Scalability of Niche PSO

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— **In contrast to optimization techniques intended to find a single, global solution in a problem domain, niching (speciation) techniques have the ability to locate multiple solutions in multimodal domains. Numerous niching techniques have been proposed, broadly classified as temporal (locating solutions sequentially) and parallel (multiple solutions are found concurrently) techniques. Most research efforts to date have considered niching solutions through the eyes of genetic algorithms (GAs), studying simple multimodal problems. Little attention has been given to the possibilities associated with emergent swarm intelligence techniques. Particle swarm optimization (PSO) utilizes properties of swarm behaviour not present in evolutionary algorithms such as GAs, to rapidly solve optimization problems. This paper investigates the ability of two genetic algorithm niching techniques, sequential niching and deterministic crowding, to scale to higher dimensional domains with large numbers of solutions, and compare their performance to a PSO-based niching technique, NichePSO.**

I. Introduction

Optimization is a paradigm present in nearly every aspect of life. It describes the drive to constantly find improved ways of solving old and new problems. In the context of technological development and innovation, optimization describes the search for techniques that make better use of available resources to solve problems. Scientific and engineering applications regularly require algorithms to locate and finetune optimal solutions. For example, solving systems of equations, a comparatively simple task when only a few equations and unknowns are involved, grows notably more complex as the problem's dimensionality increases [1].

Most evolutionary and swarm intelligence techniques introduced to date, are designed to locate and convergence on a single solution in a search space, where the quality of the solution depends on a problem dependent fitness function. These techniques implicitly assume that there exists only a single solution in the search space, and therefore that the search space is unimodal. When presented with a multimodal problem domain, typical unimodal techniques will either favor a single solution, or fail to convergence due the confusion introduced by multiple possible solutions. Niching techniques attempt to overcome the deficiencies of unimodal optimization techniques by explicitly assuming that multiple solutions may exist in a search space. Niching techniques are modelled after a phenomenon in nature where animal species favor different ecologies based on their individual needs, resulting in several species co-existing in a macro-ecology [2].

Niching has been investigated using both genetic algorithms (GAs) [11]-[21] and particle swarm optimization (PSO) [11], [12], [13]. Most results reported to date are based on the application of niching techniques to simple multimodal functions, usually in a single dimension with only a few solutions. This paper investigates and compares the scalability of two well-known genetic algorithm (GA) niching techniques, sequential niching (SN) and deterministic crowding (DC), to a PSO niching technique, NichePSO [3].

Section II presents a background to GAs and PSO, followed by an introduction to niching and a description of the algorithms under scrutinity in section III. Section IV presents experimental results.

II. Background

GAs use ideas from natural evolution to evolve a solution to a problem over a number of evolutionary steps, called *generations*. A *population* of *individuals*, where each individual represents a candidate solution to a problem, is advanced over a number of generations by recombining the genetic representation of individuals. In the function optimization problems presented later in this paper, each individual is represented by a fixed length bit-string, which is converted to a numerical value before being presented to a *fitness function*. A fitness function evaluates the quality of the solution presented by an individual, representable as a numerical metric, known as the individual's *fitness*. Generational operators are used to improve the fitness of individuals. These operators include:

Crossover: Parts of the genetic representation of two parent individuals are swapped to produce offspring individuals.
Mutation: Individual bits in a parent individual are flipped with a probability p_m to produce an offspring.
Selection: The selection process creates a new generation of individuals C_g, where g is a time index, from a previous generation C_{g-1}, by selecting individuals using some heuristic. One such heuristic is *elitism*, which is biased towards only selecting highly fit indivuals for a next generation.

Through the application of the above operators, a population of individuals is evolved until an acceptable solution is found to the underlying optimization problem described by the fitness function. For a more thorough treatment of the underlying GA theory, see [4].

