optima in a specified search space. Of the functions tested thus far, the Griewank and Rastrigin functions can be singled out as suitable for testing scalability. Descriptions of the Griewank and Rastrigin functions were presented in section 6.4. For each dimension the positions of the optima are relatively symmetrical. Search spaces can be manipulated to contain progressively higher numbers of optima. In addition, the absolute Sine function was used. The function exhibits a landscape containing regularly spaced optima with similar shapes and fitness over the entire search space. All three these benchmark functions can be described as massively multimodal, as optima are repeated indefinitely. Therefore, it is easy to visualize the function in various search spaces. The absolute Sine function is defined as:

$$f(\mathbf{x}) = \prod_{i=1}^{n} |\sin(x_i)| \tag{6.20}$$

Scalability of the VBPSO was only investigated for the three multimodal functions described above. Many multimodal functions have irregular function landscapes making it difficult to assess the positions of the optima in different search spaces. However, results obtained for these three functions do give an indication of the ability of the VBPSO to scale to higher multimodality. For this study search spaces were increased for each function to contain no more than 100 optima. Beyond that number, executing the VBPSO becomes increasingly computationally expensive. Also, it is believed that practical applications do not necessarily require niching strategies that are capable of locating such large numbers of optima.

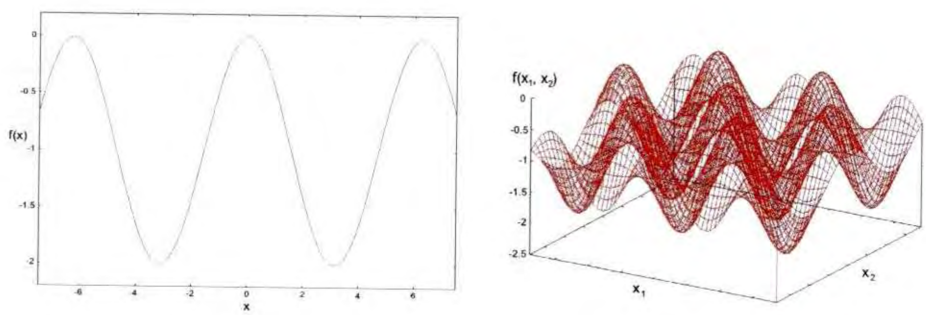
The relationship between the initial optimal swarm size, $|S|$, and the number of solutions was investigated by testing the performance of the VBPSO on a range of swarm sizes for each function in each specified search space. Thus, an optimal swarm size could be deduced for each function in each search space. For this study, search spaces were chosen which allows the optima to be easily determined through a manual process. These search spaces contain numbers of optima ranging from 2 to 96.

Figures 6.23 to 6.25 illustrate the function landscapes in some of these ranges in one and two dimensions. For clarity, the figures depict the optima as maxima, not minima.



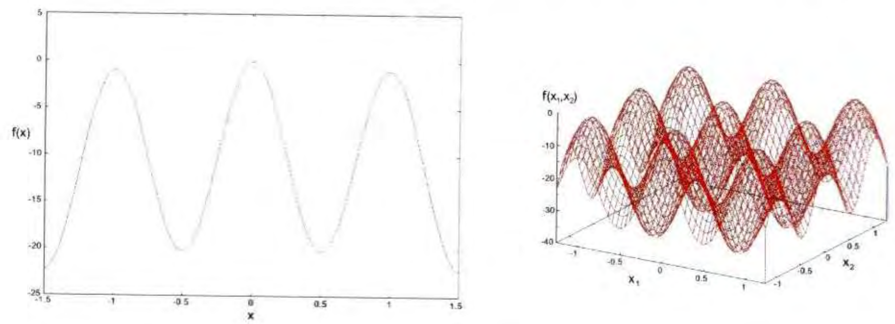
(a) One dimension (b) Two dimensions

Figure 6.23: The absolute Sine function in one and two dimensions



(a) One dimension (b) Two dimensions

Figure 6.24: The Griewank function in one and two dimensions



(a) One dimension (b) Two dimensions

Figure 6.25: The Rastrigin function in one and two dimensions

Positions of optima

Knowing the exact number of optima of a function in a designated search space is critical to support the use of a benchmark function as a test function. Similar distances between optima in the direction of the axes, as in the case of the absolute Sine and Rastrigin functions, enables easier assessment of the number of optima in a demarcated search space. However, in the case of the Griewank function, optima are not arranged symmetrically around the global optimum at $[0, 0]$, and the number and positions of the optima were not obvious. Carefully scrutinizing the search space and testing candidate positions were required to assess exact numbers of optima.

Experimental setup and results

Results of the performance of the vector-based PSO on the absolute Sine, Griewank and Rastrigin functions for a range of search spaces are presented in Tables 6.55 to 6.63 showing the average number of optima for each swarm size, as well as the success rate. Swarm sizes were increased from a small size until the algorithm produced a success rate of at least 98%. The same trend has been observed for all swarm sizes. Figures 6.26, 6.27 and 6.28 show a random selection of graphs where the average number of optima is plotted against the swarm size for the absolute Sine, Griewank and Rastrigin functions respectively.

