A NICHING PARTICLE SWARM OPTIMIZER

R.Brits, A.P. Engelbrecht, F. van den Bergh

Department of Computer Science, University of Pretoria, Pretoria, South Africa
riaan@raptorinternational.com, engel@driesie.cs.up.ac.za,fvdbergh@cs.up.ac.za

ABSTRACT

This paper describes a technique that extends the unimodal particle swarm optimizer to efficiently locate multiple optimal solutions in multimodal problems. Multiple subswarms are grown from an initial particle swarm by monitoring the fitness of individual particles. Experimental results show that the proposed algorithm can successfully locate all maxima on a small set of test functions during all simulation runs.

1. INTRODUCTION

This paper presents the *niching particle swarm optimization* algorithm, NichePSO. NichePSO is aimed at locating multiple, optimal solutions in a multimodal problem. Particle swarm optimizers (PSO), using the *lbest*, and *gbest* algorithms, have been shown to effectively solve unimodal optimization problems [16]. These PSO algorithms are however not well equipped to locating multiple optimal solutions, because of their social propagation of information regarding a global good solution [2]. Maintaining diversity to avoid convergence to local optima and locating a single global solution, has received much research attention (see [7], [18], [11], [15]).

Niching algorithms, up to now analyzed mostly under the wing of genetic algorithms (GAs), have as their goal the identification and maintenance of multiple optimal solutions [12, 6]. A real world application of niching, applied to the inversion of teleseismic waves, can be found in [10]. This paper introduces a novel particle swarm optimization algorithm to detect multiple optima in parallel. To our knowledge, this is the first PSO niching technique that searches for multiple solutions in parallel.

In section 2, we present an overview of the PSO algorithm. Section 3 gives a brief overview of existing niching techniques, both in the fields of genetic algorithms, and particle swarm optimizers. The new NichePSO algorithm is presented and discussed in section 4, with experimental results discussed in section 5.

2. PARTICLE SWARM OPTIMIZERS

Particle swarm optimizers are optimization algorithms, modeled after the social behavior of flocks of birds [9]. PSO is a population based search process where individuals, referred to as particles, are grouped into a swarm. Each particle in the swarm represents a candidate solution to the optimization problem. In a PSO system, each particle is "flown" through the multidimensional search space, adjusting its position in search space according to its own experience and that of neighboring particles. A particle therefore makes use of the best position encountered by itself and that of its neighbors to position itself toward an optimal solution. The effect is that particles "fly" toward a minimum, while still searching a wide area around the best solution. The performance of each particle (i.e. the "closeness" of a particle to the global optimum) is measured using a predefined fitness function which encapsulates the characteristics of the optimization problem.

Each particle i maintains the following information: (1) $\mathbf{x}_i$, the *current position* of the particle, (2) $\mathbf{v}_i$, the *current velocity* of the particle, and (3) $\mathbf{y}_i$, the *personal best position* of the particle. The personal best position associated with a particle i is the best position that the particle has visited thus far, i.e. a position that yielded the highest fitness value for that particle. If f denotes the objective function, then the personal best of a particle at a time step t is updated as:

$$\mathbf{y}_i(t+1) = \begin{cases} \mathbf{y}_i(t) & \text{if } f(\mathbf{x}_i(t+1)) \geq f(\mathbf{y}_i(t)) \\ \mathbf{x}_i(t+1) & \text{if } f(\mathbf{x}_i(t+1)) < f(\mathbf{y}_i(t)) \end{cases} \quad (1)$$