Particle swarm optimisation is an optimization algorithm, modeled after the social behavior of flocks of birds [5]. 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:

- $\mathbf{x}_i$: The *current position* of the particle;
- $\mathbf{v}_i$: The *current velocity* of the particle;
- $\mathbf{y}_i$: 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:

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

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

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

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

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

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

Neighborhoods are usually determined using particles indices [6], although topological neighborhoods have also been used [7]. 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 j^{th}-dimension of particle i's velocity vector, $\mathbf{v}_i$, and its position vector, $\mathbf{x}_i$, is updated as follows:

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

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 [8] 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 [9]. This model uses only a particle's personal best position found thus far in the velocity update. $\mathbf{v}_i$ is then updated as

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

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.

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 occured frequently when running small subswarms. (The smallest subswarms possible in NichePSO, consists of two particles.) Van den Bergh introduced a new algorithm, called the GCPSO [8], 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

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

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 term $\rho(t)$ thus defines the area in which a new solution is searched [10].

III. Niching

A. Background

Unimodal optimization techniques assume that a single solution exists in a problem's search space. In multimodal search domains, where multiple, equally acceptable solutions exist, unimodal techniques will either find a single solution, or fail to converge due to the confusion introduced by the presence of multiple solutions. In contrast, niching algorithms attempt to find multiple solutions to optimization problems.

Niching techniques are based on the competitive interaction of animal species in ecologies where resources are limited [2]. Niches can be seen as partitions of the environment (or problem search space), while species are simply the division of the population competing within the environment. Niching techniques can be broadly categorised into two different approaches: *parallel* and *sequential* niching. 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.

Niching principles have been applied to multimodal problems specifically using genetic algorithms [11], [12]. These algorithms include fitness sharing [13], dynamic niche sharing [14], sequential niching [15], crowding [16], deterministic crowding [11], probabilistic crowding [17], restricted tournament selection [18], coevolutionary shared niching [19] and dynamic niche clustering [20], [21]. PSO based niching techniques include the 'stretched' PSO [22], the *nbest* PSO [1] and NichePSO [3].

Some authors have investigated the scalability of their niching techniques [23], [11], [2]. The rest of this paper compares the scalability of DC, SN and NichePSO. These techniques are introduced in the next section.

B. Niching Algorithms

Sequential niching is a simple algorithm that identifies multiple solutions in a problem space by adapting the objective function's fitness landscape through the application of a *derating* function at a position where a potential solution was found [15]. After a possible solution is identified, the algorithm is restarted to search for other solutions. Repeating this process a sufficient number of times, locates multiple, unique solution. A derating function inhibits the fitness appeal of areas surrounding previously located solutions in a problem search space. For each iteration of the algorithm, the location of the located solution is remembered and the fitness of individuals in subsequent runs are adapted using

$$M_{t+1}(\mathbf{x}) \equiv M_t(\mathbf{x}) \times G(\mathbf{x}, \mathbf{s}_t) \tag{8}$$

where $M_0(\mathbf{x}) \equiv f(\mathbf{x})$, $f(\cdot)$ is a fitness function and $\mathbf{s}_t$ is the best individual found during run t of the algorithm.

Deterministic Crowding (DC), based on work done in [16] by De Jong, evolves a population of individuals by deriving offspring from parents and then letting them compete against each other for a position in a next generation. The algorithm can be summarised as follows (pseudo-code from [11]):

```
Repeat for $g$ generations:
 1. Do $n/2$ times:
 (a) Select 2 parents, $p_1$ and $p_2$, randomly from $C_g$.
 (b) Cross $p_1$ and $p_2$, yielding $c_1$ and $c_2$.
 (c) Apply mutation/other operators, yielding $c_1^{'}$ and $c_2^{'}$.
 (d) IF $[d(p_1, c_1^{'}) + d(p_2, c_2^{'})] \leq [d(p_1, c_2^{'}) + d(p_2, c_1^{'})]$
   • If $f(c_1^{'}) > f(p_1)$ replace $p_1$ with $c_1^{'}$
   • If $f(c_2^{'}) > f(p_2)$ replace $p_2$ with $c_2^{'}$
 ELSE
   • If $f(c_2^{'}) > f(p_1)$ replace $p_1$ with $c_2^{'}$
   • If $f(c_1^{'}) > f(p_2)$ replace $p_2$ with $c_1^{'}$
```