Table 6.55: Average number of solutions versus swarm sizes for the absolute Sine function - part 1

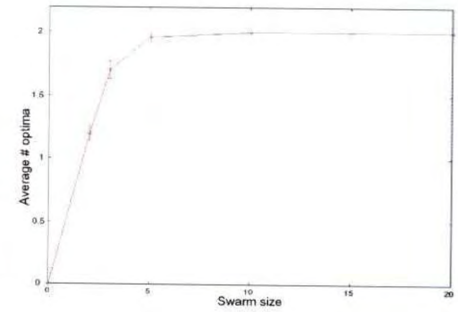
2 optima $[0, 2\pi]$			4 optima $[0, 2\pi]^2$			8 optima $[0, 2\pi]^3$			12 optima $[0, 2\pi]^2 * [0, 3\pi]$		
Swarm size	Average $\sharp$ optima	Success rate	Swarm size	Average $\sharp$ optima	Success rate	Swarm size	Average $\sharp$ optima	Success rate	Swarm size	Average $\sharp$ optima	Success rate
2	1.2 ± 0.06	60%	5	2.44 ± 0.1	61%	10	3.78 ± 0.16	47.25%	50	11.34 ± 0.1	94.5%
3	1.7 ± 0.07	85%	10	3.48 ± 0.08	87%	20	6.64 ± 0.15	83%	60	11.42 ± 0.1	95.17%
5	1.96 ± 0.03	98%	15	3.9 ± 0.04	97.5%	30	7.38 ± 0.09	92.25%	70	11.70 ± 0.07	97.5%
10	2 ± 0	100%	18	3.94 ± 0.03	98.5%	40	7.82 ± 0.06	97.75%	80	11.82 ± 0.05	98.5%
15	2 ± 0	100%	20	3.94 ± 0.03	98.5%	50	$7.9 \pm$ p.05	98.75%			
20	2 ± 0	100%	25	4 ± 0	100%	60	8 ± 0	100%			
			30	4 ± 0	100%						

Table 6.56: Average number of solutions versus swarm sizes for the absolute Sine function - part 2

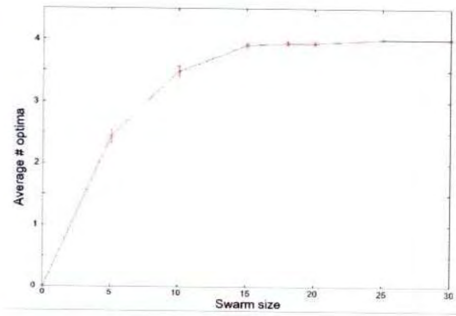
16 optima $[0, 2\pi]^4$			24 optima $[0, 2\pi]^3 * [0, 3\pi]$			32 optima $[0, 2\pi]^3 * [0, 4\pi]$		
Swarm size	Average ♯ optima	Success rate	Swarm size	Average ♯ optima	Success rate	Swarm size	Average ♯ optima	Success rate
20	7.14 ± 0.16	44.63%	20	7.74 ± 0.24	32.25%	20	7.72 ± 0.25	24.13%
40	11.92 ± 0.26	74.5%	40	14.56 ± 0.27	60.67%	40	14.76 ± 0.31	46.13%
60	14.34 ± 0.17	89.63%	60	18.36 ± 0.32	76.5%	60	19.36 ± 0.38	60.5%
80	15.24 ± 0.12	95.25%	80	21 ± 0.19	87.5%	80	23.8 ± 0.37	74.38%
100	15.56 ± 0.07	92.25%	100	22.08 ± 0.19	92%	100	26.08 ± 0.26	81.5%
120	15.84 ± 0.05	99%	120	23 ± 0.13	95.83%	120	28.36 ± 0.19	88.63%
140	15.94 ± 0.03	99.63%	140	23.28 ± 0.09	97%	140	29.4 ± 0.23	91.88%
160	16 ± 0	100%	150	23.4 ± 0.11	97.5%	160	30.48 ± 0.16	95.25%
			160	23.48 ± 0.10	97.5%	180	31.12 ± 0.10	97.25%
			170	23.64 ± 0.09	98.5%	200	31.32 ± 0.11	98.13%

Table 6.57: Average number of solutions versus swarm sizes for the absolute Sine function - part 3

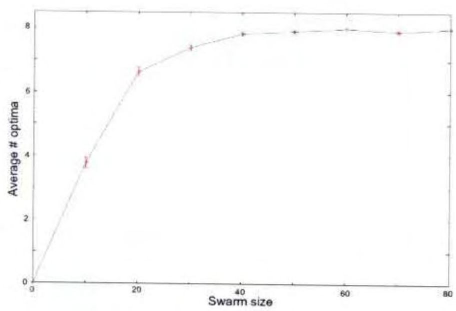
48 optima $[0, 2\pi]^2 * [0, 4\pi]^2$			64 optima $[0, 2\pi]^2 * [0, 4\pi]^2$			96 optima $[0, 2\pi] * [0, 3\pi] * [0, 4\pi]^2$		
Swarm size	Average ♯ optima	Success rate	Swarm size	Average ♯ optima	Success rate	Swarm size	Average ♯ optima	Success rate
50	$\pm$	%	100	35.16 ± 0.51	54.94%	100	39.93 ± 0.57	26.88%
100	34.08 ± 0.26	74.5%	200	53.04 ± 0.39	82.88%	200	66.6 ± 0.60	39.75%
150	$\pm$	%	240	57.72 ± 0.31	90.19%	300	81.46 ± 0.40	59.34%
200	44.12 ± 0.16	91.92%	280	60.04 ± 0.20	93.81%	400	90.56 ± 0.31	94.33%
220	44.76 ± 0.31	93.25%	300	61.36 ± 0.23	95.88%	450	93 ± 0.25	96.88%
240	46 ± 0.15	95.83%	320	61.6 ± 0.17	96.25%	500	92.84 ± 0.25	96.71%
260	46.36 ± 0.17	96.58%	340	61.76 ± 0.22	96.5%	550	93.88 ± 0.20	97.79%
280	47.04 ± 0.14	98%	360	62.48 ± 0.16	97.63%	600	94.08 ± 0.23	98%
			380	62.64 ± 0.15	97.88%			
			400	63.04 ± 0.14	98.5%			



