A Particle Swarm Model for Tracking Multiple Peaks in a Dynamic Environment using Speciation

Daniel Parrott and Xiaodong Li
School of Computer Science and Information Technology,
RMIT University, Melbourne, Vic 3001, Australia
Email: {dparrot, xiaodong}@cs.rmit.edu.au

***Abstract*— A particle swarm optimisation model for tracking multiple peaks in a continuously varying dynamic environment is described. To achieve this, a form of speciation allowing development of parallel subpopulations is used. The model employs a mechanism to encourage simulataneous tracking of multiple peaks by preventing overcrowding at peaks. Possible metrics for evaluating the performance of algorithms in dynamic, multimodal environments are put forward. Results are appraised in terms of the proposed metrics, showing that the technique is capable of tracking multiple peaks and that its performance is enhanced by preventing overcrowding. Directions for further research suggested by these results are put forward.**

I. Introduction

The particle swarm model is a tool used for the optimisation of continuous, non-linear problems [1]. Optimisation is achieved by 'flying' particles through a solution space representing a problem with the particle's position representing a possible solution. Particles evaluate the fitness of a solution represented by their coordinates and record the position of the best solution they have found so far (their personal best or *pbest* value). Particles communicate with their neighbours to record the best position found by other particles (the global best or *gbest* value). Using the knowledge of their own best position and others' best position, particles derive a velocity vector which is used to update their position. By using a swarm of particles behaving in this fashion, a solution space can be searched for a global optimum. The equations describing this behaviour are as follows:

$$v_i(t) = wv_i(t-1)+c_1r_1(p_i-x_i(t-1))+c_2r_2(p_g-x_i(t-1)) \quad (1)$$

$$x_i(t) = x_i(t-1) + v_i(t) \ , \quad (2)$$

where $v_i(t)$ represents the velocity of particle i at time t, $x_i(t)$ its position, p_i and p_g the previous best position of the particle (*pbest*) and its neighbours (*gbest*) respectively, c_1 and c_2 are two positive constants, w the inertia weighting and r_1 and r_2 two random numbers in the range [0,1].

Particle Swarm Optimisation (PSO) variations have been developed to search multimodal environments [2] and to track a single peak in a dynamic environment [3]. However, to the best knowledge of the authors a PSO model for tracking multiple peaks in a dynamic environment has not been developed. In this respect PSO development lags behind that of genetic algorithms, variants of which have been developed to operate in dynamic multimodal envirnoments (e.g. [4]).

Dynamic multimodal environments may change in several ways - peaks may shift spatially, change shape and change height. Note that a dynamic multimodal environment in which only fitness changes is roughly equivalent to a static multimodal environment - once the peaks' (static) locations are found no further searching is needed. On the other hand, if only position changes and not height, only the single peak representing the global optimum needs to be tracked. In a fully dynamic multimodal environment, the peak representing the global optimum may decrease while a local optimum increases, changing not just the position of the global optimum as the peaks shift but also the peak which must be tracked to find it. A peak may 'disappear' as it is obscured by a higher peak above it, or peaks may appear or disappear entirely. To effectively search such a space an evolutionary computation technique should track multiple peaks, rather than a single peak temporarily representing the global optimum, in order to maintain the location of the global optimum. A two-dimensional, three-peak mulitmodal space as used later in this paper is shown in Figure 1.

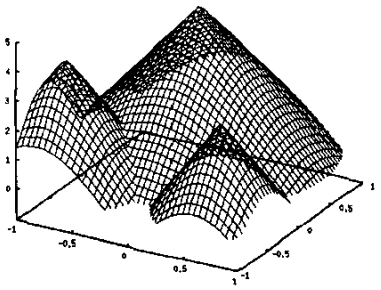

Fig. 1. Three-peak multimodal environment

This paper provides a technique for multimodal optimisation in a dynamic environment using a form of speciation similar to that developed in work by Li et al. on a genetic algorithm for multimodal optimisation [5]. This technique uses a local 'species seed' which provides the local best value to particles whose position in the solution space is within a user-specified radius of the seed. This encourages particles to converge upon local optima rather than all converging to a single

0-7803-8515-2/04/$20.00 ©2004 IEEE

global optimum, hence developing multiple sub-populations in parallel.

