Chapter 6

A Comparative Analysis of Niching Techniques

This chapter presents an empirical comparison between the newly introduced *nbest* and NichePSO algorithms and two existing GA niching techniques.

6.1 Introduction

A plethora of evolutionary niching and speciation techniques are in existence. Chapter 2 presented a summary of some of the more well-known variations. All the presented techniques can be categorized as being either *sequential* or *parallel* niching techniques. Sequential niching locates niches in serialized runs of the same algorithm. Each run of the algorithm that successfully locates a niche/solution, modifies the fitness function to keep subsequent runs from unnecessarily duplicating search efforts. Parallel niching implements measures to concurrently identify and maintain niches (see section 3.2 for a more complete discussion). Both the *nbest* and NichePSO algorithms, introduced in the previous chapters, are parallel niching techniques. The *nbest* algorithm uses

- spatial particle neighborhoods and
- an alternative formulation of the fitness function

to concurrently find and maintain niches, while NichePSO uses *subswarms*.

The introduction of new algorithms in a research field necessitates a comparative analysis to determine whether they offer an advantage, or can be considered as alternatives to existing techniques. Since no existing, unique niching techniques exist in the PSO field, such a study would not be possible. The objective function stretching optimizer introduced by Parsopoulos *et al* is consciously excluded here, for reasons set out in section 3.4. Consequently, it seems appropriate to compare the new PSO techniques to well-known GA based niching techniques. It should be noted that the goal of this study was to develop new, unique PSO based solutions to niching problems. A number of authors have undertaken studies where existing evolutionary optimization approaches were re-factored for PSOs. As was found in chapter 3, this was not possible for GA niching techniques. They cannot be directly mapped to the PSO, due to differences in the behavior of the GA and PSO. These differences were analyzed in section 3.5. It was therefore necessary to develop techniques specifically suited to the dynamics of the PSO. In the rest of this chapter, a comparison is presented between PSO and GA based niching techniques.

6.2 The Algorithms

In order to form a general base for comparison, the PSO niching algorithms are compared to both a

- sequential, and a
- parallel GA niching technique.

As an example of sequential niching, Beasley *et al*'s sequential niching (SN) technique, described in section 3.3.3, is used. SN uses repeated runs of a normal generational genetic algorithm to locate multiple solutions [4]. After each run of a simple GA, the fitness function is adapted to reflect the position of the recently located solution. Subsequent runs of the GA with the modified fitness function avoid areas in the search space where solutions have already been found, forcing the optimizer to explore unknown sections of the search space. SN has been criticized, the biggest concern being that the derating

modifications made to the fitness function may conceal other optima [62]. Never the less, SN is the only sequential GA niching technique.

As a parallel niching technique, deterministic crowding (DC), presented in section 3.3.5 is used. DC is a replacement strategy that maintains several sets of similar individuals over a number of generations [61]. DC's custom selection policy only replaces offspring in a next generation when they perform better (fitness-wise) than their parents. A phenotypic similarity metric is used to quantify similarity between parents and offspring. As a parallel niching technique, DC is preferred over the more well-known fitness sharing [32], as it does not directly modify the fitness evaluation, i.e. it is not an explicit niching technique such as SN.

6.3 Experimental Setup

6.3.1 Test Problems

To test the different niching techniques, several functions presented in previous chapters were used. The following systems of equations, because of their irregular spacing of maxima, were used:

- System S3, defined in equation (4.15) on page 70, illustrated in figure 4.3.
- System S4, defined in equation (4.13) on page 68, illustrated in figure 4.2.
- System S5, defined in equation (4.21) on page 74, illustrated in figure 4.4(b).

The following multimodal functions, defined in chapter 5, are used (see page 95):

- Function F1, defined in equation (5.6), illustrated in figure 5.3(a).
- Function F2, defined in equation (5.7), illustrated in figure 5.3(b).
- Function F3, defined in equation (5.8), illustrated in figure 5.3(c).
- Function F4, defined in equation (5.9), illustrated in figure 5.3(d).
- Function F5, defined in equation (5.10), illustrated in figure 5.4.

