Chapter 8

The Vector-Based PSO Applied to Dynamic Environments

This chapter presents results yielded by the dynamic vector-based PSO on the moving peaks benchmark function [25]. A number of scenarios illustrating a variety of dynamic changes are designed. The dynamic species-based PSO is tested for the same scenarios, and the performances of the two algorithms are compared. The chapter concludes with an investigation into the sensitivity of the dynamic vector-based PSO to changes in severity.

8.1 Introduction

To test the dynamic vector-based PSO exhaustively for all possible moving optima is a major undertaking. Ideally, new optimal positions should be located each time any change in the environment occurs. Environmental changes are brought about in different ways, often related to the problem that the algorithm tries to solve. Some tracking algorithms incorporate ways to detect changes, which may not always be necessary, as the algorithm may be informed when input parameters change. For some problems adaptation of optimal positions at specific intervals may suffice. The dynamic vector-based PSO does not detect the changes, but assume that the algorithm is triggered in some way when a change in the landscape occurs.

This chapter reports on a number of separate scenarios. Each scenario illustrates some

way in which the landscape changes. For each scenario a number of peaks was chosen and movements over the search space set up, using the moving peaks benchmark function (refer to section 7.5.1). Changes include changes in the position of the peaks, changes in the fitness, and situations where peaks disappear and appear again at different positions.

8.2 Experimental setup and results

The following settings were used for all experiments:

- Experiments are performed in a two-dimensional search space in the range $x_1, x_2 \in [-1, 1]$.
- The initial number of particles is set to 30 for all experiments.
- A uniform distribution of particles throughout the search space is ensured by using Sobol sequences as a random number generator. Sobol sequences are described in chapter 5 and are used throughout this study.
- The granularity parameter is set to 0.05 for all the experiments. The value was estimated taking into consideration the size of the search space and the number of expected peaks. According to the conclusions derived at in section 5.7.1, the granularity should be smaller than the smallest interniche distance in order to locate all optima. In this case it can be stated that the algorithm should locate all optima where the Euclidian distance separating them, is more than 0.05.
- The random sequences used in the velocity update equation, $r_1 \sim U(0, 1)$ and $r_2 \sim U(0, 1)$ are scaled by constants $c_1, c_2 \in [0, 2]$. Similar to experiments with the vector-based PSO described in chapter 5, $c_1 = c_2 = 1$, that is, the maximum step size in the direction of the neighbourhood best position, referred to by c_2, is equal to c_1, the maximum step size in the direction of the personal best position.
- The inertia weight, w, is used as a scaling factor associated with the velocity in the previous time step. As in the case of the vector-based PSO, w is set to the lower bound, 0.8.
- After each modification to the objective function that results in a changed function landscape, all previous particles are discarded and small subswarms, each consisting of a newly initialized particle at the previous optimal position, as well as three additional particles

near to that position (stage 1). During stage 2, 30 particles are initialized throughout the search space. These subswarms are optimized by repeating the updating process 500 times before the next environment change.

- The algorithm is run 50 times for each experiment.

Results are presented as follows:

Function evaluations: During each optimization phase, the number of function evaluations of the objective function was counted and recorded. Results are presented as the average number of evaluations for the first step as well as the subsequent steps of the algorithm, where a step indicates the part of the algorithm that is executed before the next objective function change. The average number of evaluations over 50 runs was calculated separately for the first step of the algorithm (when optima are located) and the subsequent steps (where optima are tracked).

Average number of optima: The average number of optima located over 50 runs, is listed for each step of the algorithm.

Offline error: The offline error, or average distances from the true optimal positions over 50 runs, is calculated and listed for each step of the algorithm. For objective functions tested in previous chapters, partial derivatives were used, but when a function contains cone-shaped peaks, the derivative is equal to the slope of the cone, except at the exact optimal position where it is 0. Therefore, for the moving peaks benchmark, the use of the offline error is a better indication of the quality of the solution.

Success rate: The success rate is the total number of optima located over 50 runs calculated as a percentage of the total number of possible optima. Again, the success rate is reported for each step of the algorithm.

In the following section a number of scenarios are described, the function landscapes at each step illustrated, and results presented.

Scenario 1

Using De Jong and Morrison's test problem generator [25] described by equation (7.3), a test function with three peaks was created. The positions of the peaks change over six steps in such a way that no two peaks merge with one another. Table A.4 lists the settings of the function

for each step, while Figure 8.1 illustrates the function landscape at each step. At each step the function landscape shows three separate peaks. Peaks are labeled $P1$ to $P3$ to correspond with columns in the table. Changes in the function landscape are described as follows:

Peak 1: The initial position of the peak is at $[-0.6, -0.8]$. The x_2-component remains the same, while the x_1-component is incremented by 0.2 at each step, resulting in a movement parallel to the x_2 axis, stopping at $[0.4, -0.8]$. The height and radius of the peak remains the same.

Peak 2: Movement of peak 2 commences at $[-0.5, 0.3]$. The x_1-component is incremented by 0.2 and the x_2-component by 0.1 at each step. The peak moves across the space towards $[0.5, 0.8]$ in the adjacent quadrant. The height and radius of the peak remains the same.

Peak 3: The starting position of peak 3 is at $[0.5, 0]$. The x_1-component is decremented by 0.1 at each step while the x_2-component does not change. The peak moves along the x_1 axes and stops at the centre of the search space at $[0, 0]$.

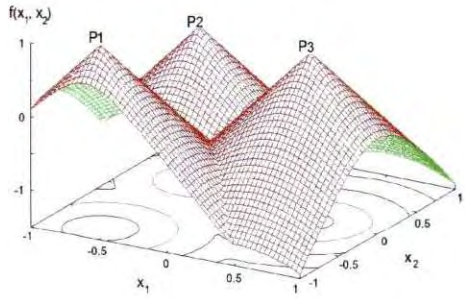
Scenario 1 illustrates simple movements of optima across the search space. Since it is assumed that no new peaks appear, only stage 1 of the algorithm is activated. Once the initial optima have been found, fewer particles are required to track the optima, depending on the number of optima. Results presented in Table 8.2 show that the average number of function evaluations during the last 5 steps of all the experiments is less than one third of the average number of evaluations required to locate the initial optima. Therefore it can be concluded that, once optima are located, the computational complexity required by the dynamic vector-based PSO to track existing optima is much less than frequent re-optimization would entail, given that the severity is not too large. According to the results, the offline error is less than 10^{-12}, indicating very good quality solutions.

Table 8.1: Scenario 1: Positions of 3 optima over 6 steps

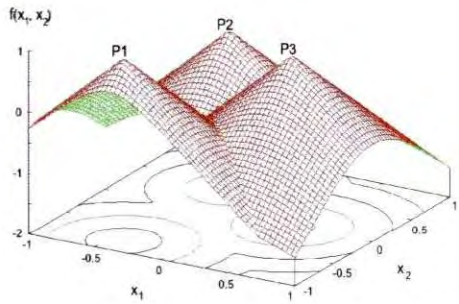
Step	Peak 1				Peak 2				Peak 3			
	x_1	x_2	H	R	x_1	x_2	H	R	x_1	x_2	H	R
1	-0.6	-0.8	1	2	-0.5	0.3	0.6	2	0.5	0	0.8	2
2	-0.4	-0.8	1	2	-0.3	0.4	0.6	2	0.4	0	0.8	2
3	-0.2	-0.8	1	2	-0.1	0.5	0.6	2	0.3	0	0.8	2
4	0	-0.8	1	2	0.1	0.6	0.6	2	0.2	0	0.8	2
5	0.2	-0.8	1	2	0.3	0.7	0.6	2	0.1	0	0.8	2
6	0.4	-0.8	1	2	0.5	0.8	0.6	2	0	0	0.8	2