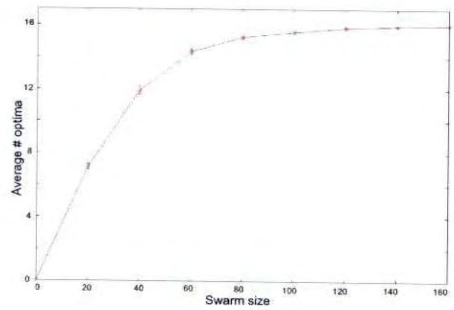
(a) 2 optima



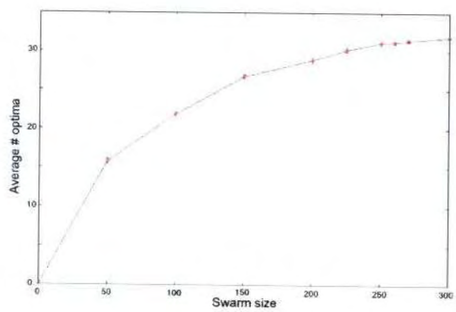
(b) 4 optima



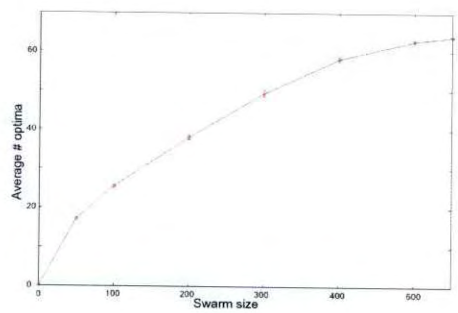
(c) 8 optima



(d) 16 optima



(e) 32 optima



(f) 64 optima

Figure 6.26: % Optima found versus swarm sizes for the absolute Sine function

Table 6.58: Average number of solutions versus swarm sizes for the Griewank function - part 1

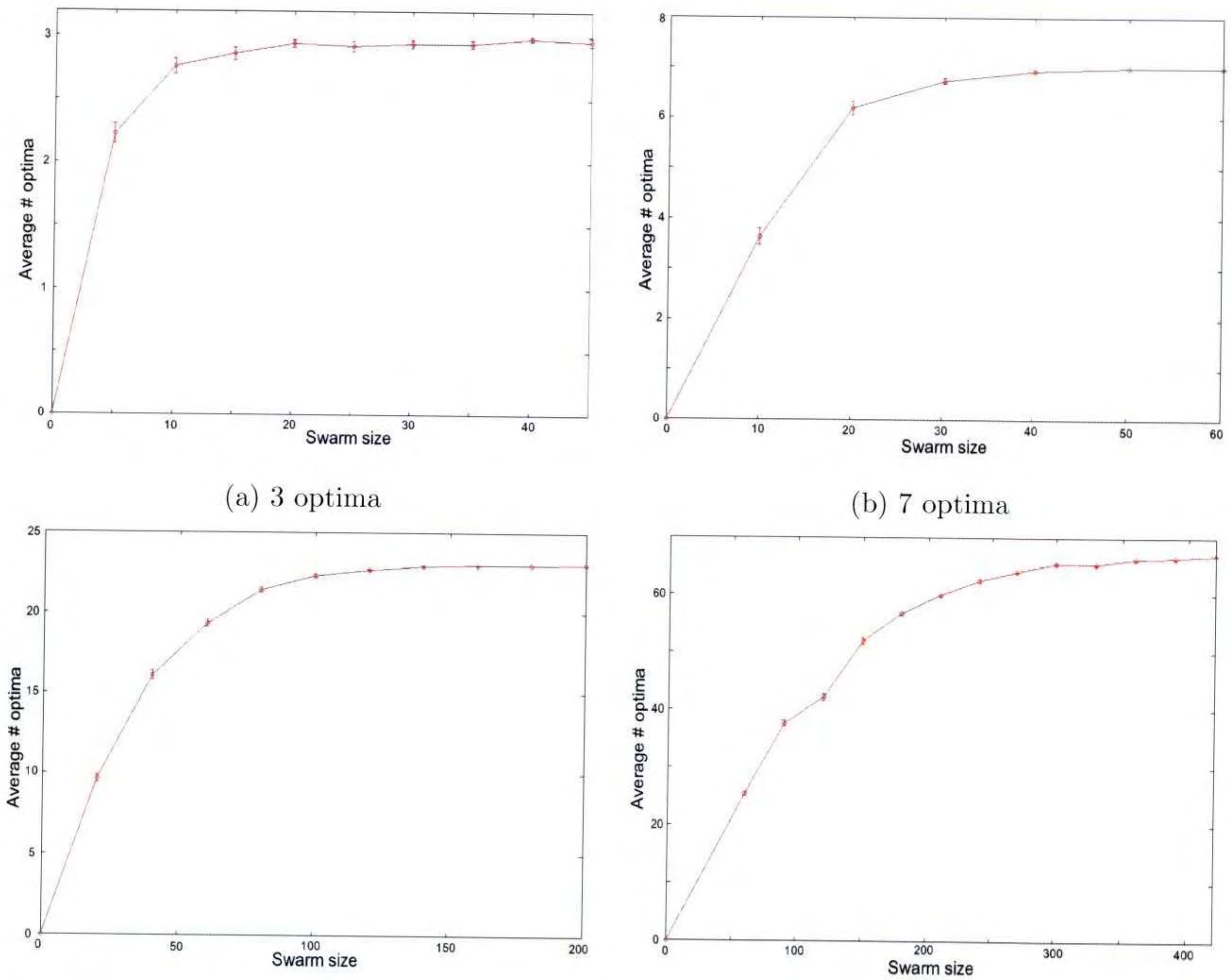
3 optima $[-7.5, 7.5]$			7 optima $[-7.5, 7.5]^2$			11 optima $[-10.5, 10.5] * [-7.5, 7.5]$			17 optima $[-10.5, 10.5]^2$		
Swarm size	Average ♯ optima	Success rate	Swarm size	Average ♯ optima	Success rate	Swarm size	Average ♯ optima	Success rate	Swarm size	Average ♯ optima	Success rate
5	2.22 ± 0.08	74%	10	3.64 ± 0.16	52%	20	7.68 ± 0.20	69.82%	20	9 ± 0.31	52.94%
10	2.76 ± 0.06	92%	20	6.2 ± 0.14	88.57%	30	9.48 ± 0.16	86.18%	30	12.88 ± 0.17	75.76%
15	2.86 ± 0.05	95.33%	30	6.74 ± 0.06	96.29%	40	10.4 ± 0.10	94.55%	40	15 ± 0.19	88.24%
20	2.94 ± 0.03	98%	40	6.94 ± 0.03	99.14%	45	10.52 ± 0.1	95.64%	20	15.88 ± 0.17	93.41%
25	2.92 ± 0.04	97.33%	50	7 ± 0	100%	50	10.96 ± 0.03	99.64%	180	16.48 ± 0.10	96.94%
30	2.94 ± 0.03	98%	60	7 ± 0	100%				210	16.76 ± 0.07	98.59%