6.3.2 Parameter Settings

For each of the *nbest*, NichePSO, SN and DC algorithms, 30 simulations were performed on each of the test problems outlined in section 6.3.1. The following sections describe GA and PSO parameter settings respectively.

GA Setup

For all experiments, populations consisting of 20 individuals were used. For Beasley *et al*'s SN algorithm, the following settings, as used in [4], were chosen:

Parameter	Value
p_c	0.9
p_m	0.01

where p_c and p_m signify the probability of whether the crossover and mutation operators are applied. A single-point crossover operator was used. As selection operator, *stochastic universal sampling* (SUS) is used, as suggested in [62]. One-dimensional problems used a 30-bit chromosome representation. For two-dimensional problems, two chromosomes of 15-bits each were used. The *halting window* approach described in [4], was used to terminate the algorithm. The approach monitors the average fitness of a population at each generation. If the average fitness has not improved on the fitness reported h generations earlier, the algorithm is terminated. For all runs of the SN algorithm, a halting window of $h = 20$ was used. Apart from this control setting, a maximum number of 2000 iterations was allowed. To determine the niche radius (see section 3.3.3), the method suggested by Beasley *et al* was used (originally suggested by Deb [20]). For a d-dimensional problem with l optima, the niche radius r was calculated as

$$r = \frac{\sqrt{d}}{2 \times \sqrt[d]{l}} \tag{6.1}$$

This technique assumes that fitness function parameters are normalized to $[0, 1]$. An exponential derating function G_e,

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

was used (given in section 3.3.3, repeated here for clarity); $\mathbf{x}$ is an individual's phenotypic representation, $\mathbf{x}^*$ represents the best individual located using a particular generation's phenotype, and $\|\cdot\|$ is the Euclidean distance defined in each problem's search space.

Mahfoud's DC uses an internal selection scheme (see section 3.3.5). Crossover and mutation probabilities were set at

Parameter	Value
p_c	1.0
p_m	0.01

since the DC algorithm favors a very lower mutation probability and a high crossover probability [61]. DC also used a halting window-based termination criterion of $h = 20$, and a maximum number of generations of $g_{max} = 2000$.

PSO Setup

For all experiments, swarms consisting of 20 particles were used. A halting window approach, similar to that presented above was implemented, assuming that for NichePSO, the halting window conditions were applied to all swarms, and the main swarm was empty. The inertia weight was linearly scaled from 0.7 to 0.1, over a maximum of 2000 iterations of the respective algorithms. Coefficients c_1 and c_2 were both set to 1.2. These parameter settings allow particles to gradually decrease the magnitude of velocity and position updates, at the same time ensuring that they follow convergent trajectories [87]. Further parameters were as given in table 5.1.

6.4 Results and Discussion

Tables 6.1 and 6.2 quantify the performance of the tested niching techniques. In both tables, entries marked with a '*' indicate that experiments on the relevant test problem and algorithm combination were not carried out. Marked entries apply specifically to the situation where the *nbest* algorithm was used to find multiple solutions to problems with optima of varying fitness. If only fitness is considered on problems such as function F2 and F4 (see figures 5.7 and 5.9), topological neighborhoods of particles overlap. Particles

are therefore drawn to solutions with better fitness, and only converge on solutions with optimal fitness.

6.4.1 Computational Cost

Table 6.1 compares the computational cost of each of the niching techniques in terms of the number of fitness function evaluations required to converge. Note that in values reported in the format $a \pm b$, a refers to a mean calculated over all simulations, and b refers to the standard deviation calculated over the same values. The given results represent the actual number of fitness function evaluations that took place – distance calculations at each iteration of the respective algorithms were not factored in. This fact brings an interesting point forward: DC does not compare each individual in the population to every other population member at each generation of the algorithm: Offspring are only compared to their parents. The net effect of this is that DC requires a consistently higher number of fitness function evaluations. The following comments can be made based on the results shown in table 6.1:

- SN required fewer evaluations on the systems of equations in problems S3, S4 and S5.
- Although SN generally required a low number of fitness function evaluations, it should be taken into consideration that the basis of the SN algorithm necessitates the calculation of a derated fitness at each generation, that becomes more complex as the number of generations increases.
- DC consistently required more fitness function evaluations than the other algorithms.
- When comparing *nbest* and NichePSO, it is clear that NichePSO required a substantially smaller number of fitness function evaluations. In addition to its quota of fitness function evaluations, *nbest* also required the calculation of a matrix representing inter-particle distances, at each iteration of the algorithm.
- Both *nbest* and NichePSO consistently require less than the average number of fitness function evaluations to converge.