Table 8.2: Scenario 1: Results

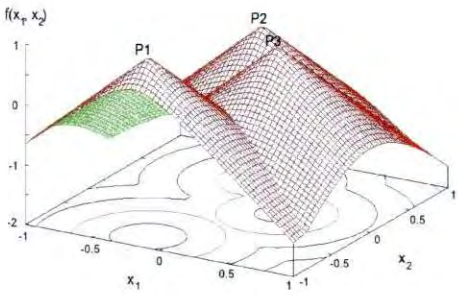
Step	Average ♯ evaluations	Average ♯ optima	Offline error	Success rate
1	25558 ± 490	3 ± 0	1.77E-13 ± 1.22E-13	100%
2	6934 ± 12	3 ± 0	7.16E-18 ± 1.57E-18	100%
3	6922 ± 7	3 ± 0	4.82E-18 ± 6.82E-19	100%
4	6927 ± 10	3 ± 0	3.47E-17 ± 5.94E-18	100%
5	6961 ± 30	3 ± 0	3.12E-18 ± 4.77E-19	100%
6	6924 ± 10	3 ± 0	1.94E-17 ± 4.24E-18	100%



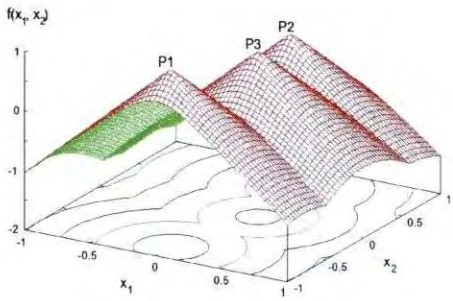
(a) Step 1



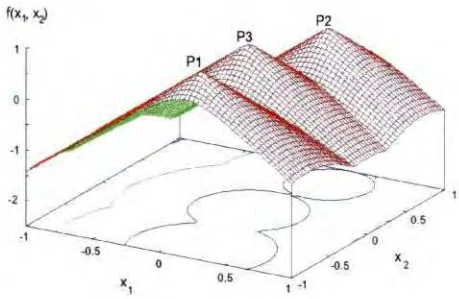
(b) Step 2



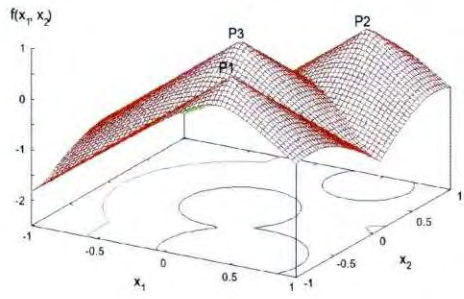
(c) Step 3



(d) Step 4

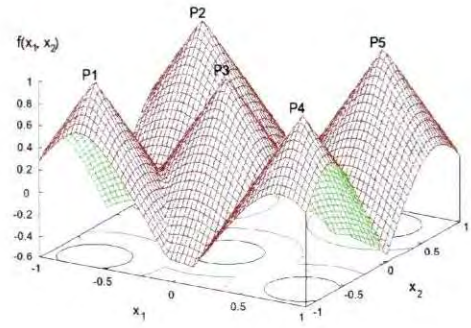


(e) Step 5

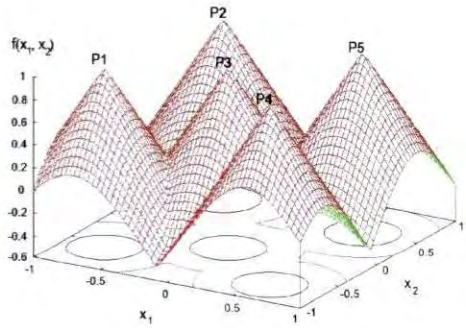


(f) Step 6

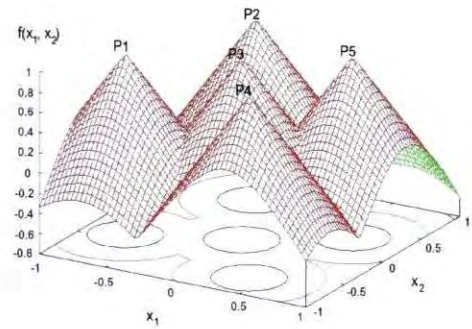
Figure 8.1: Scenario 1: Function landscapes



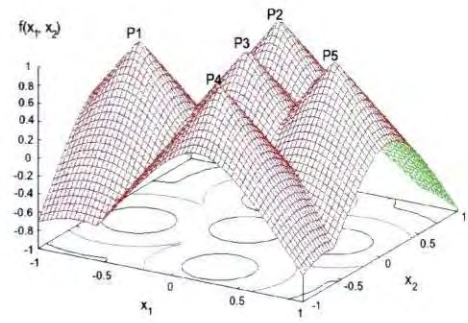
(a) Step 1



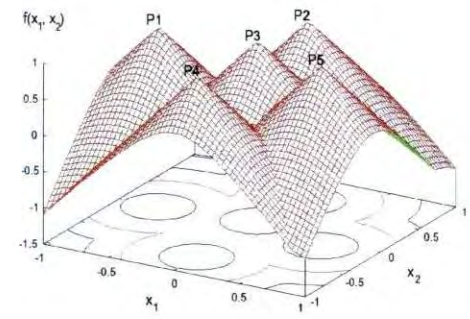
(b) Step 2



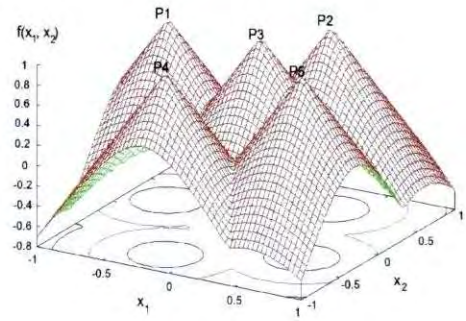
(c) Step 3



(d) Step 4

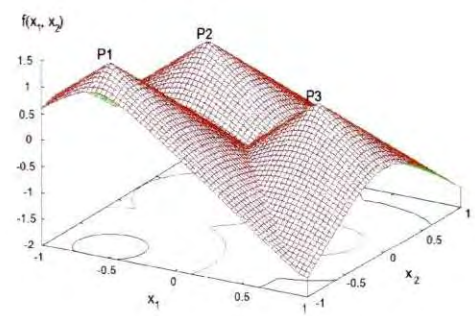


(e) Step 5

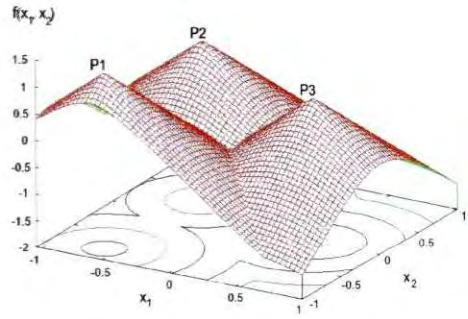


(f) Step 6

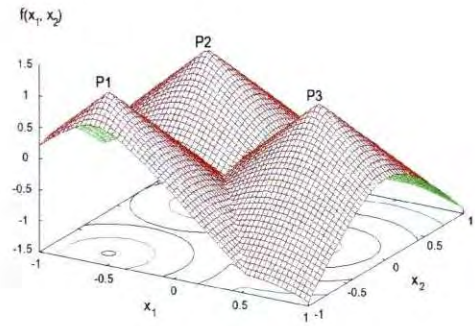
Figure 8.2: Scenario 2: Function landscapes