Table 6.59: Average number of solutions versus swarm sizes for the Griewank function - part 2

23 optima $[-7.5, 7.5]^3$			31 optima $[-14.5, 14.5]^2$			39 optima $[-17.5, 17.5]^2$		
Swarm size	Average ♯ optima	Success rate	Swarm size	Average ♯ optima	Success rate	Swarm size	Average ♯ optima	Success rate
20	9.66 ± 0.23	42%	40	19.68 ± 0.24	63.48%	50	21.21 ± 0.56	54.38%
40	16.1 ± 0.29	70%	60	25.2 ± 0.31	81.29%	100	33.29 ± 0.34	85.33%
60	19.34 ± 0.22	84.09%	80	27.92 ± 0.19	90.06%	120	34.94 ± 0.29	89.64%
80	21.42 ± 0.19	93.13%	100	29.08 ± 0.15	93.81%	140	36.77 ± 0.18	94.36%
100	22.3 ± 0.21	96.96%	120	30.24 ± 0.13	97.55%	160	37.56 ± 0.16	96.31%
110	22.68 ± 0.07	98.61%	130	30.36 ± 0.09	97.94%	180	38.31 ± 0.13	98.26%
			140	30.44 ± 0.10	98.19%			

Table 6.60: Average number of solutions versus swarm sizes for the Griewank function - part 3

53 optima $[-10.5, 10.5] * [-7.5, 7.5]$			67 optima $[-7.5, 7.5]^4$			83 optima $[-17.5, 17.5] * [-10.5, 10.5] * [-7.5, 7.5]$		
Swarm size	Average ♯ optima	Success rate	Swarm size	Average ♯ optima	Success rate	Swarm size	Average ♯ optima	Success rate
100	35.54 ± 0.33	67.09%	60	25.4 ± 0.34	37.91%	100	48.46 ± 0.37	58.51%
150	44.44 ± 0.24	83.7%	90	37.64 ± 0.51	56.18%	200	72.10 ± 0.40	86.84%
200	49.42 ± 0.23	93.28%	120	42.18 ± 0.59	62.96%	250	76.82 ± 0.28	92.63%
220	51.13 ± 0.15	96.53%	150	52 ± 0.52	77.61%	300	80.54 ± 0.22	96.96%
240	50.83 ± 0.17	96%	180	56.8 ± 0.39	84.78%	320	80.98 ± 0.19	97.59%
260	51.82 ± 0.14	97.74%	210	60 ± 0.28	89.55%	340	81.08 ± 0.17	97.69%
270	52.21 ± 0.12	98.49%	240	62.8 ± 0.33	93.43%	350	82.13 ± 0.18	98.94%
			270	64.1 ± 0.12	95.67%			
			300	65.5 ± 0.13	97.76%			
			310	65.96 ± 0.10	98.44%			



(a) 3 optima

(b) 7 optima

(c) 23 optima

(d) 67 optima

Figure 6.27: % Optima versus swarm sizes for the Griewank function

Table 6.61: Average number of solutions versus swarm sizes for the Rastrigin function