In the implementation discussed, Morrison and De Jong's dynamic function generator DF1 [6] is used to continuously vary the shape, height and position of peaks within a solution space. Particles update their fitness at their current position and their own recorded best-so-far position in order to remain up-to-date in the dynamic environment.

A parameter p_{max} is used to limit the number of particles in a sub-population (i.e. the number of particles sharing a common *gbest*). Those particles located within the sub-population's space (whose distance to the *gbest* position is less than the species radius) beyond the number of allowable particles are reinitialised at random locations about the search space.

Research relevant to the problem is explored in Section 2. Section 3 describes the technique itself. Results are given in Section 4, along with the problem generator on which the technique was tested and the metrics used to evaluate its performance. A discussion of these results is given in Section 5. Following that are the conclusions drawn from this research and suggestions of directions for further research.

II. Related Research

While research on optimisation in dynamic multimodal environments has been carried out using GA, none has yet been published on the use of particle swarms in such an environment. Evolutionary computation research relevant to dynamic environments, multimodal environments and dynamic multimodal environments is briefly reviewed here.

A. Genetic Algorithm for Dynamic Multimodal Environments

A genetic algorithm designed to find optima in a dynamic, multimodal enviroment is described by Ursem (the Multinational GA, [4]). Multinational GAs use multiple GA populations or nations to track multiple peaks in a dynamic environment, with each nation having a policy representing the best point of the nation. A hill-valley detection algorithm is used to sample points on a line drawn between policies and its results used to migrate individuals from one nation to another, to merge nations and to establish new nations on newly found peaks. It should be noted that the hill-valley detection algorithm works only on points between policies (known optima) - the remainder of the space remains unsampled unless by mutation. The Multinational GA was tested using several methods of moving a pair of peaks in two-dimensional environments.

The concepts of using multiple populations to track peaks and of migrating indiviudals from one population to another, combining populations and using existing populations to seed new populations could be applicable to PSO.

B. PSO Algorithms for Dynamic Environments

PSO algorithms appear to be well suited to dynamic environments. Eberhart and Shi investigated using PSO to track a single peak varying spatially only [3]. However, they noted that in a dynamic environment the height and position may change in a number of environments simultaneously (but omitted that the shape and number of peaks may also change). The authors considered that ability to adapt to a periodic change occurring every hundred generations should be sufficient. Using a standard particle swarm to track a single peak in three dimensions (parabolic function $f(x) = x^2 + y^2 + z^2$), they obtained errors several orders of magnitude less than those of comparable GA-based approaches.

To adapt PSO to dynamic environments, Hu and Eberhart [7] suggested monitoring environments for a change and updating the *gbest* and *pbest* values of particles when a change is detected. Again, they have only tested their algorithms on single-optimum environments. Carlisle and Dozier investigated a similar mechanism in their work [8] and also suggested periodic resetting of personal and population fitness values.

C. PSO and GA Algorithms for Multimodal Environments

Various methods of niching, fitness sharing and speciation have been used in evolutionary algorithms to find optima in multimodal environments.

Brits et al. [9] adapted the unimodal particle swarm optimiser using niching to find multiple optima in parallel in a static multimodal environment, a PSO variant they refer to as NichePSO. Particles are initialised uniformly throughout the search space using Faure sequences (the authors stated that success of the algorithm depends on the proper initial distribution of particles). Particles in the main swarm do not share knowledge about the best solution - they use only their own knowledge ('cognition only'). When a particle's fitness shows little change over several iterations a subswarm is created with it and its closest topological neighbour as members. Particles entering the subswarm's space (a sphere centred on the position of the best particle in the subswarm with radius defined as the distance between the centre and the particle in the subswarm furthest from the centre) automatically become part of the subswarm. The algorithm was reported to be successful at detecting global maxima and sometimes local maxima (although this point was not emphasized in the paper).