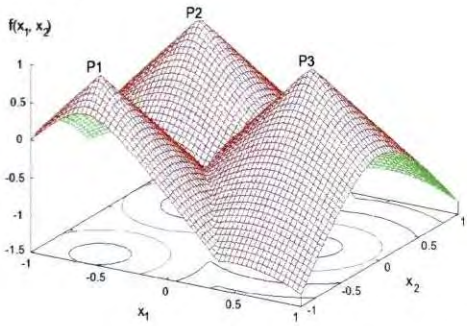
(a) Step 1



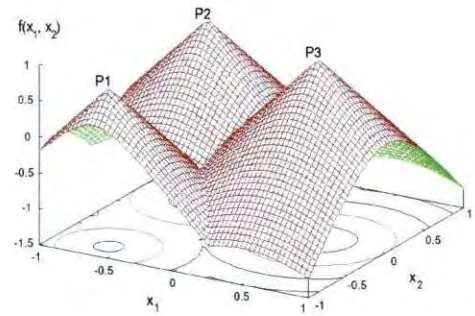
(b) Step 2



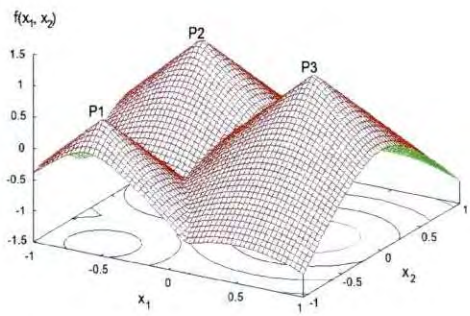
(c) Step 3



(d) Step 4

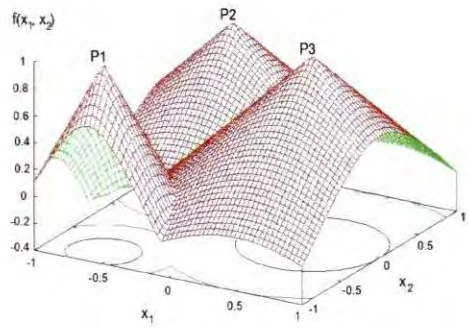


(e) Step 5

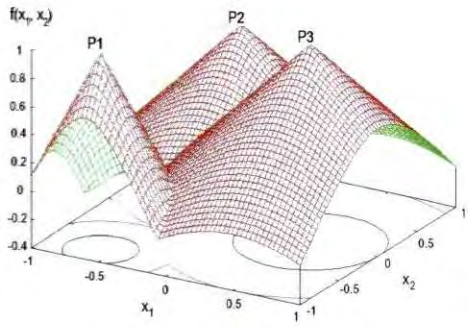


(f) Step 6

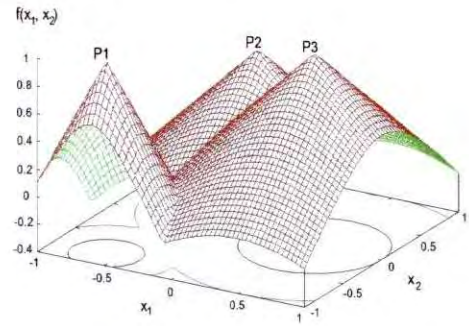
Figure 8.3: Scenario 3: Function landscapes



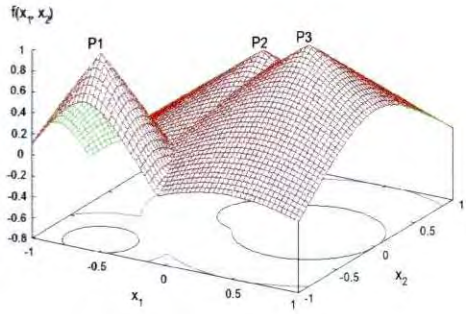
(a) Step 1



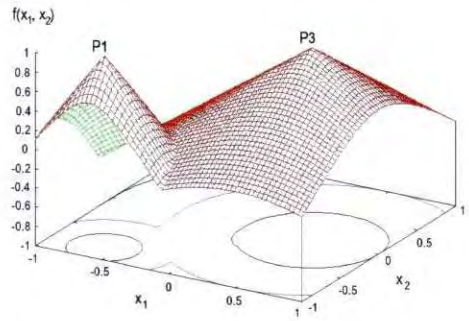
(b) Step 2



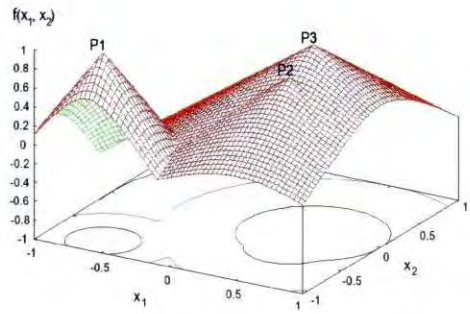
(c) Step 3



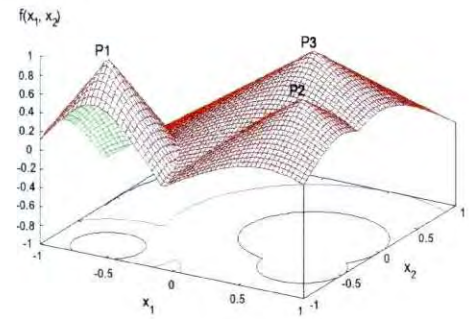
(d) Step 4



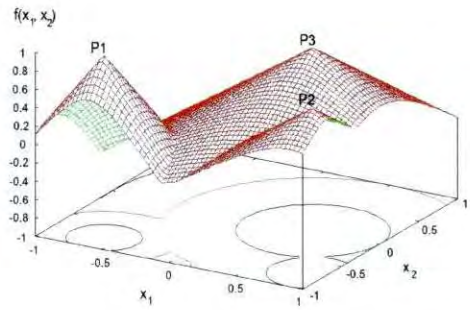
(e) Step 5



(f) Step 6



(g) Step 7



(h) Step 8

Figure 8.4: Scenario 4: Function landscapes

Scenario 2

A second test function was created where no peaks have disappeared and no new ones appear, but the number of peaks is increased to five. Changes in the objective function and therefore the positions of the optima, take place over six steps. Table 8.3 lists the settings of the function for each step. Figure 8.2 illustrates the function landscape at each step, indicating the peaks as $P1$ to $P5$. Movement of these peaks across the function landscape is described as follows:

Peak 1: The initial position of peak 1 is at $[-0.7, -0.8]$. Movement takes place parallel to the x_2-axis as the x_2-component is incremented by 0.2 at each step while the x_1-component remains the same. Movement stops at position $[-0.7, 0.2]$. The heights and radii of all peaks in this scenario remain the same.

Peak 2: Peak 2 moves parallel to the x_2-axis, from $[-0.7, 0.6]$ to $[0.3, 0.6]$, the x_1-component being incremented by 0.2 at each step.

Peak 3: Peak 3 moves along the x_1-axis, starting at position $[0, -0.3]$ to position $[0, 0.2]$. The x_2-component is incremented by 0.1.

Peak 4: This peak describes another movement parallel to the x_1-axis, but in the opposite direction from peaks 2 and 3. Movement starts at position $[0.8, -0.7]$ and stops at $[-0.3, -0.7]$. The x_1 component is decremented by 0.2 at each step.

Peak 5: Finally, peak 5 describes another movement parallel to the x_2-axis, but in the opposite direction from that of peak 1. The starting position is at $[0.6, 0.7]$ and movement stops at $[0.6, -0.3]$, meaning that the x_2-component is decremented by 0.2 at each step.

Table 8.4 presents the results. Similar to scenario 1, only stage 1 of the algorithm is used. The average number of function evaluations required to locate the initial optima, do not differ much from that of scenario 1. However, because the number of optima that are tracked, is increased to 5, the average number of function evaluations during the last 5 steps increases, but only to about half of the number required to find the initial optima. Therefore the same conclusion can be reached as in scenario 1.