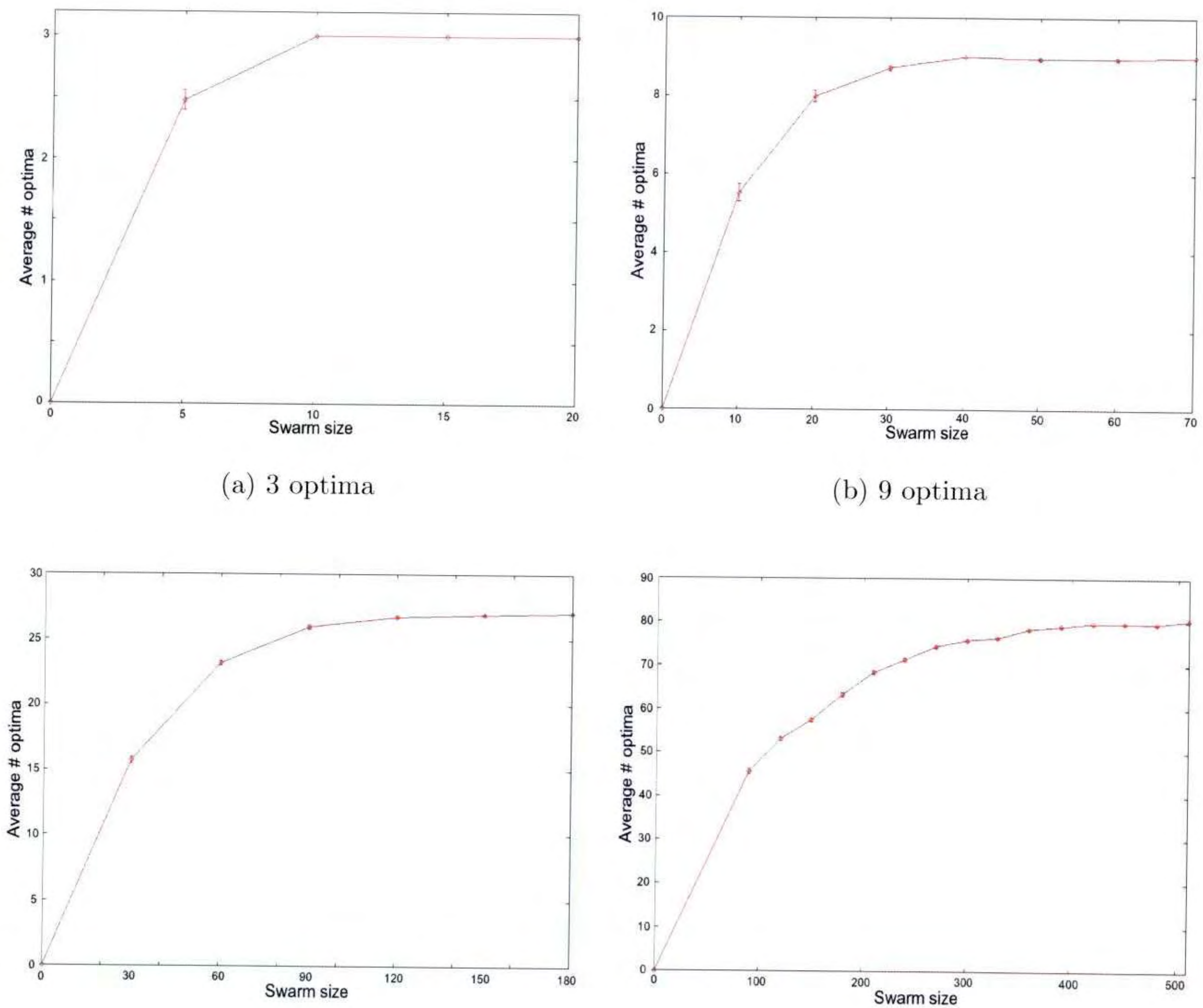
3 optima $[-1.5, 1.5]$			9 optima $[-1.5, 1.5]^2$			15 optima $[-1.5, 1.5] * [-2.5, 2.5]$			21 optima $[-1.5, 1.5] * [-3.5, 3.5]$		
Swarm size	Average ♯ optima	Success rate	Swarm size	Average ♯ optima	Success rate	Swarm size	Average ♯ optima	Success rate	Swarm size	Average ♯ optima	Success rate
5	2.48 ± 0.08	82.67%	10	5.52 ± 0.22	61.33%	20	10.12 ± 0.27	67.47%	30	14.28 ± 1.25	68%
10	3 ± 0	100%	20	8 ± 0.15	88.89%	30	12.64 ± 0.18	84.27%	40	17.92 ± 0.19	85.33%
15	3 ± 0	100%	30	8.72 ± 0.06	96.89%	40	13.96 ± 0.14	96%	60	20 ± 0.14	95.24%
20	3 ± 0	100%	40	9 ± 0	100%	50	14.56 ± 0.09	97.07%	70	20.2 ± 0.11	96.19%
			50	8.96 ± 0.03	99.56%	60	14.64 ± 0.09	97.6%	80	20.68 ± 0.08	98.48%
			60	8.98 ± 0.02	99.78%	70	14.84 ± 0.06	98.93%			
			70	9 ± 0	100%						

Table 6.62: Average number of solutions versus swarm sizes for the Rastrigin function - part 2

27 optima $[-1.5, 1.5]$			35 optima $[-2.5, 2.5] * [-3.5, 3.5]$			45 optima $[-1.5, 1.5]^2 * [-2.5, 2.5]$		
Swarm size	Average ♯ optima	Success rate	Swarm size	Average ♯ optima	Success rate	Swarm size	Average ♯ optima	Success rate
30	15.72 ± 0.26	58.22%	50	24.2 ± 0.36	69.14%	50	24.8 ± 0.41	55.11%
60	23.14 ± 0.21	85.7%	100	33.16 ± 0.17	94.74%	100	37.8 ± 0.27	84%
90	25.92 ± 0.15	96%	120	34.12 ± 0.10	97.49%	120	39.56 ± 0.29	87.91%
110	25.96 ± 0.16	96.15%	130	34.1 ± 0.13	97.6%	140	41.4 ± 0.20	92%
120	26.68 ± 0.08	98.81%	140	34.72 ± 0.06	99.2%	160	42.68 ± 0.15	94.84%
						180	43.44 ± 0.16	96.53%
						200	44.16 ± 0.14	98.13%

Table 6.63: Average number of solutions versus swarm sizes for the Rastrigin function - part 3