Two main approaches to PSO exist, namely *lbest* and *gbest*, where the difference is in the neighborhood topology used to exchange experience among particles. For the *gbest* model, the best particle is determined from the entire swarm. If the position of the best particle is denoted by the vector $\hat{\mathbf{y}}$, then

$$\begin{aligned} \hat{\mathbf{y}}(t) \in \{\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_s\} \mid f(\hat{\mathbf{y}}(t)) \\ = \min\{f(\mathbf{y}_0(t)), f(\mathbf{y}_1(t)), \ldots, f(\mathbf{y}_s(t))\} \end{aligned} \quad (2)$$

where s is the total number of particles in the swarm. For the *lbest* model, a swarm is divided into overlapping neighborhoods of particles. For each neighborhood N_j, a best particle is determined with position $\hat{\mathbf{y}}_j$. This best particle is referred to as the *neighborhood best* particle, defined as

$$\begin{aligned} N_j = \{\mathbf{y}_{i-l}(t), \mathbf{y}_{i-l+1}(t), \ldots, \mathbf{y}_{i-1}(t), \mathbf{y}_i(t), \\ \mathbf{y}_{i+1}(t), \ldots, \mathbf{y}_{i+l-1}(t), \mathbf{y}_{i+l}(t)\} \end{aligned} \quad (3)$$

$$\hat{\mathbf{y}}_j(t+1) \in N_j \mid f(\hat{\mathbf{y}}_j(t+1)) = \min\{f(\mathbf{y}_i)\}, \forall \mathbf{y}_i \in N_j \quad (4)$$

Neighborhoods are usually determined using particles indices [7], although topological neighborhoods have also been used [18]. The *gbest* PSO is a special case of *lbest* with $l = s$, where l is the number of particles per neighborhood, and s is the total number of particles in the swarm; that is, the neighborhood is the entire swarm.

For each iteration of a *gbest* PSO algorithm, the jth-dimension of particle i's velocity vector, $\mathbf{v}_i$, and its position vector, $\mathbf{x}_i$, is updated as follows:

$$\begin{aligned} v_{i,j}(t+1) &= wv_{i,j}(t) + c_1 r_{1,j}(t)(y_{i,j}(t) - x_{i,j}(t)) + \\ &\quad c_2 r_{2,j}(t)(\hat{y}_j(t) - x_{i,j}(t)) \quad (5) \\ \mathbf{x}_i(t+1) &= \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \quad (6) \end{aligned}$$

where w is the inertia weight, c_1 and c_2 are the acceleration constants and $r_{1,j}(t), r_{2,j}(t) \sim U(0,1)$. The reader is referred to [20] for a study of the relationship between the inertia weight and the acceleration constants in order to select values which will ensure convergent behavior. Another PSO learning approach, known as the *cognition only* model, was tested by Kennedy [8]. This model uses only a particle's personal best position found thus far in the velocity update. $\mathbf{v}_i$ is then updated as

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_{1,j}(t)(y_{i,j}(t) - x_{i,j}(t)) \quad (7)$$

Clearly, there is no sharing of social information, and each particle effectively performs an individual search in its local area, based on its personal experience.

The PSO algorithm performs repeated applications of the update equations until a specified number of iterations has been exceeded, or until velocity updates are close to zero.

3. NICHING TECHNIQUES

Niching methods attempt to find multiple solutions to optimization problems. They have been studied extensively in the field of *genetic algorithms* (GAs) (see [12] and [6] for a complete discussion) and several different approaches exist. *Parallel* niching methods identify and maintain several niches in a population simultaneously. *Sequential* niching methods find multiple solutions by iteratively applying niching to a problem space, while marking a potential solution at each iteration to ensure that search efforts are not duplicated. Some GA niching methods are summarized below:

Sharing Goldberg [4] regards each niche as a finite resource, and shares this resource among all individuals in the niche. An individual's fitness, f_i, is adapted to its shared fitness $f_i' = f_i / \sum_j sh(d_{i,j})$. The sharing function is defined as $sh(d) = 1 - (d/\sigma_{share})^\alpha$ if the distance d between individuals i and j is less than σ_{share}, otherwise its zero. The distance measure d can be genotypic or phenotypic, depending on the problem to be solved.