Table 8.3: Scenario 2: Positions of 5 optima over 6 steps

Step	Peak 1				Peak 2				Peak 3				Peak 4				Peak 5			
	x_1	x_2	H	R	x_1	x_2	H	R	x_1	x_2	H	R	x_1	x_2	H	R	x_1	x_2	H	R
1	-0.7	-0.8	1	2	-0.7	0.6	1	2	0	-0.3	1	2	0.8	-0.7	1	2	0.6	0.7	1	2
2	-0.7	-0.6	1	2	-0.5	0.6	1	2	0	-0.2	1	2	0.6	-0.7	1	2	0.6	0.5	1	2
3	-0.7	-0.4	1	2	-0.3	0.6	1	2	0	-0.1	1	2	0.4	-0.7	1	2	0.6	0.3	1	2
4	-0.7	-0.2	1	2	-0.1	0.6	1	2	0	0	1	2	0.2	-0.7	1	2	0.6	0.1	1	2
5	-0.7	0	1	2	0.1	0.6	1	2	0	0.1	1	2	0	-0.7	1	2	0.6	-0.1	1	2
6	-0.7	0.2	1	2	0.3	0.6	1	2	0	0.2	1	2	-0.2	-0.7	1	2	0.6	-0.3	1	2

Table 8.4: Scenario 2: Results

Step	Average ♯ evaluations	Average ♯ optima	Offline error	Success rate
1	23068 ± 453	5 ± 0	8.48E-18 ± 3.30E-18	100%
2	11583 ± 11	5 ± 0	2.32E-17 ± 1.97E-17	100%
3	11578 ± 14	5 ± 0	2.81E-18 ± 5.84E-19	100%
4	11556 ± 12	5 ± 0	7.10E-18 ± 1.30E-18	100%
5	11566 ± 14	5 ± 0	2.70E-17 ± 2.88E-18	100%
6	11533 ± 13	5 ± 0	2.34E-18 ± 5.04E-19	100%

Scenario 3

Scenario 3 portrays a dynamic environment where the fitness of the optima undergo changes, but the positions stay the same. Optima are tracked over 6 steps, using only stage 1 of the algorithm, that is, no new peaks are located. Such changes may cause the global optimal position to shift from one peak to another. Table 8.5 lists the settings of the function for each step. Figure 8.3 illustrates the function landscape at each step, indicating the peaks as $P1$ to $P3$ to correspond with the column headings in Table 8.5, namely Peak 1, Peak 2 and Peak 3. Changes in the peaks are described as follows:

Peak 1: Peak 1 is positioned at $[-0.6, -0.8]$ throughout the run, but the height of the peak, H decreases from 1.5 to 0.5. The slope of the peak, R, stays the same at 2, meaning that the size of the base of the peak increases.

Peak 2: Peak 2 does not change during the run. The position remains at $[-0.5, 0.3]$, the height, H, is equal to 1 throughout, while the slope, R, remains at 2.

Peak 3: The position of peak 3 as well as the slope, R, remain at $[0.5, 0]$ and 2. The height, H, increases from 0.6 to 1.1.

Table 8.6 presents the results. The current global optima are also indicated. This scenario illustrates one of the benefits of tracking multiple optima in a multi-modal environment: If

a current suboptimal solution becomes the solution with the best fitness as the environment changes, no additional effort is required to keep track of the global optimum.

Table 8.5: Scenario 3: Three optima, stationary positions, fitness changes

Step	Peak 1				Peak 2				Peak 3			
	x_1	x_2	H	R	x_1	x_2	H	R	x_1	x_2	H	R
1	-0.6	-0.8	1.5	2	-0.5	0.3	1	2	0.5	0	0.6	2
2	-0.6	-0.8	1.3	2	-0.5	0.3	1	2	0.5	0	0.7	2
3	-0.6	-0.8	1.1	2	-0.5	0.3	1	2	0.5	0	0.8	2
4	-0.6	-0.8	0.9	2	-0.5	0.3	1	2	0.5	0	0.9	2
5	-0.6	-0.8	0.7	2	-0.5	0.3	1	2	0.5	0	1	2
6	-0.6	-0.8	0.5	2	-0.5	0.3	1	2	0.5	0	1.1	2

Scenario 4

Scenario 4 illustrates a situation where the positions of three optima change over eight steps. In the course of the movement, one of the peaks is obscured by a larger peak, but appears again when it moves out from under the higher peak. Table 8.7 shows the settings of the function for each step, and Figure 8.4 illustrates the function landscapes. Peaks are indicated as $P1$ to $P3$ to correspond to peak 1, peak 2 and peak 3 in the table. The movement of these peaks across the function landscape is described below:

Peak 1: The initial position of the peak is at $[-0.6, -0.8]$. The x_2-component remains the same, while the x_1-component is incremented by 0.2 at each step, resulting in a movement parallel to the x_2 axis, stopping at $[0.4, -0.8]$. The height and radius of the peak remains the same.

Peak 2: Movement of peak 2 commences at $[-0.5, 0.3]$. The x_1-component is incremented by 0.2 and the x_2-component by 0.1 at each step. The peak moves across the space towards $[0.5, 0.8]$ in the adjacent quadrant while the height and radius of the peak remains the

Table 8.6: Scenario 3: Results

Step	Average ♯ evaluations	Average ♯ optima	Offline error	Success rate	Global optimum
1	24699 ± 510	3 ± 0	1.37E-15 ± 1.16E-15	100%	Peak 1
2	6580 ± 9	3 ± 0	8.77E-18 ± 4.48E-18	100%	Peak 1
3	6571 ± 9	3 ± 0	5.21E-18 ± 9.27E-19	100%	Peak 1
4	6565 ± 10	3 ± 0	5.18E-18 ± 9.02E-19	100%	Peak 2
5	6553 ± 9	3 ± 0	1.35E-17 ± 8.09E-18	100%	Peak 2,3
6	6559 ± 10	3 ± 0	9.14E-18 ± 1.66E-18	100%	Peak 3

same. During the movement peak 2 moves underneath peak 3 and disappears from view, but the next step shows the peak starting to appear on the other side of the larger peak. In subsequent steps the peak can be clearly seen.

Peak 3: The starting position of peak 3 is at $[0.5, 0]$. The x_1-component is decremented by 0.1 at each step while the x_2-component does not change. The peak moves along the x_1-axis and stops at the centre of the search space at $[0, 0]$.

Scenario 4 is used in two experiments. In the first experiment, only stage 1 of the algorithm is activated. Table 8.8 presents the results, showing that the algorithm is capable of merging subswarms tracking optima which might have disappeared. Results show that only two optima are located during steps 5 and 6. In step 5 two subswarms are merged. In step 6 only two optima are tracked, which is reflected in the results showing a significantly smaller average number of function evaluations.

The second experiment using this scenario utilizes the full capacity of the dynamic vector-based PSO. Where stage 1 of the algorithm tracks existing optima, merging subswarms if necessary, stage 2 explores the problem space to detect new optima. Coverage of the entire search space is implied, requiring additional computational complexity. Three peaks are tracked over 8 steps. Table 8.9 presents the results.

Results show that 3 optima are located and tracked over the next 3 steps. One peak then disappears and over the next 2 steps only 2 optima are tracked. Step 6 shows a change in the landscape where a small peak appears in the next step. This peak is located 86.67% of the time. In the last step the peak is located in all runs of the algorithm. However, when stage 2 of the algorithm is activated, the average number of function evaluations exceeds that of the initial phase to locate optima. It would seem that the dynamic vector-based PSO is less efficient than re-optimization if a procedure to detect new optima is included.

Discussion

The four scenarios tested in this section demonstrated strengths and weaknesses of the vector-based PSO. The algorithm is capable of tracking existing optima in an accurate and efficient manner. Optimal solutions that disappear are handled in an elegant and efficient manner. Results show that tracking of optima requires less computational complexity than re-optimization. However, exploring the entire search space for new optima that may appear, is computationally more expensive that re-optimization.

