Locating and tracking multiple dynamic optima by a particle swarm model using speciation

Parrott, Daniel; Li, Xiaodong

Parrott, & Li, X. (2006). Locating and tracking multiple dynamic optima by a particle swarm model using speciation. IEEE Transactions on Evolutionary Computation, 10(4), 440–458. https://doi.org/10.1109/tevc.2005.859468

Published Version: https://doi.org/10.1109/tevc.2005.859468

Locating and Tracking Multiple Dynamic Optima by a Particle Swarm Model Using Speciation

Daniel Parrott and Xiaodong Li, *Member, IEEE*

***Abstract*—This paper proposes an improved particle swarm optimizer using the notion of species to determine its neighborhood best values for solving multimodal optimization problems and for tracking multiple optima in a dynamic environment. In the proposed species-based particle swam optimization (SPSO), the swarm population is divided into species subpopulations based on their similarity. Each species is grouped around a dominating particle called the species seed. At each iteration step, species seeds are identified from the entire population, and then adopted as neighborhood bests for these individual species groups separately. Species are formed adaptively at each step based on the feedback obtained from the multimodal fitness landscape. Over successive iterations, species are able to simultaneously optimize toward multiple optima, regardless of whether they are global or local optima. Our experiments on using the SPSO to locate multiple optima in a static environment and a dynamic SPSO (DSPSO) to track multiple changing optima in a dynamic environment have demonstrated that SPSO is very effective in dealing with multimodal optimization functions in both environments.**

***Index Terms*—Multimodal optimization, optimization in dynamic environments, particle swam optimization (PSO), tracking optima in dynamic environments.**

I. Introduction

In RECENT YEARS, particle swarm optimization (PSO) has been used increasingly as an effective technique for solving complex and difficult optimization problems [12], [20], [21]. However, most of these problems handled by PSOs are often treated as a task of finding a single global optimum. In the initial PSO proposed by Eberhart and Kennedy [21], each particle in a swarm population adjusts its position in the search space based on the best position it has found so far, and the position of the known best-fit particle in the entire population (or neighborhood). The principle behind PSO is to use these particles with best known positions to guide the swarm population to converge to a single optimum in the search space.

How to choose the best-fit particle to guide each particle in the swarm population is a critical issue. This becomes even more acute when the problem being dealt with has multiple optima, as the entire swarm population can potentially be misled to local optima. One approach to combat this problem is to allow the population to search for multiple optima (either global or local) simultaneously. Striving to locate multiple optima has two advantages. First, by locating multiple optima, the likelihood of finding the global optimum is increased; second, when dealing with real-world problems, for some practical reasons, it is often desirable for the designer to choose from a diverse set of good solutions, which may be equally good global optima or even second best optima.

The uniqueness of PSO's ability in adaptively adjusting particles' positions based on the dynamic interactions with other particles in the population makes it well suited for handling multimodal optimization problems. If suitable particles can be determined as the appropriate neighborhood best particles to guide different portions of the swarm population moving toward different optima, then essentially we will be able to use a PSO to optimize over a multimodal fitness landscape. Ideally, multiple optima will be found. The question is how to determine which particles would be suitable as neighborhood bests; and how to assign them to the suitable particles in the population so that they will move toward different optima accordingly.

An environment that is both multimodal and dynamic such as that shown in Fig. 1 presents additional challenges. In fully dynamic multimodal environments, optima may shift spatially, change both height and shape, or come into or go out of existence. Hence, the height of the global optimum may decrease, while the height of a local optimum increases, eventually changing not just the position of the global optimum as optima shift but also the optimum, which represents the global optimum. An optimum may disappear as it is obscured by a higher optimum above it, or optima may appear or disappear entirely. To effectively search such a space requires answers to two further questions: how can particles update their known best positions in order to track optima and how can a population be arranged to balance the need to track existing optima against the need to distribute particles to search the remaining space for new optima?