While providing useful ideas for a dynamic-environment PSO, the gradual absorption of all particles, inability to break populations and use of a convergence-biased PSO in subswarms do not allow it to be directly applied. Requiring a certain distribution of particles for the method to succeed is clearly not useful for a dynamic environment as particles in a dynamic environment will congregate around the peaks which must then be tracked.

Parsopoulas et al. studied altering the fitness value via fitness function stretching [10] to adapt PSO to sequentially find peaks in a multimodal environment. However, in a dynamic environment it is required to develop multiple populations in parallel. Kennedy [11] has investigated modifying the PSO algorithm with stereotyping - clustering based on particles previous position, with cluster centers substituted for individual's or neighbour's previous bests - which by causing clusters

to focus on local regions provides an algorithm suitable for finding optima in multimodal environments in parallel.

Li et al. [5] have developed a species-based GA for use in static multimodal environments which is applicable with little modification to PSO techniques. A spatial speciation technique is used which creates clusters of individuals around species seeds, representing a local best particle not yet assigned to another species, with other particles within the species radius (that are not members of another species) forming a species. Species are conserved by copying them into the next generation if they do not survive the GA breeding process. The technique has similarities to spherical k-means clustering [12] in its use of population radius.

III. Speciation Algorithm

Tracking multiple dynamic peaks requires a technique that allows the development of multiple sub-populations in parallel. Any such technique should:

- allow an unbiased search for the local optimum by members of the exploiting sub-population;
- encourage particles to find multiple peaks;
- provide a natural method for individuals to join sub-populations, for sub-populations to join and split, and for sub-population formation; and
- prevent too many particles focusing on a few peaks to the detriment of the total population's ability to search the solution space and track other peaks.

These requirements are features of the algorithm described which uses speciation to create multiple sub-populations in parallel, with each sub-population attempting to track and 'exploit' a local peak. These sub-populations or species are centred on the best known position of the fittest particle in a local region defined as a sphere of radius r (the speciation radius) centred on the best position of the fittest particle or 'species seed'. All particles belonging to the species adopt the *pbest* position of the species seed as their *gbest* position. Hence, a candidate species member is defined as any particle x such that the distance d between it and the species seed s is less than the speciation radius r:

$$d(x, s) \leq r \ , \tag{3}$$

where the distance $d(x, s)$ is defined as the Euclidean distance between two points in n dimesnions:

$$d(x, s) = \sqrt{\sum_{i=1}^{n} (x_i - s_i)^2} \ . \tag{4}$$

Where a particle is a candidate member of two species, it will be allocated to the species with the fitter species seed (Figure 2).

Using this mechanism, every particle is either a species seed (possibly for a species with only itself as a member) or a member of a species. Species themselves will be reformed each iteration of the algorithm, frequently with a different species seed and set of members than that of any species of the previous iteration (although the same set of particles will likely remain near each other for periods of the time as they track the same peaks). In this way the requirements that the technique should allow unbiased search for local peaks, encourage the finding of multiple peaks and provide a natural way for sub-populations to form and alter are met.

However, a mechanism for preventing too many particles attempting to track a single peak is still needed. In a dynamic environment, it is necessary to track not just the current global optimum but also local optima which are potentially the global optimum in the near future. To accomplish this a maximum species population parameter p_{max} has been introduced such that only the best p_{max} candidate members (including the species seed) will be allocated as members of the species. The lower fitness candidate members which would cause the species population to exceed p_{max} are reinitialised at random positions in the solution space. In this way, the total population can be prevented from focusing its attention on too few areas and encouraged to explore the total solution space.

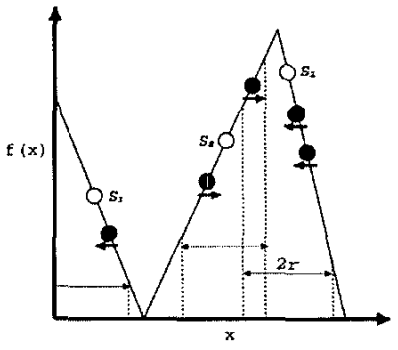

Fig. 2. Representation of speciation. Note the candidate member of both species 1 and 2 moving towards species seed S_1, the fitter seed.