Table 8.7: Scenario 4: Positions of 3 optima over 8 steps - 1 peak obscured and appears again

Step	Peak 1				Peak 2				Peak 3			
	x_1	x_2	H	R	x_1	x_2	H	R	x_1	x_2	H	R
1	-0.6	-0.8	1	2	-0.5	0.7	0.8	1	0.5	0	1	1
2	-0.6	-0.8	1	2	-0.3	0.5	0.8	1	0.5	0	1	1
3	-0.6	-0.8	1	2	-0.1	0.3	0.8	1	0.5	0	1	1
4	-0.6	-0.8	1	2	0.1	0.1	0.8	1	0.5	0	1	1
5	-0.6	-0.8	1	2	0.3	-0.1	0.8	1	0.5	0	1	1
6	-0.6	-0.8	1	2	0.5	-0.3	0.8	1	0.5	0	1	1
7	-0.6	-0.8	1	2	0.7	-0.5	0.8	1	0.5	0	1	1
8	-0.6	-0.8	1	2	0.9	-0.7	0.8	1	0.5	0	1	1

Table 8.8: Scenario 4: Results using stage 1 of the VBPSO algorithm

Step	Average ♯ evaluations	♯ optima in search space	Average ♯ optima located	Offline error	Success rate
1	23899 ± 387	3	3 ± 0	2.61E-05 ± 2.61E-05	100%
2	6659 ± 9	3	3 ± 0	1.03E-17 ± 1.90E-18	100%
3	6634 ± 5	3	3 ± 0	1.03E-17 ± 1.90E-18	100%
4	6652 ± 7	3	2.8 ± 0.06	1.09E-17 ± 2.00E-18	93.33%
5	6191 ± 134	2	2 ± 0	1.40E-17 ± 2.33E-18	100%
6	4326 ± 2	2	2 ± 0	1.40E-17 ± 2.33E-18	100%
7	4326 ± 2	3	2 ± 0	1.40E-17 ± 2.33E-18	66.67%
8	4324 ± 2	3	2 ± 0	1.40E-17 ± 2.33E-18	66.67%

Table 8.9: Scenario 4: Results using stage 1 and stage 2 of the VBPSO algorithm

Step	Average ♯ evaluations	♯ optima in search space	Average ♯ optima located	Offline error	Success rate
1	24441 ± 504	3	3 ± 0	4.19E-15 ± 4.15E-15	100%
2	31692 ± 450	3	3 ± 0	5.59E-09 ± 5.59E-09	100%
3	34043 ± 391	3	2.92 ± 0.04	1.33E-05 ± 1.22E-05	97.33%
4	33443 ± 763	3	2.48 ± 0.07	4.69E-06 ± 3.97E-06	82.67%
5	28151 ± 354	2	2 ± 0	3.71E-12 ± 3.71E-12	100%
6	28554 ± 519	2	2 ± 0	1.69E-10 ± 1.35E-10	100%
7	29914 ± 657	3	2.54 ± 0.07	7.87E-13 ± 7.84E-13	84.67%
8	31750 ± 479	3	3 ± 0	1.75E-06 ± 1.33E-06	100%

8.3 Comparing the performance of the dynamic vector-based PSO to that of the dynamic species-based PSO

This section compares the performance of the vector-based PSO to that of the dynamic species-based PSO developed by Parrott and Li [70]. The algorithm is described in chapter 7. The appeal of the species-based PSO lies in its elegance and simplicity. Since species seeds are established and subswarms reformed each time particle positions are updated, the algorithm provides a natural way in which optima can be tracked. Therefore the basic dynamic species-based PSO requires minimal modification. The version tested in this section implemented a procedure to re-evaluate each particle's personal best fitness value before updating particle positions. Introducing a maximum species population size and re-initializing redundant particles at random positions in the solution space was not attempted, as the population size is small and species radii differ from one another and over time.

8.3.1 Experimental settings and results

The dynamic species-based PSO was tested using the four scenarios described in the previous section. In all cases the initial population size was set to 30. For each experiment the algorithm was run 50 times. The dimensions of the search space, settings used by the update equation and the manner in which random positions of the initial particles are generated, are similar to the experiments conducted with the dynamic vector-based PSO. At each step the process to establish species seeds, and updating the velocity and particle positions once, were repeated 500 times. The species-based PSO uses a species radius that has to be set in advance. For each scenario the size of the species radius was estimated by running the first step of the algorithm a number of times with different species radius values and selecting the value yielding the best performance of the algorithm.

Scenario 1

A test function where the positions of 3 peaks change over 6 steps was created. A description of this scenario was given in section 6.7. Table 8.1 lists the settings of the function for each step, while Figure 8.1 shows the function landscape at each step. For these experiments the species radius was set to 0.7.

Table 8.10 presents the results. The first step has a success rate of 100% that decreases

over steps 2 to 4 to 83.33%, 76.67%, and 70%. The corresponding function landscapes as illustrated in Figure 8.1 show that distances between the optima decrease in these cases, that is, the species radii also decreases. Thus the decrease in performance can be ascribed to the fact that, in a multimodal landscape, different species have different species radii. This effect is more pronounced in dynamic environments, the degree depending on the objective function. The static setting of the species radius in the dynamic species-based PSO does not address this problem. Therefore the dynamic vector-based PSO has an advantage as each niche (corresponding to a species in the dynamic species-based PSO) is assigned a unique niche radius in the initial step of the algorithm, while smaller subswarms are used to track the optima.

Table 8.10: Scenario 1: Results for the dynamic species-based PSO

Step	Average ♯ evaluations	Average ♯ optima	Average ♯ extra species	Offline error	Success rate
1	15030 ± 0	3 ± 0	0.3 ± 0.08	5.22E-03 ± 1.2E-03	100%
2	30090 ± 0	2.5 ± 0.07	0 ± 0	5E-03 ± 1.32E-03	83.33%
3	45150 ± 0	2.3 ± 0.07	0 ± 0	5.27E-03 ± 1.3E-03	76.67%
4	60210 ± 0	2.1 ± 0.04	0.1 ± 0.04	6.11E-03 ± 1.41E-03	70%
5	75270 ± 0	3 ± 0	0 ± 0	1.28E-02 ± 1.84E-03	100%
6	90330 ± 0	3 ± 0	0.1 ± 0.04	1.86E-02 ± 2.1E-03	100%

Scenario 2

This scenario describes a similar situation to scenario 1, but the number of optima is increased to 5. This scenario was described in section 6.7. Table 8.3 lists the settings of the function for each step, while Figure 8.2 shows the function landscape at each step. For these experiments the species radius was set to 0.6.

Table 8.11 presents the results. Again, decreasing performance over the last 4 steps can be ascribed to a changing species radius. The explanation is similar to that of scenario 1,

emphasizing the fact that the dynamic VBPSO outperforms the dynamic SPSO when niche radii change in a dynamic environment.

Table 8.11: Scenario 2: Results for the dynamic species-based PSO

Step	Average ♯ evaluations	Average ♯ optima	Average ♯ extra species	Offline error	Success rate
1	15030 ± 0	5 ± 0	0 ± 0	1.51E-03 ± 1.84E-04	100%
2	30090 ± 0	5 ± 0	0 ± 0	5.74E-03 ± 8.86E-04	100%
3	45150 ± 0	4.9 ± 0.04	0 ± 0	7.88E-03 ± 1.04E-03	98%
4	60210 ± 0	4.8 ± 0.06	0.8 ± 0.09	1.50E-02 ± 1.27E-03	96%
5	75270 ± 0	4.9 ± 0.04	1.3 ± 0.14	2.17E-02 ± 1.49E-03	98%
6	90330 ± 0	4.6 ± 0.07	0.3 ± 0.07	1.82E-02 ± 1.44E-03	92%

Scenario 3

In this scenario only the heights of the peaks change over six steps. Positions of the optima remain the same. This scenario was described in section 6.7. Table 8.5 lists the settings of the function for each step, while Figure 8.3 shows the function landscape at each step. For these experiments the species radius was set to 0.7.