Sequential niching is a simple, fast algorithm [1] that identifies multiple solutions by adapting the objective function's fitness landscape through the application of a derating function at a position where a potential solution was found. After a possible solution is identified, the algorithm is restarted to search for other solutions. Repeating this process a sufficient number of times will locate all the global minima. A derating function has the simple effect of lowering the fitness appeal of previously located solutions.

Deterministic Crowding (DC), based on work done in [3] by De Jong, evolves a population by deriving offspring from parents and then letting them compete against each other for a position in a next generation. (The interested reader is referred to [12] for a complete analysis.) The repeated application of this algorithm leads to the development of similarity traits, identified as niches.

Restricted Tournament Selection [5] is similar to DC, but randomly adapts selected individuals and then lets them compete against the most similar individuals from the population. These competitors may not necessarily be the individual's parent.

More recently, particle swarm techniques have been developed to locate multiple optimal solutions. These techniques include:

Objective function "stretching" [15] was applied as a sequential PSO niching technique, similar to that of Beasley *et al* [1]. Once the PSO algorithm has identified a local maximum $f(\mathbf{x}^*)$, (through comparing particle fitnesses to a minimum threshold value) the objective function is *stretched*, such that for each point $\mathbf{x}$, where $f(\mathbf{x}) > f(\mathbf{x}^*)$, $\mathbf{x}$ is unaffected. All other points, such that $f(\mathbf{x}) \leq f(\mathbf{x}^*)$ holds, are stretched so that $\mathbf{x}^*$ becomes a local minimum. All particles are then repositioned randomly. Van den Bergh points out that the stretching technique may alter the search space by introducing false maxima, keeping the PSO from discovering all possible solutions [20].

The ***nbest*** PSO introduced in [2] uses local, topological neighborhoods that shrink over time to locate multiple solutions to systems of equations. The *nbest* algorithm also redefines the fitness function to be more suited to systems of equations. Topological neighborhoods are similar to neighborhoods defined in the *lbest* algorithm (see section 2), but without global convergence characteristics.

4. NICHING PARTICLE SWARM OPTIMIZER

The *niching particle swarm optimization* (NichePSO) algorithm, that successfully locates multiple solutions to function optimization problems, is presented in this section. Optimization in NichePSO subswarms utilizes the Guaranteed Convergence Particle Swarm Optimization (GCPSO) algorithm [20, 21]. An overview of GCPSO is first given, and then the NichePSO is presented.

4.1. The Guaranteed Convergence Particle Swarm Optimizer

The *gbest* algorithm exhibits an unwanted property: when $\mathbf{x}_i = \mathbf{y}_i = \hat{\mathbf{y}}$ (for any particle i), the velocity update in equation (2) depends solely on the $w\mathbf{v}_i(t)$ term. When a particle approaches the global best solution, its velocity property approaches zero, which implies that eventually all particles will stop moving. This behavior does not guarantee convergence to a global best solution, or even a local best, only to a best position found thus far. This particle behavior occurred frequently when running small subswarms. (The smallest subswarms possible in our implementation, consists of two particles.) Van den Bergh introduced a new algorithm, called the GCPSO [20], to proactively counteract this behavior in a particle swarm. Let τ be the index of the global best particle. The velocity and position updates for $\mathbf{x}_\tau$ are the redefined to be

$$\begin{aligned} v_{\tau,j}(t+1) &= -x_{\tau,j}(t) + \hat{y}_j(t) + wv_{\tau,j}(t) + \\ &\quad \rho(t)(1 - 2r_{2,j}(t)) \quad (8) \\ x_{\tau,j}(t+1) &= \hat{y}_j(t) + wv_{\tau,j}(t) + \rho(t)(1 - 2r_{2,j}(t)) \quad (9) \end{aligned}$$

The term $-\mathbf{x}_\tau$ 'resets' the particle's position to the global best position $\hat{\mathbf{y}}$, $w\mathbf{v}_\tau$ signifies a search direction, and $\rho(t)(1 - 2\mathbf{r}_2(t)))$ adds a random search term to the equation. The constant $\rho(t)$ defines the area in which a better solution is searched[20].