To allow the algorithm to operate in a dynamic environment, each particle's *pbest* fitness value is re-evaluated at its recorded *pbest* positon each iteration. As the algorithm was designed to operate in a continuously varying environment, this extra evaluation is warranted although it doubles the number of fitness evaluations performed by the algorithm.

IV. Method

A. Test Function

Morrison and De Jong's DF1 dynamic test function generator [6] has been used to generate the environment. This function is capable of generating a given number of peaks in a given number of dimensions that vary both spatially (position and shape of the peak) and in terms of fitness. Fitnesses cycle between a maximum and minimum value creating a saw-tooth profile when fitness is graphed against iterations for a peak. The rate at which environments alter is set by a parameter A used as input to a logistics function:

$$Y_i = A \cdot Y_{(i-1)} \cdot (1 - Y_{(i-1)}) \ . \tag{5}$$

The function can produce a series of stepsizes varying from 0 to constant values through to a chaotic series of numbers. A value of A less than 1.0 produces a static environement while increasing A beyond 1.0 produces increasingly dynamic environments with pseudo-random movement.

B. Measurement

Measuring the effectiveness of an evolutionary algorithm in a dynamic environment is substantially more difficult than in a static environment. Traditional measures used in static environments, such as mean fitness, best current fitness and time to convergence lose their relevance when full convergence is not desired and the best possible fitness is continuously changing. As a result, the average minimum error over a run was used, where minimum error each iteration was defined as:

$$error = 1 - \frac{current\ best\ fitness}{current\ global\ maximum\ possible\ fitness} \quad (6)$$

The minimum and maximum errors for the run were recorded and the standard deviation calculated to measure the variability of the error. To reduce the variability of results, each set of parameters was run 50 times and the averages reported.

C. Environment

The experiments were run in a 2-dimensional environment with range [-1.0, 1.0] for 500 iterations each with a default dynamism set by A=1.2 . The PSO parameters c_1 and c_2 were set to 1.4 and inertia weight to 0.7. The default number of agents and maximum species population p_{max} were both set to 60 (hence operating as if the maximum species population were not a factor), the default speciation radius r set to 0.1 and the environment created with three peaks by default.

Fifty runs were performed for each experiment and the results averaged over the runs. Parameters were kept at the default settings other than those explicitly varied.

V. Results

A. Default parameters with Logistics Function parameter A varying

The model was run with values of A of 0.0, 1.1, 1.2, 1.5 and 2.0 to vary the dynamism of the environment, with results given in Table I. As expected, as A was increased the average and maximum errors over the run also increased. Standard deviation for the dynamic cases was about 0.03 to 0.04 greater than the average error. These results indicate that the algorithm with default settings finds it increasingly difficult to track the global optimum as the level of dynamism increases.

B. Default with parameters with number of agents varying

As the number of agents was increased from 30 to 150 in steps of 30 the average error decreased from 0.1 to 0.04 in an almost linear fashion (the change in average error decreased slightly with each increase in agent numbers, suggesting that the 'law of diminishing returns' was in effect). As shown in Table II, standard deviation also decreased from 0.13 to 0.07 while maximum error decreased from 0.45 to 0.30 over the same range of agent numbers. These results show that increasing the number of agents decreased the average error as would be expected.

C. Default with number of peaks varying, agent/peak ratio constant

The model was run with a constant ratio of 20 agents per peak and 1, 3 and 10 peaks. Average error decreased from 0.08 for 1 peak to 0.07 for 3 peaks and 0.03 for 10 peaks. as shown in Table III. This decrease in error is attributed to the increased agent density.

D. Varying species p_{max} and speciation radius r

The algorithm was run with p_{max} set to values of 2, 5, 10, 20, 40 and 60 while the speciation radius r was set to 0.1, 0.2, 0.5, and 1.0 and the number of peaks held constant at three. Results are shown in Figure 3.