63 optima $[-1.5, 15]^2 * [-3.5, 3.5]$			75 optima $[-1.5, 1.5] * [-2.5, 2.5]^2$			81 optima $[-1.5, 1.5]^4$		
Swarm size	Average ♯ optima	Success rate	Swarm size	Average ♯ optima	Success rate	Swarm size	Average ♯ optima	Success rate
50	25.88 ± 0.36	41.08%	100	47.08 ± 0.50	62.77%	90	45.7 ± 0.60	56.42%
100	45.08 ± 0.32	71.56%	150	57.72 ± 0.45	76.96%	120	53.2 ± 0.47	65.68%
150	54.16 ± 0.32	85.97%	200	65.56 ± 0.38	87.41%	150	57.5 ± 0.44	70.99%
200	59.44 ± 0.23	94.34%	250	69.2 ± 0.30	92.27%	180	63.3 ± 0.47	78.15%
220	59.7 ± 0.22	94.7%	275	70.92 ± 0.25	94.56%	210	68.4 ± 0.40	84.44%
240	60.6 ± 0.23	96.19%	300	72.28 ± 0.22	96.37%	240	71.3 ± 0.33	88.02%
260	61.8 ± 0.16	98.10%	350	73.5 ± 0.17	98%	270	74.3 ± 0.29	91.73%
			360	73.64 ± 0.20	98.19%	300	75.8 ± 0.31	93.58%
						330	76.4 ± 0.25	94.32%
						360	78.4 ± 0.19	96.79%
						390	79 ± 0.16	97.53%
						400	79.48 ± 0.18	98.12%



(a) 3 optima

(b) 9 optima

(c) 27 optima

(d) 81 optima

Figure 6.28: % Optima versus swarm sizes for the Rastrigin function

Table 6.64: Minimum swarm sizes required to locate 98% of possible optima for the absolute Sine function

Search space	Optima	Swarm size	R^2
$[0, 2\pi]$	2	5	
$[0, 2\pi]^2$	4	18	
$[0, 2\pi]^3$	8	50	
$[0, 2\pi]^2 * [0, 3\pi]$	12	80	
$[0, 2\pi]^4$	16	120	0.9980
$[0, 2\pi]^3 * [0, 3\pi]$	24	170	
$[0, 4\pi] * [0, 2\pi]^3$	32	210	
$[0, 4\pi] * [0, 3\pi] * [0, 2\pi]^2$	48	280	
$[0, 4\pi]^2 * [0, 2\pi]^2$	64	400	
$[0, 4\pi]^2 * [0, 3\pi] * [0, 2\pi]$	96	600	

From the results presented in this section, minimum swarm sizes required to locate at least 98% of the optima in predefined search spaces were determined for all three functions. Tables 6.64, 6.65 and 6.66 summarize the minimum swarm sizes for which an average success rate of above 98% have been reported.

For each of the functions, Figure 6.29 illustrates the optimal swarm sizes, $|S|^*$, plotted against the actual number of optima, x, that can be located in the search space. For each of the functions a linear regression was done to describe the dependence of optimal swarm size on the number of possible optima, given that $x \geq 0$. For the absolute Sine function the linear equation is

$$|S|^* = 6.26x$$

For the Griewank function the equation is

$$|S|^* = 4.44x$$

For the Rastrigin function the equation is

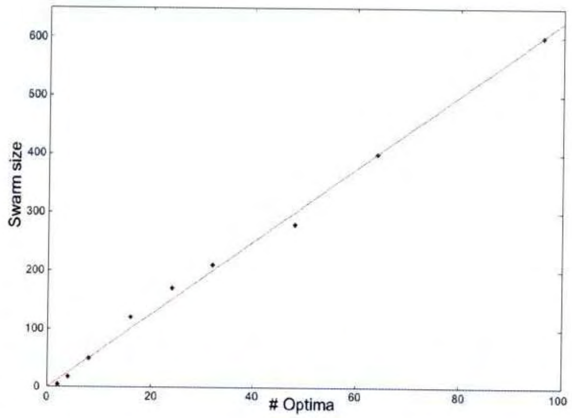
$$|S|^* = 4.56x$$

Table 6.65: Minimum swarm sizes required to locate 98% of possible optima for the Griewank function

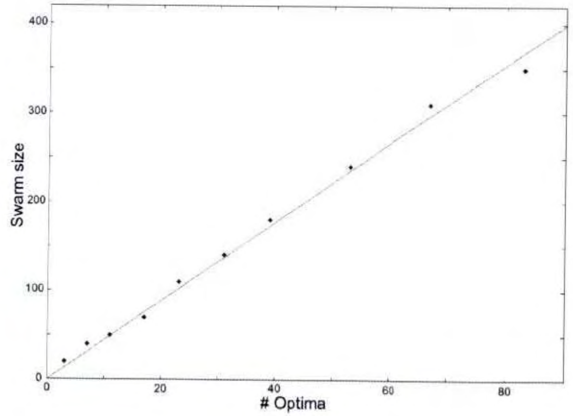
Search space	Optima	Swarm size	R^2
$[-7.5, 7.5]$	3	20	0.9977
$[-7.5, 7.5]^2$	7	40	
$[-10.5, 10.5] * [-7.5, 7.5]$	11	50	
$[-10.5, 10.5]^2$	17	70	
$[-7.5, 7.5]^3$	23	110	
$[-14.5, 14.5]^2$	31	140	
$[-17.5, 17.5]^2$	39	180	
$[-10.5, 10.5] * [-7.5, 7.5]$	53	240	
$[-7.5, 7.5]^4$	67	310	
$[-17.5, 17.5] * [-10.5, 10.5] * [-7.5, 7.5]$	83	350	

Table 6.66: Minimum swarm sizes required to locate 98% of possible optima for the Rastrigin function