Table 8.12 presents the results, showing a 100% success rate in all cases. Good performance of the dynamic species-based PSO can be expected in this situation as the peaks do not move and the species radii do not change. If the niche radius setting yields good performance for the first step, the same can be expected for subsequent steps. The performance even gets better, as no extra species are identified in steps 3 to 6, meaning that, at that stage, all particles have moved into the regions surrounding the species seeds.

Table 8.12: Scenario 3: Results for the dynamic species-based PSO

Step	Average ♯ evaluations	Average ♯ optima	Average ♯ extra species	Offline error	Success rate
1	15030 ± 0	3 ± 0	0.7 ± 0.11	1.07E-02 ± 1.71E-03	100%
2	30090 ± 0	3 ± 0	0.2 ± 0.06	6.07E-03 ± 1.45E-03	100%
3	45150 ± 0	3 ± 0	0 ± 0	3.30E-03 ± 1.02E-03	100%
4	60210 ± 0	3 ± 0	0 ± 0	7.85E-04 ± 2.26E-04	100%
5	75270 ± 0	3 ± 0	0 ± 0	6.09E-04 ± 2.03E-04	100%
6	90330 ± 0	3 ± 0	0 ± 0	5.84E-04 ± 2.04E-04	100%

Scenario 4

In this scenario the problem is such that 3 optima move across the landscape over 8 steps. In the course of the movement one peak is obscured by an adjacent peak and appears again on the other side of the peak. This scenario was described in section 6.7. Table 8.7 lists the settings of the function for each step, while Figure 8.4 shows the function landscape at each step. For these experiments the species radius was set to 0.7.

Table 8.13 presents the results. Good performance is reported for the first two steps, but in the next two steps the performance declines. These results coincide with diminishing distances between two of the peaks and the fact that the species radius remains the same throughout. In steps 5 and 6 both optima are located, as well as an average of 1.1 extra optima, indicating that the dynamic species-based PSO does not successfully reflect the situation when peaks disappear. In steps 7 and 8 the peak that was obscured, surfaces again, but the success rate is similar to that of step 4. Locating new optima cannot be expected from this version of the dynamic species-based PSO, as the re-distribution of redundant particles have not been implemented. However, it can be expected of the algorithm to reflect the situation when a peak is obscured. From these results and the results reported by the dynamic vector-based PSO it can be concluded that the dynamic vector-based PSO performs better than the dynamic species-based PSO in situations where the number of peaks in the function landscape decreases.

Table 8.13: Scenario 4: Results for the dynamic species-based PSO

Step	Average ♯ evaluations	Average ♯ optima	Average ♯ extra	Offline error	Success rate
1	15030 ± 0	3 ± 0	0.7 ± 0.11	9.02E-03 ± 1.50E-03	100%
2	30090 ± 0	3 ± 0	0.2 ± 0.06	9.50E-03 ± 1.67E-03	100%
3	45150 ± 0	2.9 ± 0.04	0.2 ± 0.06	1.07E-02 ± 1.84E-03	96.67%
4	60210 ± 0	2.8 ± 0.06	0.3 ± 0.07	9.73E-03 ± 1.74E-03	93.33%
5	75270 ± 0	2 ± 0	1.1 ± 0.1	9.71E-03 ± 1.74E-03	100%
6	90330 ± 0	2 ± 0	1.1 ± 0.1	9.71E-03 ± 1.74E-03	100%
7	105390 ± 0	2.8 ± 0.06	0.3 ± 0.07	9.80E-03 ± 1.76E-03	93.33%
8	120450 ± 0	2.8 ± 0.06	0.3 ± 0.07	9.80E-03 ± 1.76E-03	93.33%

8.3.2 Discussion

From the results obtained when testing both the dynamic vector-based PSO and the dynamic species-based PSO using similar scenarios, it was concluded that the dynamic vector-based PSO perform better than the dynamic species-based PSO in cases where the positions of peaks in the landscape change. When only the heights but not the positions of the peaks change, the performance is the same.

8.4 Sensitivity of the dynamic vector-based PSO to changes in severity

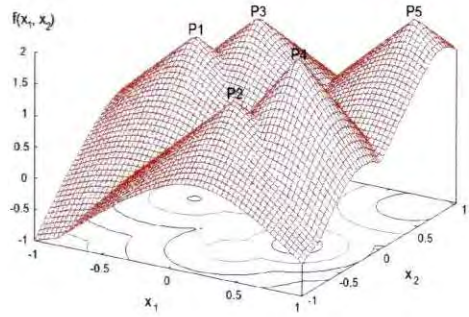
A changing problem landscape implies changes in the positions of optima and suboptima, the value of the objective function at those positions, or both. *Spatial severity*, referred to by the parameter ζ, quantifies the amount of change in the position of an optimum in problem space before the next optimization effort. In section 8.2, the performance of the dynamic vector-based

PSO was studied by setting up a number of changing environments. All of these experiments were conducted in a two-dimensional search space in the range $x_1, x_2 \in [-1.0, 1.0]$. Changes in the positions of the optima were effected by incrementing or decrementing the positions on the two axes, x_1 and x_2 by either 0, 0.1 or 0.2. In all scenarios the spatial severity, ζ, was never more than 0.28 (0.2 on both axes).

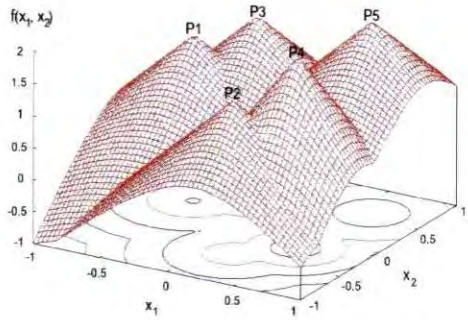
Experimental setup and results

This section investigates the influence of a range of severity values on the performance of the dynamic vector-based PSO. A function containing 5 peaks was created using the moving peaks benchmark as described by equation (7.3) [25]. One of the peaks moves diagonally through the landscape. Figure 8.5 illustrates the progression of the peak through a 5-peak landscape where the severity is 0.28. Peaks are numbered $P1$ to $P5$. Peak $P5$ progresses through the landscape while the other peaks remain stationary.

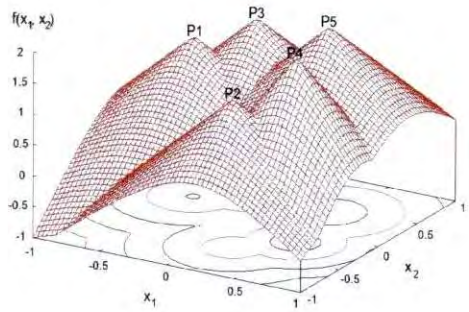
Eight experiments were conducted where the dynamic vector-based PSO tracks optima through the search space. For all experiments the starting position of peak $P5$ is at $[0.8, 0.8]$. Initially, the severity was set to 0.28 and increased by an increment of 0.1 on both axes for each experiment. For each value of ζ the algorithm was run 50 times and average results calculated. Stage 2 of the algorithm was disabled as no peaks appear or disappear. The number of steps required to move through the search space depends on the severity. Table 8.14 presents the experimental results. Figure 8.6 shows the average number of optima located, plotted against the severity.