Notably each of the r settings of 0.2, 0.5 and 1.0 performed comparatively well at a specific p_{max} setting (10, 20 and 40 respectively) with an average error between 0.039 and 0.042. Figure 3 suggests there may be a superior p_{max} setting between 20 and 40 agents for r = 1.0. The default setting of r=0.1 with p_{max} 20 is relatively bad and r=0.1 appears to perform uniformly poorly in the experiment.

Figures 4 and 5 shows the difference in clustering between the $\{r$=0.2, p_{max}=10$\}$ instance and the $\{r$=1.0, p_{max}=40$\}$ instance. The low species population case shows less clustering of agents around the peaks and a more evenly distributed population than the high species population case. Note that in Figure 5 there is a peak without attending agents (at approximate (x, y) coordinates (-0.5, 0.8)); as suggested by the contour lines, this 'peak' is submerged beneath its neighbour and hence undetectable by the agents.

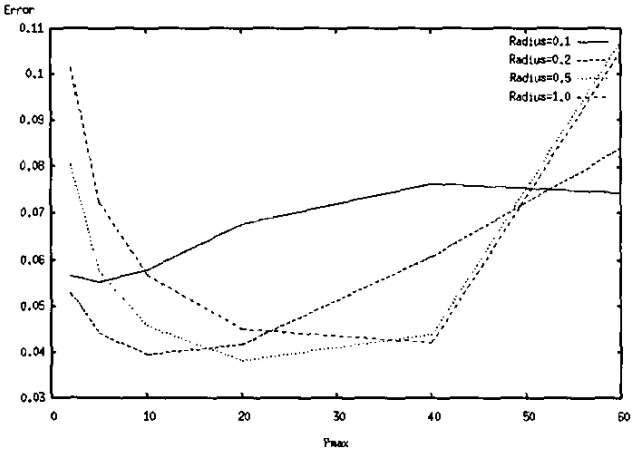

Fig. 3. Error versus p_{max} with varying population radius

E. Varying species p_{max} and A

With the r set to 0.1, p_{max} was assigned values of 2, 5, 10, 20, 40 and 60 while the parameter A was set to 1.1, 1.2,

TABLE I

RESULT OF INCREASING DYNAMISM BY VARYING A

A	Average error	SD of average error	Minimum error	Maximum error
0	0	0.01	0	0.16
1.1	0.06	0.09	0	0.39
1.2	0.08	0.11	0	0.45
1.5	0.1	0.13	0	0.49
2	0.12	0.16	0	0.53

TABLE II

RESULT OF INCREASING NUMBER OF AGENTS

No. of Agents	Average error	SD of average error	Minimum error	Maximum error
30	0.1	0.13	0	0.45
60	0.07	0.11	0	0.4
90	0.06	0.09	0	0.39
120	0.05	0.08	0	0.35
150	0.04	0.07	0	0.3

TABLE III

RESULT OF INCREASING NUMBER OF AGENTS

No. of Peaks	No. of Agents	Average error	SD of average error	Minimum error	Maximum error
1	20	0.08	0.12	0	0.81
3	60	0.07	0.1	0	0.42
10	200	0.03	0.04	0	0.18