4.2. The *NichePSO* algorithm

Parallel niching algorithms locate and track multiple solutions simultaneously, using a fitness-based criterion to detect and mark

Niching Particle Swarm Optimization Algorithm

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

Figure 1:

individual solutions. The NichePSO algorithm can be classified as a parallel niching algorithm utilizing subswarms. Løvbjerg *et al* [11] used subswarms to improve swarm diversity and avoid premature convergence. This approach can be adapted to maintain and optimize niches in the objective function space.

Figure 1 summarizes the NichePSO algorithm. A number of issues regarding this algorithm are now discussed:

Initialization The success of the NichePSO depends on the proper initial distribution of particles throughout the search space. To ensure uniform distribution, *Faure*-sequences were used to generate initial particle positions (as described in [19]). *Faure*-sequences are distributed with high uniformity within an *n*-dimensional unit cube.

Main Swarm Training The main swarm is trained using the *cognition only* model [8]. Equation (7) shows that in this model, only a conscience factor, in the form of a personal best, is considered when updating particle positions. Therefore no social information, in the form of a *global best* solution, such as in the *gbest* and *lbest* algorithms, will influence position updates. This arrangement allows each particle to perform a local search.

Identification of niches A fundamental question when searching for different niches, is how to identify them. Parsopoulos *et al* [14] use a threshold value ϵ such that when $f(\mathbf{x}_i) < \epsilon$ for particle i, the particle is removed from the swarm, and labeled as a potential global solution. The objective function's landscape is then stretched to keep other particles from exploring this area in the search space. If the isolated particle's fitness is not close to a desired level, the solution can be refined by searching the surrounding function landscape with the addition of more particles. This approach proves to be effective when considering Parsopoulos *et al*'s results. This threshold parameter ϵ is however subject to fine tuning, and locating good solutions with it depends strongly on the objective function's landscape and dimensionality. To avoid the use of this tunable parameter, we use a similar approach that monitors changes in the fitness of a particle. If a particle's fitness showed very little change over a small number of iterations of the learning algorithm, a subswarm is created with the particle and its closest topological neighbor. More formally, the variance in particle i's fitness, σ_i, is tracked over a number of iterations, e_σ, where e_σ was set to 3 in our experiments. When $\sigma_i < \delta$, a subswarm may be created with $\mathbf{x}_i$. The threshold δ is a much more intuitive parameter than ϵ. To avoid problem dependence, σ_i is normalized according to x_{min} and x_{max}. It is possible that this approach can find local minima, satisfying $\sigma_i < \delta$. If local minima are undesired, our approach can be combined with that of a minimum fitness threshold ϵ, to ensure that solutions do meet a minimum fitness criterion. The 'closest neighbor' to particle $\mathbf{x}_i$ is simply the particle $\mathbf{x}_k$ where $\mathbf{x}_k = \arg\min_{\mathbf{x}_k}\{\|\mathbf{x}_i - \mathbf{x}_k\|\}$; k is the index of any particle in the main swarm, and $k \neq i$.

Subswarm radius Subswarm S_j's radius is defined as

$$R_j = \max\{\|S_{\mathbf{x}_{j,g}} - S_{\mathbf{x}_{j,i}}\|\} \tag{10}$$

where g is the index of the global best particle in subswarm j, and $S_{\mathbf{x}_{j,i}}$ represents all other particles in S_j subject to $i \neq g$.