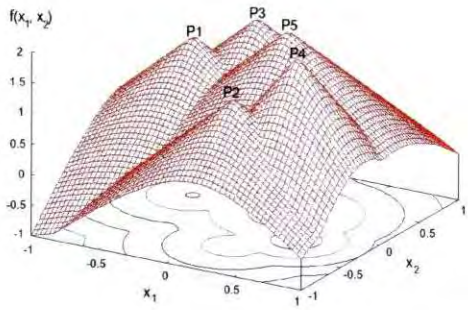
(a) Step 1



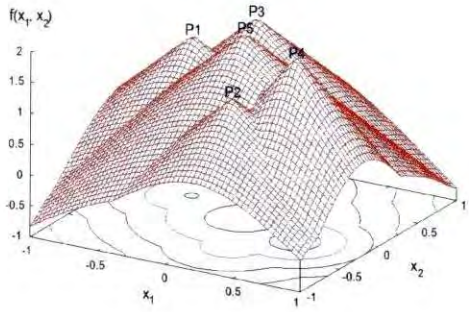
(b) Step 2



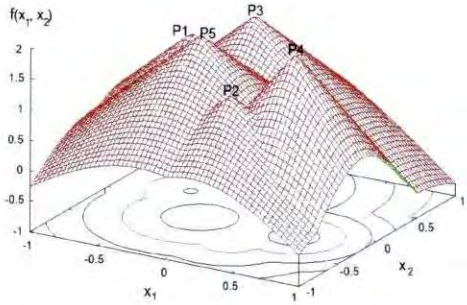
(c) Step 3



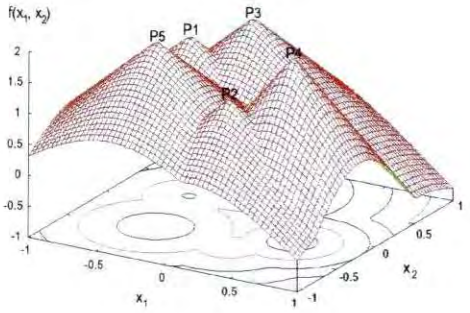
(d) Step 4



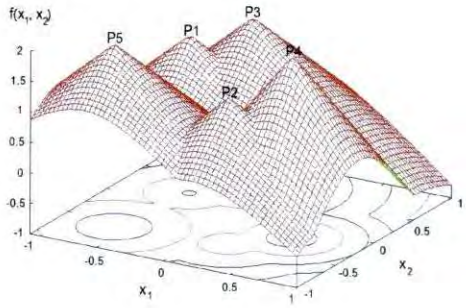
(e) Step 5



(f) Step 6



(g) Step 7



(h) Step 8

Figure 8.5: Movement of one peak through a landscape containing 5 peaks

Table 8.14: Performance of the dynamic vector-based PSO with increased severity

Experiment	x_1 increment	x_2 increment	♯ Steps	Severity	Average ♯ evaluations (step 1)	Average ♯ evaluations (step 2-8)	Average ♯ optima (step 1)	Average ♯ optima (step 2-8)	Success rate
1	0.2	0.2	9	0.28	26854 ± 384	10966 ± 6	5 ± 0	5 ± 0	100%
2	0.3	0.3	6	0.42	26033 ± 263	10854 ± 80	5 ± 0	5 ± 0	100%
3	0.4	0.4	5	0.57	27928 ± 397	10828 ± 70	5 ± 0	4.95 ± 0.03	87.5%
4	0.5	0.5	4	0.71	26415 ± 325	10238 ± 148	5 ± 0	4.27 ± 0.08	33.33%
5	0.6	0.6	3	0.85	26633 ± 295	10435 ± 134	5 ± 0	4.37 ± 0.09	25%
6	0.7	0.7	3	0.99	27521 ± 499	10190 ± 148	5 ± 0	4.35 ± 0.05	15%
7	0.8	0.8	3	1.13	26149 ± 414	10204 ± 149	5 ± 0	4.14 ± 0.05	15%
8	0.9	0.9	2	1.27	28621 ± 536	11048 ± 6	5 ±0	4.1 ± 0.04	10%

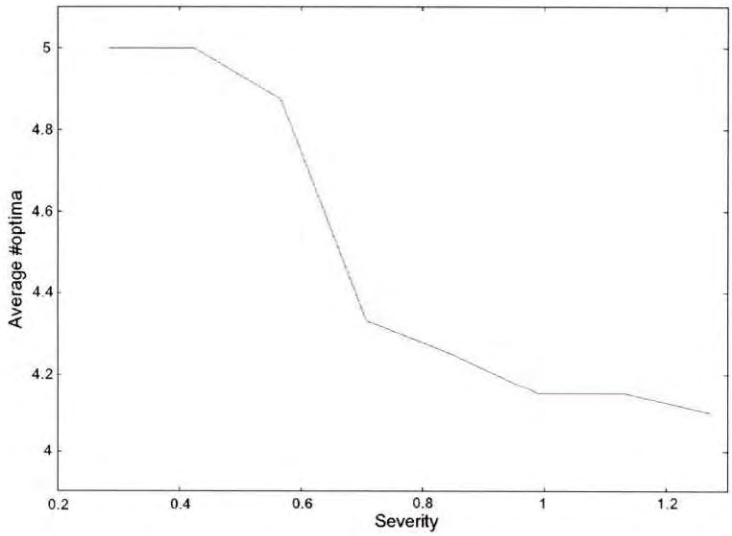


Figure 8.6: Average #optima versus severity for 5-peak function

Discussion

Intuitively, successful tracking is associated with a severity smaller than the niche radius of a peak, as the dynamic vector-based PSO retains previous optimal positions from where a small subswarm moves towards the new optimum. It stands to reason that, to track the new optimum successfully, the previous optimal position must still be on the slope leading to the new optimal position, that is, within the niche radius of the new optimum. However, niche radii cannot always be calculated accurately. In addition, some areas of the slope of a peak extends further than other areas, especially if the moving peak progresses through a landscape containing several other peaks. Results for the experiments conducted with the settings described in this section show a sharp decline in the success rate between severity values of 0.57 and 0.71 (distances between successive peaks). Careful scrutinizing of the function landscapes illustrated in Figure 8.5 reveals such severity values to be approximately equal to the upper bound of the niche radius. Therefore, a new optimum will still be tracked successfully, even if the position has moved to the boundary of the previous niche. To conclude, the dynamic vector-based PSO can be described as remarkably robust when tracking optima in a dynamic function landscape.

8.5 Conclusion

This chapter demonstrated that the parallel vector-based particle swarm optimizer can be modified to track multiple moving optima in a number of selected dynamic environments. Scenarios were chosen to represent different situations. Good results were reported when existing optima were tracked. The computational complexity required to track existing optima was much less than that of locating the optima in the first place. However, results are highly problem-dependent, and locating new peaks is not always successful. It is also computationally more expensive to search the entire problem space for these peaks, and in some cases re-optimization of the search space might have to be considered.

Comparing the dynamic vector-based PSO to the dynamic species-based PSO showed the dynamic vector-based PSO to be superior in all cases. To conclude the chapter, the sensitivity of the dynamic vector-based PSO to changes in severity was investigated. Results emphasized the dynamic vector-based PSO to be an efficient and robust algorithm.

Chapter 9

Conclusion

This chapter briefly summarizes the contributions and results of this thesis and discusses some directions for future research.

9.1 Summary

The principle focus of this thesis was directed towards the development of a PSO niching strategy that would address deficiencies and problems encountered by niching or speciation algorithms in general. The original single-solution PSO searches for one or more global optima, a task where traditional optimization methods may fail if the function landscape contains many sub-optimal solutions. One of the strengths of the single-solution PSO is the ability to direct the swarm away from a sub-optimal solution to a better solution. However, if a problem can be described by means of a multimodal function, a range of solutions is much more useful than a single solution. If the "best" solution is not viable for some reason, a "second-best" solution may suffice, especially since the value of the function at different optimal positions often differ very slightly or not at all. Niching algorithms yield such a range of solutions where each solution corresponds to an optimal position in the search space. Many niching algorithms use some strategy to divide a swarm into subswarms where each subswarm is expected to locate an optimal or suboptimal solution [13] [57]. When such subswarms are optimized, either sequentially or in parallel, multiple optima will only be located if the tendency of a subswarm to move away from a sub-optimal solution to a better solution is suppressed.