The paper is organized as follows. Section II describes related work on multimodal optimization, and their relevance to the proposed species-based PSO (SPSO). Section III presents the classic PSO. Section IV introduces the notion of species and its relation to multimodal optimization. Section V describes the proposed SPSO in a static multimodal environment, including the performance measures, test functions, experimental setup, results, and discussion. These topics are again covered in Section VI for the dynamic SPSO (DSPSO) in a dynamic environment. Finally, Section VII draws some conclusions and gives directions for future research regarding the SPSO.

II. Related Work

Significant research into multimodal optimization and optimization in dynamic environments has been conducted using evolutionary algorithms (EAs). However, these two fields are

Manuscript received June 14, 2004; revised January 17, 2005.

The authors are with the School of Computer Science and Information Technology, RMIT University, Melbourne, Victoria 3001, Australia (e-mail: dparrott@cs.rmit.edu.au; xiaodong@cs.rmit.edu.au).

Digital Object Identifier 10.1109/TEVC.2005.859468

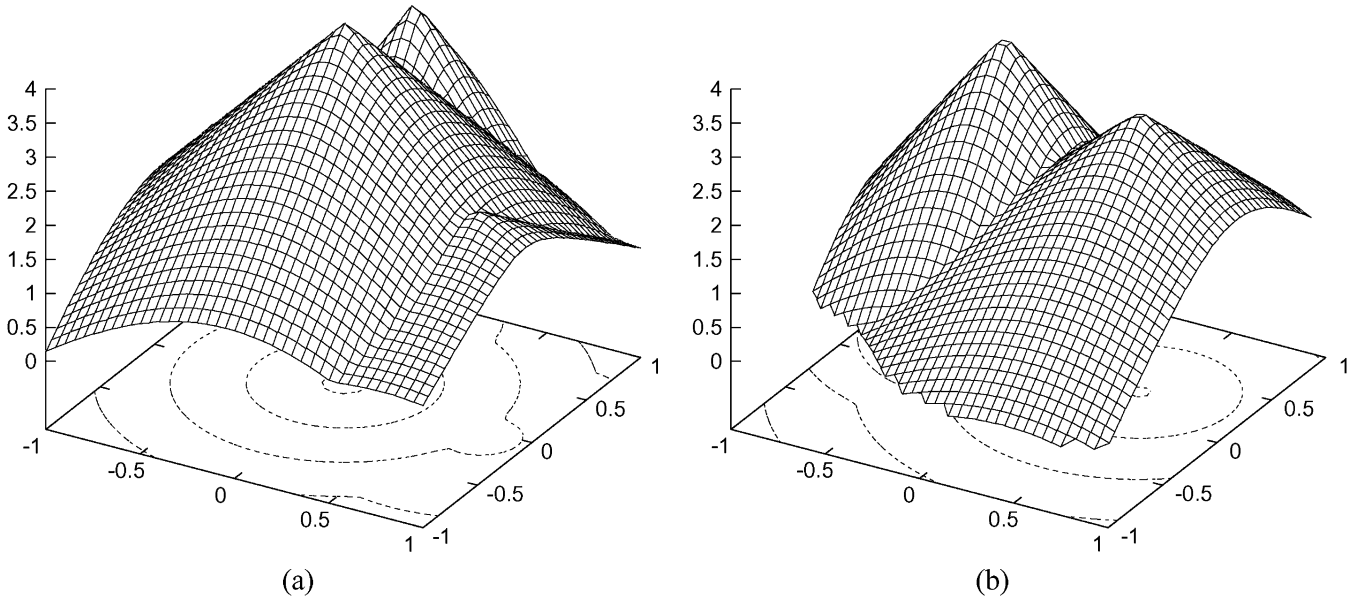

Fig. 1. Three peak multimodal environment, before and after movement of optima. Note that the small peak to the right of the figure becomes hidden and that the highest point switches optimum. (a) Before. (b) After.

typically treated separately and only recently have EAs for environments which are both multimodal and dynamic been proposed (e.g., [2], [4], [6], [18], and [33]). A recent survey on EAs in dynamic environments has been provided by Jin and Branke [18].