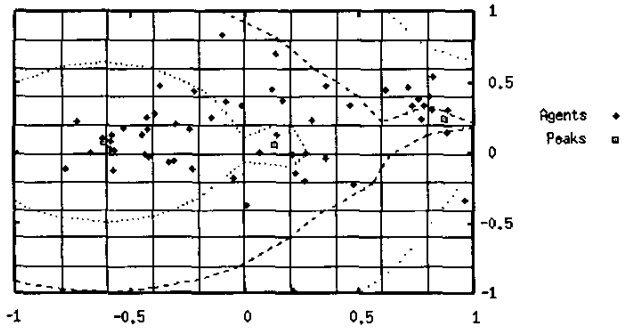

Fig. 4. Swarm with $\{r$=0.2, p_{max}=10$\}$

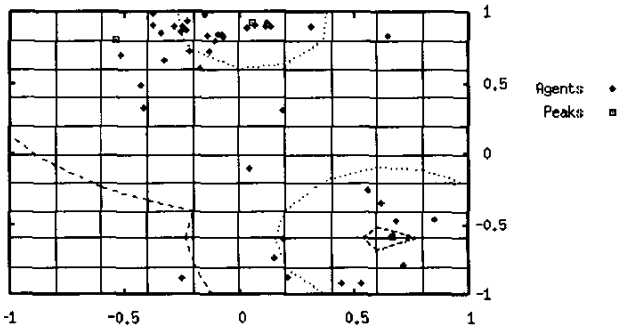

Fig. 5. Swarm with $\{r$=1.0, p_{max}=40$\}$

1.5, and 2.0 and the number of peaks held constant at three. Results are shown in Figure 6.

As expected from earlier results, the average error increases as A increases. In the A = 1.1 and A=1.2 cases there is a minimum error at p_{max}=10 and p_{max}=5 respectively while for A=1.5 and A=2.0 error is at a minimum with the lowest p_{max} value, indicating that the level of dynamism was too great for the algorithm to effectively track the peaks at these settings.

VI. DISCUSSION

The speciation technique used allows the particle swarm model to succesfully track multiple peaks in a dynamic, multimodal environment. The results showing that average error decreases as population increases and that average error increases as dynamism increases are to be expected. Simply having more particles to find peaks should decrease error while changing peaks faster will make them more difficult to track and increase the error.

Increasing the number of peaks while keeping the ratio of agents to peaks constant decreases the average error. This is believed to be the result of simply having more agents available to track the peak. With a greater number of agents in the same area, any peak becoming the global optimum is more likely to have an agent nearby to exploit it and having more agents available to track a peak will decrease error.

The use of a p_{max} parameter has a beneficial effect on the algorithm when compared to the results obtained with sub-population size limited only by the population size. Roughly equal results were obtained at three different settings of r and p_{max}: $\{r$=0.2, p_{max}=10$\}$, $\{r$=0.5, p_{max}=20$\}$ and $\{r$=1.0, p_{max}=40$\}$. Why this is so is not clear; possibly there is an

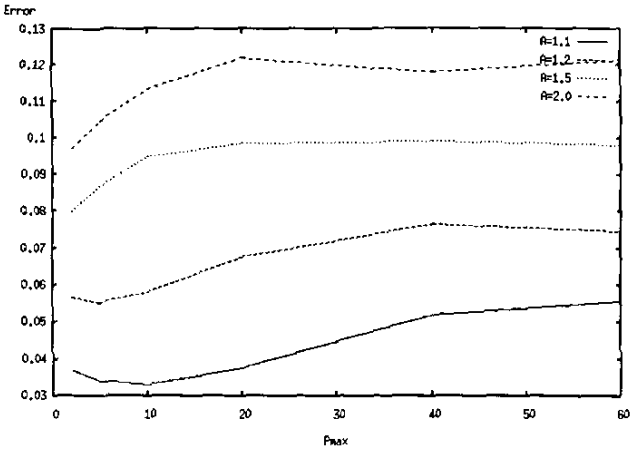

Fig. 6. Error versus p_{max} with varying dynamism

appropriate population density around the peak which balances exploitation of the peak with exploration of the rest of the solution space. It is also possible that different mechanisms are contributing to a similar result - lower p_{max} values should result in a higher number of particles being redistributed about the search space hence favouring exploration while higher p_{max} values will lead to larger species and better tracking of peaks. That average error is lower in high dynamism environments when p_{max} values are low suggests that the resulting redistribution of particles throughout the solution space is the cause of the improved result. This in turn suggests that for continually varying highly dynamic environments, random search may outperform the PSO model described. Logically, this is to be expected: at a sufficiently high level of dynamism there will not be a relationship between a peak's position one iteration and its position the next so that an algorithm aimed at tracking the peak will have no benefit compared to a random search.

VII. Conclusion and Future Works

The results demonstrate that the speciation and crowding mechanisms are able to track multiple continually altering peaks. In highly dynamic environments low species populations lead to lower error. For a given problem and speciation radius, there appears to be an optimal maximum allowable population for a species if attempting to achieve the lowest average error. However, there may be multiple speciation radius/population combinations which give similar lowest average errors.