Absorption of particles into a subswarm Particles are absorbed into a subswarm when they move 'into' a swarm. That is, particle $\mathbf{x}_i$ will be absorbed into S_j when

$$\|\mathbf{x}_i - S_{\mathbf{x}_{j,g}}\| \leq R_j \tag{11}$$

Merging subswarms Subswarms are merged when they intersect. Swarms that intersect are likely to converge on the same solutions. When they are merged, social information about the niche is shared between the swarms, avoiding superfluous traversal of the search space. Two subswarms, S_{j1} and S_{j2}, intersect when

$$\|S_{\mathbf{x}_{j1,g}} - S_{\mathbf{x}_{j2,g}}\| < (R_{j1} + R_{j2}) \tag{12}$$

It is entirely possible that a subswarm may have a $R_j = 0$. This situation would occur when all particles have converged to an optimal solution. In this situation, (12) fails to detect swarms in the same niche. Consequently, when two swarms, S_{j1} and S_{j2} do not satisfy (12) because they have $R_{j1} = R_{j2} = 0$ they can be merged when

$$\|S_{\mathbf{x}_{j1,g}} - S_{\mathbf{x}_{j2,g}}\| < \mu \tag{13}$$

As with δ, μ can be an appreciably small number, such as 10^{-3}, to ensure that two swarms are sufficiently similar. $\|S_{\mathbf{x}_{j1,g}} - S_{\mathbf{x}_{j2,g}}\|$ is normalized to the interval $[0, 1]$. S_{j1} and S_{j2} are merged by creating a new subswarm consisting of all S_{j1} and S_{j2}'s particles.

The GCPSO Algorithm A description of this algorithm is given in sub-section 4.1. Subswarms created by NichePSO initially always consist of two particles. Utilizing the *gbest* algorithm, especially when these particles are topologically close, they may converge to suboptimal solutions. GCPSO is therefore preferred over the 'traditional' *gbest* algorithm because of its ability to avoid stagnating on suboptimal solutions.

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

Table 1: Parameter Settings

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

Table 2: Performance Results

5. EXPERIMENTAL RESULTS

This section summarizes results of the application of the NichePSO algorithm to finding maxima of multimodal functions. Our task was to find the maxima of the following functions (see figures 2 to 6):

$$F1(x) = \sin^6(5\pi x) \quad (14)$$

$$F2(x) = \left(e^{-2\log(2)\times\left(\frac{x-0.1}{0.8}\right)^2}\right) \times \sin^6(5\pi x) \quad (15)$$

$$F3(x) = \sin^6(5\pi(x^{3/4} - 0.05)) \quad (16)$$

$$F4(x) = \left(e^{-2\log(2)\times\left(\frac{x-0.08}{0.854}\right)^2}\right) \times \sin^6(5\pi(x^{3/4} - 0.05)) \quad (17)$$

$$F5(x,y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2 \quad (18)$$

These functions were also used by Beasley *et al* [1] and Spears [17] to respectively test a GA sequential niching algorithm and GA subpopulation configurations. Function $F1$ and $F3$ both have 5 maxima with a function value of 1.0. In $F1$, maxima are evenly spaced; in $F3$ maxima are unevenly spaced. In $F2$ and $F4$, local and global peaks exist at the same x-positions as in $F1$ and $F3$, but they decrease exponentially. Function $F5$ (see figure 6), the modified Himmelblau function, has 4 equal maxima with $F5(x,y) = 200$. For each of the 5 functions, 30 experiments were done with the NichePSO algorithm, with $c_1 = c_2 = 1.2$. The inertia weight w is scaled linearly from 0.7 to 0.2, over a maximum of 2000 iterations of the algorithm. These parameter settings allow the particles to gradually slow down after initially exploring with vigor; at the same time ensuring that they follow convergent trajectories [20]. Table (1) reports further parameter settings; the indicated $|S|$ is the initial number of particles in the main swarm of every experiment, before any niche subswarms are created. For functions $F1$ to $F4$, a particle consists simply of a potential x value. For function $F5$, a particle represents a (x, y) position. We evaluate our algorithm according to *accuracy* – thus how close the discovered maxima are to the actual maxima; and *success consistency* – the proportion of the experiments that converged to the optimal solution.