Chapters 2 and 3 presented an overview of optimization and the basic PSO paradigm.

Niching was defined in chapter 4 while a number of existing niching approaches was described and discussed. Problems with each of these approaches were identified so that particular attention could be given to these aspects in the design of a new strategy. Existing niching algorithms can be categorized as follows:

Algorithms that transform the landscape: To locate multiple optima, the problem landscape is transformed after an optimum has been located. The well-known "objective function stretching" approach of Parsopoulos and Vrahatis [71] [72] flattens or "stretches" the landscape where a minimum has been located so that the next minimum will now be at the global best position (minimization is assumed). Such an algorithm is per definition sequential. However, as described in chapter 4, false minima and misleading gradients can be introduced.

Algorithms using a niche radius: A popular approach used by niching algorithms is the division of a swarm into subswarms [13] [14] [16] [57]. Since the number, shapes, sizes and placing of niches in the function landscape are unknown, prior knowledge of the search space is often required to produce significant results. In addition to parameters like the problem range and the number of particles, some metric is needed to indicate the size and position of the region or niche in problem space where a subswarm resides. A number of algorithms use a niche radius, the distance between the current candidate solution and the boundary of the niche, to demarcate a niche. The species-based PSO sets a niche radius in advance [57], while NichePSO calculates the radius during execution of the algorithm [13] [14] [16]. An accurate niche radius, obtained by knowledge of the problem space or a number of test runs, yields good results if the shape and placing of the niches are relatively symmetrical. However, if the landscape is more convoluted with a variety of niche shapes and sizes, performance degrades. NichePSO does not need a pre-specified niche radius, but relies heavily on merging of niches, which requires fine-tuning of parameters as too many or too few niches may merge.

Algorithms relying on distances between particles: Attempts to avoid pre-specifying of a niche radius gave rise to the adaptive niching PSO (ANPSO) [5] and the fitness Euclidian-distance ratio-based PSO (FER-PSO) [58]. Good performances were reported, but both algorithms were computationally expensive due to the calculation of distances between particles (ANPSO) and ratios required by the FER-PSO.

The most imported objective of this study was to develop a new niching strategy that would

address the problems identified above. Although some parameters have to be set in advance to facilitate niching, it remained a challenge to develop a PSO niching algorithm requiring minimal prior knowledge of the function landscape. An additional challenge concerned the extension of the principles on which the original PSO algorithm rests to facilitate niching, resulting in an efficient and elegant solution. Such a strategy should also be robust enough to operate successfully in convoluted function landscapes where the shapes, sizes and placing of niches differ considerably.

The development of the vector-based PSO was described in detail in chapter 5. Vectors are used extensively in the original PSO. The position of an individual in the particle swarm is given by a position vector. Two other positions are also associated with each particle, namely the best position found so far by the particle itself (*pbest*) and the best position of the entire swarm (*gbest*). When updating particle positions, the original PSO uses vector addition to calculate new random positions. To facilitate niching, another vector operation, the vector dot product, is utilized. The technique rests on the assumption that a positive dot product indicates two vectors pointing in the same direction while a negative dot product is the result of two vectors pointing in opposite directions. The technique is explained in detail in chapter 5. Using this principle, niche boundaries can be calculated and the number of optima in an unknown search space be derived, each with its own niche radius. This technique addresses one of the major problems of niching algorithms, namely that of calculating niche radii for an unknown function landscape.

The vector-based algorithm progressed through three stages to produce an algorithm that locates niches sequentially and optimizes them in parallel. All three algorithms underwent extensive testing using one- and two-dimensional benchmark functions, presented in chapter 6. Care was taken to choose functions where the shapes and sizes of the niches differed and the positions of the optima were not distributed symmetrically throughout the search space. Thus the robustness of the algorithms could be assessed. Chapter 6 also presents a comparative study that was conducted to compare the vector-based PSO with the species-based PSO [57] and NichePSO [13] [14] [16]. The performance of these two algorithms degraded considerably for such functions. However, the vector-based PSO performed well throughout, even with multimodal functions that are perceived as difficult to optimize.

Since niches are not always symmetrical around the current best position (the *neighbourhood best*) in the niche, false niches are formed that have to be merged with genuine niches during optimization. Merging requires a problem-dependent merging parameter, the granularity, that has to be set in advance. The performance of three two-dimensional benchmark functions have

been evaluated for a range of granularity values. Results showed good performances where the granularity is less than the smallest interniche distance. Optima where the interniche distances are larger than the granularity, will still be located. Stated differently, in an unknown function landscape the vector-based PSO will locate all or most of the optima where the interniche distances are larger than the granularity.

Chapter 6 also incorporated a scalability study. A small subset of scalable multimodal functions were tested for a number of search spaces containing fewer than one hundred optima, using a range of swarm sizes. Results indicated that the increase in the optimal swarm size required to ensure that all optima in a demarcated search space are located, is smaller than expected.

The vector-based PSO was developed to locate multiple optima in a static environment. However, problems often require tracking of multiple optima that changes over time. Chapter 6 investigated the behaviour of the vector-based PSO technique in a changing environment. The vector-based PSO is adapted and extended to track multiple optima over time. A number of scenarios illustrating a variety of dynamic changes, was set up and tested. Results indicated that existing optima were tracked successfully, but locating new peaks was not always successful. It is also computationally more expensive to search the entire problem space for these peaks, and in some cases re-optimization of the search space might have to be considered. An investigation into the influence of severity on dynamic changes showed that the vector-based PSO could track optima successfully for severity values up to the upper bound of the current niche radius.

Objectives of this thesis were stated in the introduction. The extent to which these objectives have been met, is summarized as follows:

- The essential building blocks of the basic PSO paradigm was extended to facilitate niching by using another vector operation. By calculating the dot product of the two velocity vectors of a particle, niche boundaries could be determined.

- A new niching algorithm, the vector-based PSO, was developed and tested extensively. The algorithm proved to be efficient, elegant and robust. Results showed that the vector-based PSO located a very high percentage of multiple optimal solutions when tested with a variety of multimodal functions. Objective functions with asymmetrically shaped and placed functions also yielded good results, emphasizing the robustness of the algorithm. Finally, the algorithm can be described as elegant as it is small, simple, and uses vector operations, an integral part of the original PSO.

- Minimal prior knowledge of the function landscape is required as candidate solutions and niche radii are calculated. Therefore, differing niche sizes do not effect the success of the algorithm. A niching parameter, the granularity, is set beforehand. However, the granularity is not very sensitive and is an indication of the minimum distance between optima. Therefore, the user can decide how fine-grained the niching must be.
- Different niching approaches have been compared using a diverse problem set. The vector-based PSO outperformed the species-based PSO and NichePSO in cases where the function landscape was asymmetrical and convoluted.
- A scalability study of the vector-based PSO was conducted, showing that the algorithm scales well for a small set of multimodal functions in one to four dimensions.
- The vector-based PSO was extended to incorporate dynamic optimization problems.

9.2 Future work

This section briefly summarizes new directions for future research suggested by the work presented in this thesis:

- The vector-based PSO uses the dot product of two vectors to determine niche boundaries. The algorithms developed in this study utilize the dot product as positive or negative only. An investigation into the range of specific values of dot products may yield further information on the characteristics of the function landscape.
- The study of niching in dynamic environments using the vector-based PSO can be extended to include more benchmark functions, more scenarios and some practical implementations, e.g. training of neural network ensembles.
- Extend the dynamic VBPSO to automatically detect environment changes to remove its current dependence on knowledge of the change frequency.