Note that $d(\cdot)$ is a phenotypic distance function. The repeated application of the above algorithm's replacement technique, leads to the development of

TABLE I
Rastrigin Parameters

n	# Solutions	Population Size Swarm Size
1	3	10
2	9	36
3	27	108
4	81	324

TABLE II
Griewank Parameters

n	# Solutions	Population Size Swarm Size
1	5	20
2	25	100
3	625	2500

similarity traits between individuals, leading to independent species that each occupies its own niche.

NichePSO is a PSO based technique that locates niches through the growing of subswarms from an initial swarm of particles [3]. The algorithm can be summarised as follows:

1. Initialize the main particle swarm.
2. Train main swarm particles using one iteration of the *cognition only* model (using equation (7)).
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.

The above algorithm, particularly methods used for the identification of niches, is analysed in [2]. The 'partitioning criteria' ensures that any particle traversing the search space in an area where a subswarm is already present is absorbed into that subswarm. The guaranteed convergence PSO (GCPSO) is presented in [8], [10].

IV. Experimental Results

A. Experimental Setup

This section summarises results of the application of SN, DC and NichePSO to higher dimensional multimodal functions with large numbers of solutions.

The following test functions were considered: The Griewank function

$$f(\mathbf{x}) = \left(\frac{1}{4000}\sum_{i=1}^{n} x_i^2\right) - \left(\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)\right) + 1 \quad (9)$$

illustrated in figure 1, and the Rastrigin function:

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

illustrated for three dimensions in figure 2. Note that both figures are drawn inverted to better illustrate the multimodal nature of their surfaces. Both contain a single global minimum at the origin. With an increase in the number of dimensions, n, the number of minima increases exponentially (as indicated in tables I and II).

Since both NichePSO and DC are dependent on the number of individuals or particles to locate all the possible solutions in a search space, swarm and population sizes used are indicated in tables I and II. The indicated swarm and population sizes were based on suggestions made in [2]. The SN algorithm does not search for different solutions concurrently, and therefore used a static population size of 30 individuals for all experiments.

For Beasley *et al*'s SN algorithm, the following settings, as used in [15], 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 [24]. For each dimension of the test functions, a 30-bit bitstring was used. The *halting window* approach described in [15], 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. To determine the SN niche radius, the method suggested by Beasley *et al* was used (originally suggested by Deb [25]). For an n-dimensional problem with l optima, the niche radius r was calculated as

$$r = \frac{\sqrt{n}}{2 \times \sqrt[n]{l}} \quad (11)$$

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

TABLE III
RASTRIGIN RESULTS

n	SN % Accuracy	DC % Accuracy	NichePSO % Accuracy
1	100.00%	100.00%	100.00%
2	78.00%	67.00%	100.00%
3	66.67%	61.11%	97.45%
4	58.02%	54.23%	97.08%

function G_e,

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

was used; $\mathbf{x}$ 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 III). Crossover and mutation probabilities were set at

Parameter	Value
p_c	1.0
p_m	0.01

since the DC algorithm favors a very low mutation probability and a high crossover probability [11]. DC also used a halting window-based termination criterion of $h = 20$.

For NichePSO, swarm sizes were as given in tables I and II. A halting window approach, similar to that referenced above was implemented. Halting window conditions were applied to all swarms, and the algorithm was executed until the main swarm was empty. The inertia weight was linearly scaled from 0.7 to 0.2, over a maximum of 2000 iterations of the main and subswarm 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. Parameter settings satisfied the relationship $w > \frac{1}{2}(c_1 + c_2) - 1$ at all times [8].