A. Locating Multiple Optima in a Static Environment

Although multimodal function optimization has been studied extensively by EA researchers [1], [24], only a few works have investigated using particle swarm models. In [19], Kennedy proposed a PSO using a k-means clustering algorithm to identify the centers of different clusters of particles in the population, and then use these cluster centers to substitute the personal bests or neighborhood bests. In [19], for a population of 20 particles, it was arbitrarily decided to use five clusters, and to iterate three times in order to allow the cluster centers to be stabilized. Some serious limitations of this clustering approach can be identified.

- In order to calculate the cluster centers, the method requires three iterations over all individuals in the population. However, the number of clusters and iterations must be predetermined. This can be difficult since they are problem dependent.
- A cluster center identified is not necessarily the best-fit particle in that cluster. Consequently, using these cluster centers as *lbest* (or neighborhood best) is likely to lead to poor performance (see Fig. 2).

Brits *et al.* in [8] proposed a *nbest* PSO algorithm which defines the "neighborhood" of a particle as its n closest particles of all particles in the population (measured in Euclidean distance). The *neighborhoodbest* for each particle is defined as the average of the positions of its n closest particles. Like Kennedy's k-means clustering PSO, *nbest* also suffers from the same problem that the neighborhood best is not always the best-fit particle in that neighborhood, and the parameter n must also be prespecified by a user.

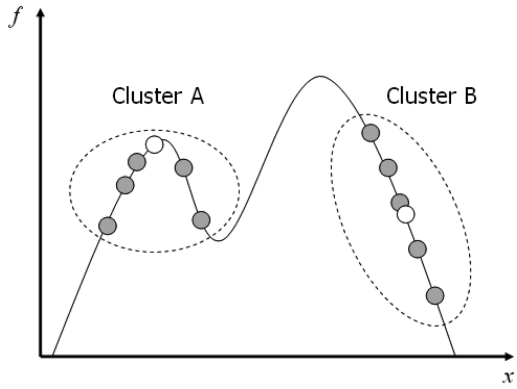

Fig. 2. Cluster A's center (white circle) performs better than all members of the cluster A, whereas cluster B's center performs better than some and worse than others (see also [19]).

In [29] and [30], Parsopoulos and Vrahitis observed that when they applied the *gbest* method (i.e., the swarm population only uses a single global best) to a multimodal function, the swarm moved back and forth, failing to decide where to land. This behavior is largely caused by particles getting equally good information from those equally good global optima. To overcome this problem, they introduced a method in which a potentially good solution is isolated once it is found (if its fitness is below a threshold value ϵ), then the fitness landscape is "stretched" to keep other particles away from this area of the search space (similar to the derating method used in sequential niched genetic algorithm (SNGA) proposed by Beasley *et al.* in [1]). The isolated particle is checked to see if it is a global optimum, and if it is below the desired accuracy, a small population is generated around this particle to allow a finer search in this area. The main swarm continues its search for the rest of the search space for other potential global optima. With this modification, their PSO was able to locate all the global optima of some selected test functions successfully. A major issue associated with

the stretching and derating methods is that applying such techniques often introduces new local optima. More detail on this can be found in [1].

Brits *et al.* proposed a niching particle swarm optimization (NichePSO) algorithm [7], which has a number of improvements over Parsopoulos and Vrahitis's model. NichePSO incorporates a cognitive only PSO model and the guaranteed convergence particle swarm optimization (GCPSO) algorithm which guarantees a PSO to converge to a local optimum [34]. In NichePSO, multiple subswarms are produced from a main swarm population to locate multiple optimal solutions in the search space. Subswarms can merge together, or absorb particles from the main swarm. Instead of using the threshold ϵ as in Parsopoulos and Vrahitis's model, NichePSO monitors the fitness of a particle by tracking its variance over a number of iterations. If there is little change in a particle's fitness over a number of iterations, a subswarm is created with the particle's closest neighbor. The authors used a swarm of population size of 20–30, and NichePSO found all global optima of the test functions used within 2000 iterations.

Although numerous niching techniques have been proposed in the past to tackle multimodal optimization problems, most of these algorithms have only been tested on problems with two dimensions and a limited number of optima [1], [7], [29], [32]. Some exceptions are [9] where Brits *et al.* tested their NichePSO on the Rastrigin function with up to five dimensions, and [24] where Li *et al.* tested their species conserving genetic algorithm (SCGA) on a Shubert function with up to four dimensions.