Parameter values for δ and μ in table 2, have been experimentally found to be effective. Table 3 reports the mean and standard deviation of fitness of all particles in all subswarms, where the fitness is the average distance of the found solutions to the true maxima. *% Converged* signifies the percentage of experiments carried out for each function that successfully located all the global maxima. NichePSO successfully located all global maxima of all the functions tested. For functions $F2$ and $F4$, with exponentially decreasing maxima (see for example figure 3), NichePSO located the global maxima, and in some of the experiments, selected local maxima. (This explains the relatively large difference in fitness between functions $F1$ and $F3$, and functions $F2$ and $F4$.)

Beasley *et al* reported slightly worse results for functions $F1$–$F4$, and only found all maxima in 76% of the experiments done for $F5$ [1].

6. CONCLUSION AND FUTURE WORK

This paper introduced a variation to the 'traditional' PSO algorithm, namely the NichePSO. The NichePSO algorithm locates multiple optimal solutions for multimodal optimization problems, through the use of subswarms and a convergent subswarm optimization algorithm, GCPSO. Experimental results showed that the new algorithm successfully located all maxima for all the simulation runs.

A number of issues still need to be resolved, including:

- The *relationship between the number of niches/solutions* and $|S|$.
- *Sensitivity* of the algorithm to changes in parameters such as σ_i and μ. (Ideally, we do not want to introduce new parameters that require intimate knowledge of a problem domain, and that could adversely affect algorithm performance.)
- Differences in algorithm performance that occur when *gbest*, instead of *GCPSO*, is used for learning in subswarms.
- Performance on *high-dimensional* problems.
- *Empirical comparison* to existing PSO niching algorithms.

7. REFERENCES

[1] D. Beasley, D.R. Bull, R.R. Martin, *A Sequential Niching Technique for Multimodal Function Optimization*, Evolutionary Computation, 1(2), p 101-125, MIT Press, 1993.

[2] R. Brits, A.P. Engelbrecht, F. van den Bergh, *Solving Systems of Unconstrained Equations using Particle Swarm Optimization* Submitted to IEEE Conference on Systems, Man, and Cybernetics, Tunisia, 2002.

[3] K.A. de Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, PhD thesis, Dept. of Computer and Communication Sciences, University of Michigan, 1975.

[4] D.E. Goldberg, J. Richardson, *Genetic Algorithm with Sharing for Multimodal Function Optimization*, In Proceedings of the Second International Conference on Genetic Algorithms, p 41-49, 1987.

[5] G. Harik, *Finding Multiple Solutions using Restricted Tournament Selection*, In L. J. Eshelman (Ed.), Proceedings of the Sixth International Conference on Genetic Algorithms, pp. 24-31, San Francisco: Morgan Kaufmann, 1995.

[6] J. Horn *The nature of niching: Genetic algorithms and the evolution of optimal, cooperative populations*, Doctoral dissertation, 95001, Urbana, University of Illinois, Illinois Genetic Algorithm Lab, 1997.

[7] J. Kennedy, *Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance*, in Proceedings of the Congress on Evolutionary Computation, pages 1931-1938, Washington DC, USA, July 1999, IEEE Service Center, Piscataway, NJ.

[8] J.Kennedy, *The particle swarm: Social adaptation of knowledge*, in Proceedings of the International Conference on Evolutionary Computation, p 303–308, 1997.

[9] J. Kennedy, R. Eberhart, *Particle Swarm Optimization*, Proc. IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ,pp. IV: 1942-1948, 1995.

[10] K.D. Koper, M.E. Wysession, D.A. Wiens, *Multimodal function optimization with a niching genetic algorithm: A seismological example*, Bulletin of the Seismological Society of America, Volume 89, pages 978-988, 1999.

[11] M. Løvbjerg, T.K. Rasmussen, T. Krink, *Hybrid Particle Swarm Optimizer with Breeding and Subpopulations* In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), San Francisco, July, 2001.