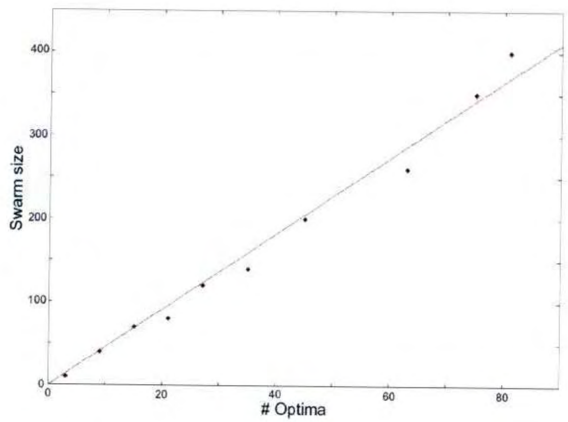
Search space	Optima	Swarm size	R^2
$[-1.5, 1.5]$	3	10	0.9985
$[-1.5, 1.5]^2$	9	40	
$[-1.5, 1.5] * [-2.5, 2.5]$	15	70	
$[-1.5, 1.5] * [-3.5, 3.5]$	21	80	
$[-1.5, 1.5]^3$	27	120	
$[-2.5, 2.5] * [-3.5, 3.5]$	35	140	
$[-1.5, 1.5]^2 * [-2.5, 2.5]$	45	200	
$[-1.5, 1.5]^2 * [-3.5, 3.5]$	63	260	
$[-1.5, 1.5] * [-2.5, 2.5]^2$	75	360	
$[-1.5, 1.5]^4$	81	400	



(a) The absolute Sine function



(b) The Griewank function



(c) The Rastrigin function

Figure 6.29: Optimal swarm sizes versus actual number of optima for three benchmark functions

Tables 6.64 to 6.66 also lists the coefficient of determination, R^2, for each regression. In all three cases $R^2 > 0.99$, indicating that the above linear functions provide a very good fit. Therefore, for all the functions considered, a linear relationship

$$|S|^* = ax + c$$

exists between the optimal swarm size and the actual number of optima where a and c are constants depending on the objective function and the search space where it is investigated. Please note that the value of c was found to be 0 for the functions considered in this study.

6.6 A comparative study of three PSO niching approaches

A number of niching methods for particle swarm optimization have been described in chapter 4. This section compares the performance of two of these methods, the species-based PSO and NichePSO, to that of the vector-based PSO. In this study the enhanced parallel vector-based PSO is used and is referred to as the vector-based PSO (VBPSO). All three niching methods optimizes subswarms in parallel, but the manner in which niches form, differs.

6.6.1 Experimental procedure

Comparison of different algorithms requires similar experimental setups. The species-based PSO and NichePSO algorithms are tested on the same seven two-dimensional functions as the vector-based PSO, using the same number of particles for each function. Given that the initial distribution of particles throughout the problem space can influence the performance of an algorithm considerably, both the species-based PSO and the vector-based PSO were implemented using Sobol sequences as a random number generator. NichePSO uses *Fauré* sequences, also yielding even distributions of initial particle positions over the search space. The number of iterations was also set to 500 for each algorithm. Average results were calculated for each algorithm.

The species-based PSO requires a niche radius to be set in advance. Each function was tested with a small range of niche radii. Results using the radius yielding the best outcomes are reported. NichePSO was tested using the CILib framework developed by the CIRG research group at the department of Computer Science, University of Pretoria (http://cilib.sourceforge.net). NichePSO requires a merging parameter, μ. In order to compare the algorithms, similar settings were used as far as possible. Therefore, for each function μ was set to the granularity used in the vector-based algorithm, but it was normalized as required by NichePSO.

6.6.2 Results

Results are reported in Table 6.67. The success rate of each algorithm for each function is reported. In order to compare the algorithms, results of the vector-based PSO are repeated in the table. The granularity, niche radius or merging parameter is listed for every function, as well as the number of particles.

6.6.3 Discussion

This section discusses the behaviour of each of the algorithms when tested on functions with differing characteristics.

The species-based PSO performed well on the Himmelblau, Griewank and Rastrigin functions with a success rate of more than 90%. Each of these functions has a number of well-defined optima where the heights and interniche distances differ very little or not at all. Given the right choice of a niche radius, a good performance of such a simple, elegant and effective algorithm can be expected. However, when the search space is more convoluted and the shapes, sizes and placing of optima in the search space are less symmetrical, the niche radii differ from niche to niche and the performance degrades. These expectations are confirmed by the results where the success rate of the Ursem F1 function is 86.67%, the Ursem F3 function 73.33% and the six hump camel function 63.89%.

NichePSO performed well on a number of the functions: the Himmelblau, Griewank, Ackley and Ursem F1 functions all have success rates of more than 90%. However, the Rastrigin function has a 80% success rate in this implementation; an unexpected result given the results obtained by the original implementation where the success rate was 100% for a region where 9 optima occurred [13] [14]. The indication is that too many niches merge and that the algorithm needs some fine tuning of parameters to prevent these occurrences. Such fine tuning, which has to be adapted for each function, has not yet been incorporated into the NichePSO implementation in the CILib framework. For the Ursem F3 and six hump camel functions where the niche sizes differ considerably, the performance degrades to success rates of 35.83% and 33.33%. For the six hump camel function only the two large niches were located in all cases. Therefore it can be concluded that too many niches merge if niche sizes differ considerably.

While NichePSO manipulates subswarms in ingenious ways, the merging process is not robust enough if the objective functions become more convoluted.

For the small subset of functions that has been tested in this study, all three algorithms performed equally well on functions where the landscapes are relatively symmetrical. However, the vector-based PSO outperformed the other two algorithms on functions where the differences between the interniche distances, the fitnesses, and niche radii of the various optima become larger, for example, the Ursem F3 and six hump camel functions. Table 6.67 show larger differences in performance for these functions. These differences can be ascribed to the strategy used by VBPSO to calculate a separate niche radius for each niche, as well as the forming of false niches. The latter occurs more often when niches are not symmetrical. Different niche radii ensure that subswarms in smaller niches are not absorbed by those in larger niches, resulting in a smaller success rate. The concept of false niches address the deficiency of niche radii in general, which assumes that a niche occupies a spherical region. False niches accommodate particles inside a niche but falling outside regions demarcated by niche radii. Thus, niches of all shapes and sizes are accommodated by VBPSO, constituting a robust algorithm that performs well for convoluted and asymmetrical function landscapes.