In [35], the attractive and repulsive PSO (arPSO) was proposed by Vesterstroem and Riget. In arPSO, the basic PSO's velocity equation is modified when the population diversity is below a threshold. This modification simply corresponds to repulsion of the particles instead of the usual attraction equation. When the diversity becomes higher than another threshold, the commonly used PSO equation will be brought back to action again.

Petrowski in [31] introduced a clearing procedure as a niching method, and more recently, Li *et al.* in [24] introduced a SCGA for multimodal optimization. Both the clearing procedure and SCGA adopted a technique for dividing the population based on the notion of species, which is added to the evolution process of a conventional genetic algorithm. Their results on multimodal optimization have shown to be substantially better than those found in literature.

The notion of species is very appealing. To some extent, it provides a way of addressing the limitations we identified with the clustering approach used in the PSO proposed by Kennedy [19]. This paper describes a SPSO incorporating the idea of species into PSO for solving the multimodal optimization problems. At each iteration step, SPSO aims to identify multiple species (each for a potential optimum) within a population, and then determines a neighborhood best for each species. These multiple adaptively formed species are then used to optimize toward multiple optima in parallel, without interference across different species. SPSO was first proposed by the authors [25] and [27]. This paper describes an improved version of SPSO with a mechanism for removing redundant duplicate particles in species for better performance efficiency, and also provides an analysis over the effects of population size and species radius. We also compare the experimental results between SPSO and NichePSO (a niche-based PSO algorithm for multimodal optimization), over a chosen set of benchmark multimodal test functions.

B. Tracking Multiple Optima in a Dynamic Environment

Most research into particle swarm models for dynamic environments has been restricted to swarms capable of tracking a single optimum, with swarm algorithms to track multiple optima simultaneously being a recent occurrence [4], [27]. Eberhart and Shi tested the particle swarm model in a dynamic environment tracking an optimum generated by the parabolic function in three dimensions [15]. The optimum's position changed up to 2% with changes occurring every 100 generations. They found that the particle swarm model compared favorably with other evolutionary methods for the same number of function evaluations. Carlisle and Dozier [10] noted that the particle swarm tracked a moving optimum successfully only when movements in the optimum's position are small. They adapted the model to track larger movements of the optimum by causing particles to replace their best-known positions with current positions. Periodic and triggered replacements of best-known positions with current positions were tested and found to result in successful tracking of the optimum. Monitoring environments for changes has also been used by Hu and Eberhart [17]. On detection of a change, a fixed proportion of particles was rerandomized in order to find the new location of the optimum. In all these methods, the swarm is modified to track a moving optimum by causing some particles to lose all knowledge of known optima's positions.

Blackwell and Bentley [2], [3] have investigated the use of swarms of charged particles in environments with one or two optima, created by the parabolic function in three dimensions, experiencing severe changes every 100 iterations or less. The use of charged particles prevents a swarm from converging to a single point and forces it to remain spread out, enabling it to better respond to change and relocate moved optima. This, however, incurs the cost of an $O(N^2)$ calculation of the repulsive forces between particles arising from charge. By using a swarm with 1/2 the particles charged and 1/2 neutral (called an atomic swarm because of the similarity to electrons circling the nucleus of an atom), an algorithm with good search and convergence characteristics was arrived at.

The atomic swarm approach has been adapted to track multiple optima simultaneously with multiple swarms ("multiswarms") by Blackwell and Branke [4]. The number of swarms is specified in advance. When swarms approach within a specified radius of each other the swarm with the worse value at its attractor or *gbest* position is randomized. In this way, multiple swarms are prevented from converging to the same peak. The atomic swarm approach is also modified to quantum swarms—rather than use charge-repulsions between particles to encourage diversity the algorithm uses quantum particles, whose position is solely based on a probability function centered around the swarm attractor. $O(N^2)$ particle-to-particle comparisons are thus avoided. The resulting multiquantum swarm outperforms charged particle and standard particle swarms on the moving peaks function with ten optima, with best results are obtained where the number of swarms is equal or close to the number of optima.