B. Results

Tables III and IV summarises performance results of the niching algorithms on the multimodal test functions. The term '*% Accuracy*' indicates the average number of solutions found by each of the algorithms. It is clear that although the GA techniques are simple and easy to implement, they tend to not scale well to more complex problem domains. As identified in [24], when high numbers of solutions are present, the SN algorithm becomes slow due to the compound nature of its modified fitness function (see equation(8)). Both the DC and NichePSO time complexity is increased by the fact that they require large numbers of individuals or particles to locate all solutions in the search space.

The subswarm technique employed in the NichePSO effectively handles large numbers of solutions, and is not hindered by an increase in problem dimensionality. In contrast, SN and DC do not perform well (regardless of empirical results to the contrary, in the case of DC [11]).

V. Conclusion and Future Work

Niching algorithms present real alternatives to unimodal optimization techniques in multimodal domains. Problems with multimodal solutions spaces, such as solving systems of equations and ensemble neural networks, can benefit from optimization approaches that explicitly assume more than a single solution, and lead to more efficient optimization.

This paper investigated the scalability of two well known GA niching algorithms, and compared it to that of a recent PSO-based niching technique, NichePSO. It was found that NichePSO performs better on more complex problem domains when compared to the GA niching algorithms.

The findings in this paper serves to confirm that algorithms introduced and testing on simple problems, do not necessarily always scale to more complex domains.

References

[1] R. Brits, A. P. Engelbrecht, and F. van den Bergh, "Solving Systems of Unconstrained Equations using Particle Swarm Optimizers," *Accepted for IEEE Conference on Systems, Man and Cybernetics*, October 2002.

[2] R. Brits, "Niching strategies for particle swarm optimization," Master's thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

[3] R. Brits, A. P. Engelbrecht, and F. van den Bergh, "A Niching Particle Swarm Optimizer," in *Proceedings of the Conference on Simulated Evolution and Learning*, (Singapore), pp. ? – ?, November 2002.

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley, Reading, MA, 1989.

[5] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. IV, (Perth, Australia), pp. 1942 – 1948, 1995.

[6] J. Kennedy, "Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance," *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1931 – 1938, July 1999.

[7] P. N. Suganthan, "Particle Swarm Optimizer with Neighborhood Operator," *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1958 – 1961, July 1999.

[8] F. van den Bergh, *An Analysis of Particle Swarm Op-*

timizers. PhD thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

[9] J. Kennedy, "Stereotyping: Improving Particle Swarm Performance with Cluster Analysis," in *Proceedings of the IEEE Congress on Evolutionary Computation*, (San Diego, USA), pp. 1507 – 1512, 2000.

[10] F. van den Bergh and A. P. Engelbrecht, "A New Locally Convergent Particle Swarm Optimizer," *Accepted for IEEE Conference on Systems, Man and Cybernetics*, October 2002.

[11] S. W. Mahfoud, *Niching Methods for Genetic Algorithms.* PhD thesis, Genetic Algorithm Lab, University of Illinois, Illinois, 1995. IlliGAL Rep. 95001.

[12] J. Horn, *The nature of niching: Genetic algorithms and the evolution of optimal, cooperative populations.* PhD thesis, Urbana, University of Illinois, Illinois, Genetic Algorithm Lab, 1997.

[13] D. E. Goldberg and J. Richardson, "Genetic Algorithm with Sharing for Multimodal Function Optimization," in *Proceedings of the Second International Conference on Genetic Algorithms*, pp. 41 – 49, 1987.

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

[15] D. Beasley, D. R. Bull, and R. R. Martin, "A Sequential Niching Technique for Multimodal Function Optimization," *Evolutionary Computation*, vol. 1, no. 2, pp. 101 – 125, 1993.

[16] K. A. de Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, Department of Computer Science, University of Michigan, Ann Arbor, Michigan, USA, 1975.