Figure 6.30 shows a bar graph where the performance of the three niching strategies is plotted for all seven functions. The number of optima found during 50 runs is represented as a percentage of the total number of optima.

6.7 Conclusion

This chapter presented results of the vector-based PSO applied to static environments. Three versions of the algorithm, namely the sequential, parallel and enhanced parallel vector-based PSO were tested extensively on a number of one- and two-dimensional objective functions. Each implementation represents an improvement on the previous version. An analysis of the algorithm was conducted where the influence of the granularity on niching ability as well as the scalability were tested. Optimal swarm sizes were deduced for three multimodal functions in a number of dimensions, and a relationship was derived between optimal swarm sizes and the number of optima. The chapter was concluded by reporting on a comparative study of three niching algorithms that emphasized the strengths, effectiveness and robustness of the vector-base PSO.

Table 6.67: Comparing three niching algorithms

Functions	♯ Particles	Vector-based PSO		Species-based PSO		NichePSO	
		Granularity	Success rate	Niche radius	Success rate	Merging parameter μ	Success rate
Himmelblau	30	0.5	100%	3.75	99.17%	0.5	100%
Griewank	40	0.5	100%	3	100%	0.5	97.33%
Rastrigin	60	0.1	99.63%	0.6	91.11%	0.1	80%
Ackley	60	0.3	99.63%	0.6	78.51%	0.3	91.11%
Ursem F1	30	0.5	100%	1.8	86.67%	0.5	95%
Ursem F3	40	0.3	100%	0.7	73.33%	0.3	35.83%
Six hump camel	50	0.3	99.44%	0.6	63.89%	0.3	33.33%

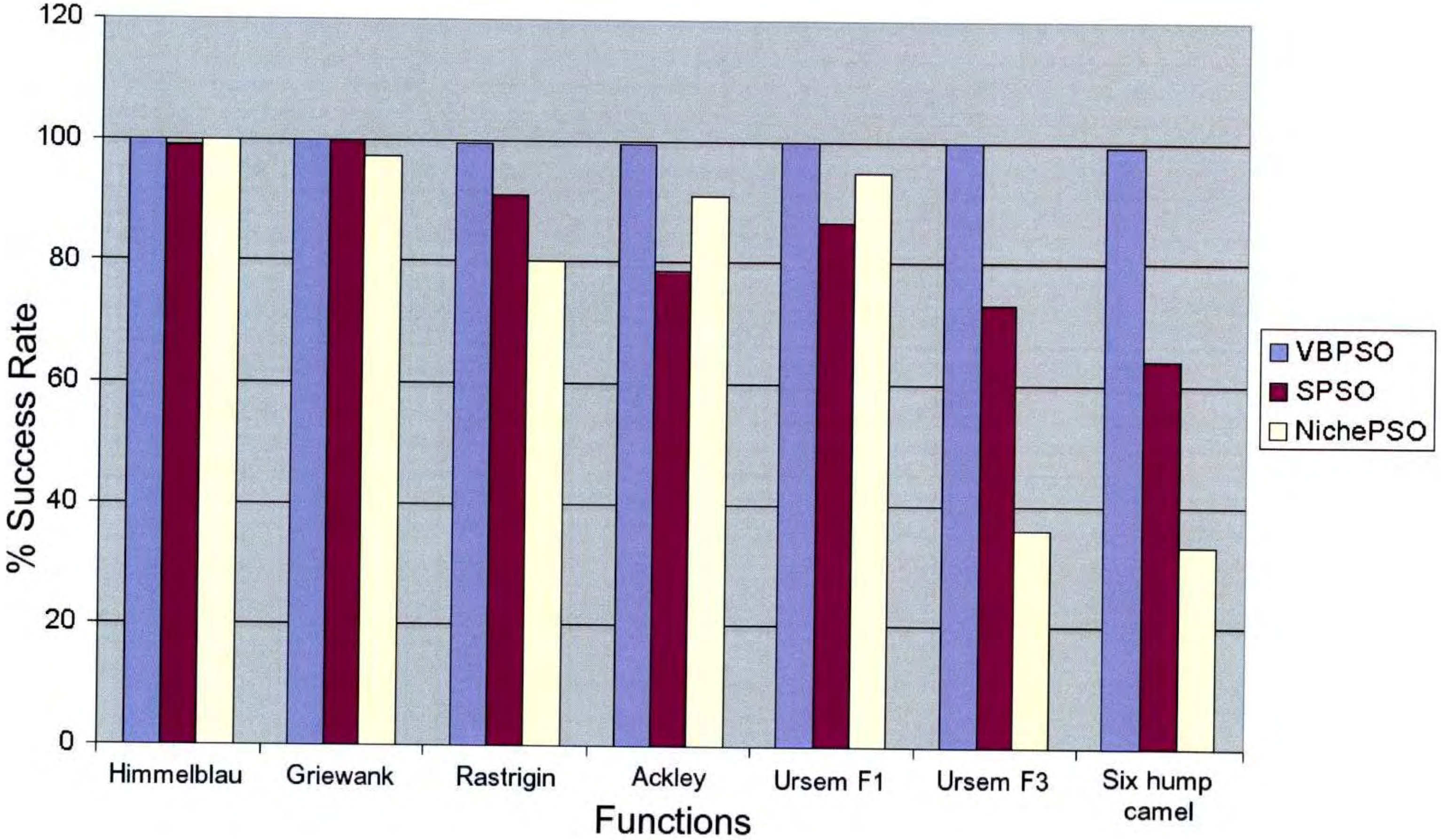


Figure 6.30: Comparing performance of three niching strategies for seven functions.