[12] S.W. Mahfoud *Niching methods for genetic algorithms* IlliGAL Rep. 95001, Urbana, University of Illinois, Illinois Genetic Algorithm Lab, 1995.

[13] B.L. Miller, M.J. Shaw, *Genetic Algorithms with Dynamic Niche Sharing for Multimodal Function Optimization* IlliGAL Rep. 95010, Urbana, University of Illinois, Illinois Genetic Algorithm Lab, December 1995.

[14] K.E. Parsopoulos, M.N. Vrahitis, *Modification of the Particle Swarm Optimizer for Locating all the Global Minima*, In V. Kurkova, N.C. Steele, R. Neruda and M. Karny, editors, Artificial Neural Networks and Genetic Algorithms, pages 324-327, Springer, 2001.

[15] K.E. Parsopoulos, V.P. Plagianakos, G.D. Magoulas, M.N.Vrahitis, *Stretching Technique for Obtaining Global Minimizers Through Particle Swarm Optimization*, In Proceedings of the Particle Swarm Optimization Workshop, p 22-29, Indianapolis, USA, 2001.

[16] Y. Shi, R.C. Eberhart, *An Empirical Study of Particle Swarm Optimization*, Proceedings of the 1999 Conference on Evolutionary Computation, pp 1945 - 1950, IEEE Service Center, Piscataway, NJ, 1999.

[17] W.M. Spears, *Simple Subpopulation schemes*, In Proceedings of the Evolutionary Programming Conference, p 296–307, 1994.

[18] P.N. Suganthan, *Particle Swarm Optimizer with Neighborhood Optimizer* in Proceedings of the Congress on Evolutionary Computation, pages 1958-1961, Washington DC, USA, July 1999, IEEE Service Center, Piscataway, NJ.

[19] E. Thiémard *Economic Generation of Low-Discrepancy Sequences with a b-ary Gray Code*, Department of Mathematics, Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland

[20] F van den Bergh, *An Analysis of Particle Swarm Optimizers*, PhD Thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

[21] F van den Bergh, AP Engelbrecht, *A New Locally Convergent Particle Swarm Optimizer*, submitted to IEEE Conference on Systems, Man and Cybernetics, Tunisia, 2002.

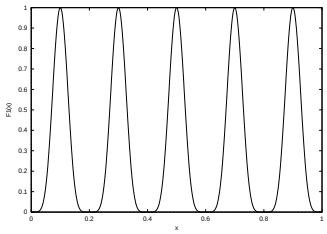

Figure 2: Function F1, evenly spaced equal maxima.

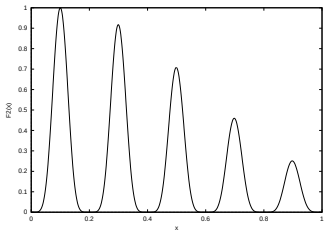

Figure 3: Function F2

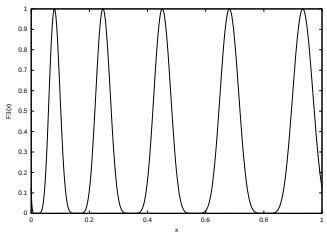

Figure 4: Function F3

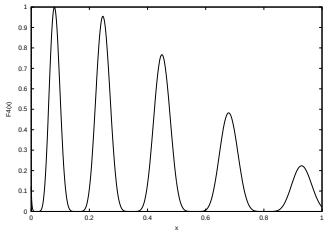

Figure 5: Function F4, Showing exponentially decreasing local maxima.

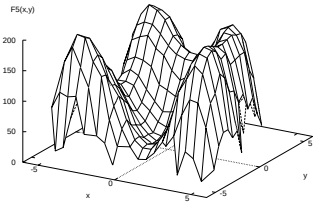

Figure 6: The 4-peak Himmelblau function.