[17] O. J. Mengshoel and D. E. Goldberg, "Probabilistic Crowding: Deterministic Crowding with Probabilistic Replacement," in *Proceedings of the Genetic and Evolutionary Computation Conference 1999*, (San Fransisco, USA, Morgan Kaufmann), pp. 409 – 416, 1999.

[18] G. R. Harik, "Finding Multimodal Solutions Using Restricted Tournament Selection," tech. rep., IlliGAL, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1995.

[19] D. E. Goldberg and L. Wang, "Adaptive Niching via Coevolutionary Sharing," tech. rep., Genetic Algorithm Lab, Urbana, University of Illinois, Illinois, August 1997. IlliGAL Rep. 97007.

[20] J. Gan and K. Warwick, "Dynamic Niche Clustering: A Fuzzy Variable Radius Niching Technique for Multimodal Optimization in GAs," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. I, pp. 215 – 222, 2001.

[21] J. Gan and K. Warwick, "A Variable Radius Niche Technique for Speciation in Genetic Algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 96 – 103, Morgan-Kaufmann, 2000.

[22] K. E. Parsopoulos and M. N. Vrahatis, "Modification of the Particle Swarm Optimizer for Locating all the Global Minima," in *Artificial Neural Networks and Genetic Algorithms* (V. Kurkova, N. Steele, R. Neruda, and M. Karny, eds.), pp. 324 – 327, Springer, 2001.

[23] D. E. Goldberg, K. Deb, and J. Horn, "Massive Multimodality, Deception and Genetic Algorithms," in *Parallel Problem Solving from Nature* (R. Maenner and B. Manderick, eds.), vol. 2, pp. 37 – 46, 1992.

[24] S. Mahfoud, "A Comparison of Parallel and Sequential Niching Methods," in *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 136 – 143, 1995.

[25] K. Deb, "Genetic Algorithms in Multimodal Function Optimization," Master's thesis, Department of Engineering Mathematics, University of Alabama, 1989.

TABLE IV
GRIEWANK FUNCTION

Dims.(n)	**SN % Accuracy**	**DC % Accuracy**	**NichePSO % Accuracy**
1	100.00%	90.00%	100.00%
2	68.00%	66.60%	100.00%
3	41.00%	56.59%	94.75%

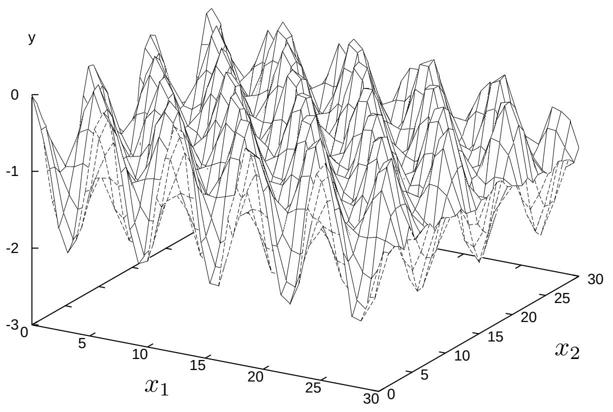

Fig. 1. Griewank Function

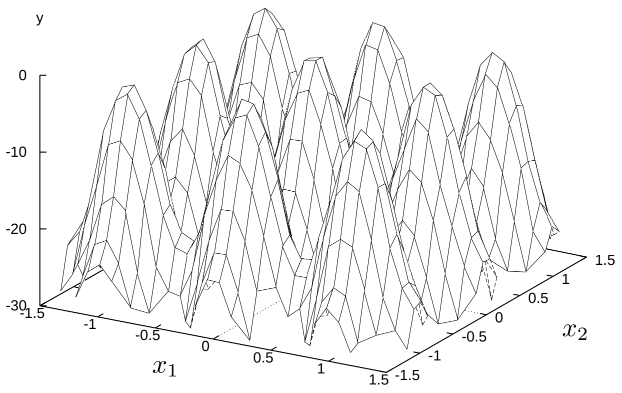

Fig. 2. Rastrigin Function