The relationship between species population size and radius is one among many areas for potential future research. The standard PSO parameters c_1, c_2 and inertia weight were kept constant throughout the experiments; highly dynamic environments may favour greater values for these parameters giving rise to faster-moving particles able to track a fast-moving peak better. More investigations of the model's ability to track large numbers of peaks are needed. An investigation of the algorithm's success in environments with peaks shifting periodically (every n iterations rather than every iteration) and in higher dimension environments is also needed.

To effectively measure tracking in higher dimension environments will require improved measurement techniques. Understanding of the algorithms behaviour in two dimensions was partially achieved by watching a live representation of the particles and peaks. This representation itself could be improved to give an indication of a peak's height and width, whether a peak is submerged and which particles are acting as seeds. This information cannot be easily represented graphically in, say, a ten dimension environment. Hence, a measure of clustering needs to be developed; an average over a run of the sum of distances between each particle and its closest peak may be useful.

This paper has given details of a particle swarm model for tracking multiple peaks in a dynamic environment using speciation and demonstrated that it works. Although the paper does not fully explore the possibilities of the algorithm, it is hoped that it may contribute to others' efforts to further research into evolutionary optimisation of dynamic, multimodal environments, a topic at the cutting edge of evolutionary computation.

References

[1] Kennedy, J., and Eberhart, R. C.: Particle swarm optimisation. In *Proceedings of the IEEE International Conference on Neural Networks*. Piscataway, NJ. IEEE Service Center (1995) 1942-1948

[2] Kennedy, J. and Spears, W. M.: Matching algorithms to problems: An experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. In *Proceedings of the IEEE International Conference on Evolutionary Computation*. IEEE Service Center (1998) 78- 83

[3] Eberhart, R.C. and Shi, Y.: Tracking and optimizing dynamic systems with particle swarms. In *Proceedings of the 2001 Congress on Evolutionary Computation, CEC2001*. IEEE Press (2001) 94 -100.

[4] Ursem, R., K.: Multinational GAs: Multimodal Optimization Techniques in Dynamic Environments. In *Proceedings of the Second Genetic and Evolutionary Computation Conference, GECCO-2000*. Morgan Kaufmann Publishers (2000) 19-26

[5] Li, J., Balazs, M.E., Parks, G.T., Clarkson, P.J.: A Species Conserving Genetic Algorithm for Multimodal Function Optimization. *Evolutionary Computation* 10(3):207 - 234, (2002)

[6] De Jong, K.A., Morrison, R.W.: A Test Problem Generator for Non-Stationary Environments. In *Proceedings of the Congress on Evolutionary Computation*. IEEE Press (1999) 2047-2053

[7] Hu, X. and Eberhart, R. C.: Adaptive particle swarm optimization: detection and response to dynamic systems. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC2002*. IEEE Press (2002) 1666-1670

[8] Carlisle, A., Dozier, G.: Adapting Particle Swarm Optimization to Dynamic Environments. In *Proceedings of the International Conference on Artificial Intelligence* Las Vegas, Nevada, USA. (2000) 429-434

[9] Brits, R., Engelbrecht, A. P., van den Bergh, F.: A Niching Particle Swarm Optimizer. In *4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL2002)*, (2002) 692-696

[10] Parsopoulos,K.E., Plagianakos,V.P., Magoulas,G.D., Vrahatis, M.N.: Stretching technique for obtaining global minimizers through Particle Swarm Optimization. In *Proceedings of the Particle Swarm Optimization Workshop*, Indianapolis, USA. (2001) 22 - 29

[11] Kennedy, J.: Stereotyping: Improving Particle Swarm Performance with Cluster Analysis. In *Proceedings of the 2000 Congress on Evolutionary Computation, CEC2000*. IEEE Press (2000) 1507-1512

[12] Fasulo, D.: An Analysis of Recent Work on Clustering Algorithms. Technical report UW-CSE01 -03-02, University of Washington, 1999