Problem	**SN**	**DC**	*nbest*	**NichePSO**	**Average**
S3	1840.73 ± 86.43	15087.67 ± 4197.35	8493.00 ± 413.27	2554.47 ± 228.22	6993.97
S4	1193.47 ± 96.30	17554.33 ± 4656.96	6934.12 ± 542.33	3965.50 ± 353.26	7411.86
S5	1926.60 ± 95.21	13816.00 ± 4224.50	7018.87 ± 670.09	2704.20 ± 134.76	6366.42
F1	4102.07 ± 576.58	14647.33 ± 4612.15	4769.00 ± 44.90	2371.86 ± 109.41	6472.57
F2	3504.80 ± 463.20	13052.33 ± 2506.65	*	2934.00 ± 475.44	6497.04
F3	4140.97 ± 553.93	13930.00 ± 3284.38	4789.00 ± 51.35	2403.73 ± 194.94	6315.93
F4	3464.07 ± 286.97	13929.33 ± 2996.15	*	2820.03 ± 517.09	6737.81
F5	3423.33 ± 402.13	14295.67 ± 3407.82	5007.67 ± 562.14	2151.47 ± 200.42	6219.54
Average	2949.51	14539.10	6168.61	2738.16	

Table 6.1: Average number of fitness function evaluations required to converge for each niching algorithm. Entries marked with a '*' indicate that experiments were not carried out for the relevant problem and algorithm.

- Overall, NichePSO required the least number of fitness function evaluations to converge.

6.4.2 Performance Consistency

Table 6.2 expresses as percentages the performance consistency of the tested niching techniques. 'Performance consistency' reflects each of the algorithm's ability to consistently locate all solutions to each of the optimization problems. All the tested techniques sufficiently maintained solutions sets. The following conclusions can be drawn from the results reported in table 6.2:

- Results for the SN algorithm compares well to that reported in [4]. The algorithm did however not perform as well on systems of equations, and generally performed worse than the average.
- Regardless of its higher computational requirement, DC did not generally yield superior performance.
- Although still better than SN, NichePSO performed equal to or worse than the average performance on the systems of equations in problems S3, S4 and S5.
- The *nbest* algorithm appears to have exhibited the most consistent performance over all the test problems (keeping in mind that it was not applied to F2 and F4).

6.5 Conclusion

This chapter presented an empirical comparison between the performances of the newly introduced PSO niching algorithms, and two well-known GA based techniques. Section 6.2 gave a brief overview of the algorithms. The test functions presented in section 6.3.1 were all well-known functions that have been used in this thesis and other literature for evaluating the effectiveness of niching techniques. Section 6.4 analyzed the experimental results obtained. It was established that both *nbest* and NichePSO are successful niching algorithms, and their performance, both in terms of computational complexity and performance consistency compare well to existing niching techniques.

Problem	**Sequential Niching**	**Deterministic Crowding**	*nbest*	**NichePSO**	**Average**
S3	76%	93%	100%	87%	89.00%
S4	66%	100%	100%	80%	86.50%
S5	83%	87%	100%	90%	90.00%
F1	100%	100%	93%	100%	98.25%
F2	83%	93%	*	93%	89.67%
F3	100%	90%	93%	100%	95.75%
F4	93%	90%	*	93%	92.00%
F5	86%	90%	100%	100%	94.00%
Average	85.88%	92.86%	97.67%	92.88%	

Table 6.2: The consistency with which each of the techniques managed to locate a complete set of solutions for each of the test problems. Entries marked with a '*' indicate that experiments were not carried out for the relevant problem and algorithm.