In the Self Organizing Scouts approach of Branke *et al.* [6], the population is divided into a parent population that searches the solution space and child populations that track known optima. The parent population is periodically analyzed for clusters of partly converged individuals which are split off as child populations centered on the best individual in the child population. Members of the parent population are then excluded from the child population's space. The size of child populations is altered to give large populations to optima demonstrating high fitness or dynamism.

Ursem has demonstrated a multipopulation method named the Multinational GA [33]. This method uses multiple GA populations known as 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 (2-D) environments.

These last three approaches indicate three key requirements of an EA operating in a dynamic multimodal environment:

- they find multiple optima in parallel;
- they track optima once located;
- they repeatedly balance the need to exploit known optima with the need to search for new optima.

In the following sections, we will begin by describing a SPSO that can accomplish the first of these tasks in a static environment. This SPSO is then modified to operate in a dynamic environment by accomplishing the second and third tasks. The DSPSO is shown to be effective in tracking multiple optima in dynamic environments.

III. Particle Swarm

The particle swarm algorithm is an optimization technique inspired by the metaphor of social interaction observed among insects or animals. The kind of social interaction modeled within a PSO is used to guide a population of individuals (so-called particles) moving toward the most promising area of the search space. In a PSO algorithm, each particle is a candidate solution equivalent to a point in a D-dimensional space, so the ith particle can be represented as $\vec{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$. Each particle "flies" through the search space, depending on two important factors: $\vec{p}_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$, the best position the current particle has found so far; and $\vec{p}_g = (p_{g1}, p_{g2}, \ldots, p_{gD})$, the global best position identified from the entire population (or within a neighborhood). The rate of position change of the ith particle is given by its velocity $\vec{v}_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$. Equation (1) updates each dimension of the velocity for the ith particle $\vec{v}_i$ for the next iteration step, whereas (2) updates the ith particle's position in the search space [20]

$$v_{id}(t) = \chi(v_{id}(t-1) + \varphi_1 r_1(p_{\mathrm{id}} - x_{\mathrm{id}}(t-1)) + \varphi_2 r_2(p_{gd} - x_{\mathrm{id}}(t-1))) \quad (1)$$

$$x_{\mathrm{id}}(t) = x_{\mathrm{id}}(t-1) + v_{\mathrm{id}}(t) \quad (2)$$

```
Randomly generate an initial population
repeat
    for i = 1 to Population Size do
        if f(x_i) < f(p_i) then p_i = x_i;
        p_g = min(p_neighbors);
        for d = 1 to D do
            equation (1);
            equation (2);
        end
    end
until termination criterion is met;
```

Fig. 3. Pseudocode of a basic PSO for minimization.

where

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \quad \text{and} \quad \varphi = \varphi_1 + \varphi_2, \quad \varphi > 4.0. \quad (3)$$

The constriction factor χ produces a damping effect on the amplitude of an individual particle's oscillations, and as a result, the particle will converge over time. φ_1 and φ_2 represent the cognitive and social parameters, respectively. r_1 and r_2 are random numbers uniformly distributed in $[0, 1]$. The velocity $\vec{v}_i$ is confined within the range of $[-V_{\mathrm{MAX}}, +V_{\mathrm{MAX}}]$. It is suggested that using V_{MAX} in conjunction with the constriction factor may be a good idea [20], though it is not necessary.

Two common approaches of choosing $\vec{p}_g$ are known as *gbest* and *lbest* methods. In a *gbest* PSO, the position of each particle in the search space is influenced by the best-fit particle in the entire population, whereas a *lbest* PSO only allows each particle to be influenced by the best-fit particle chosen from its neighborhood. The *lbest* PSO with a neighborhood size set to the population size is equivalent to a *gbest* PSO. Fig. 3 shows the pseudocode of a basic particle swarm algorithm for minimization [12]. Kennedy and Mendes studied PSO with various population topologies [22], and have shown that certain population structures could give superior performance over certain optimization functions.

IV. Identifying Species

Central to the proposed SPSO, in this paper is the notion of species. A species can be defined as a group of individuals sharing common attributes according to some similarity metric. This similarity metric could be based on the Euclidean distance for genotypes using a real-coded representation, or the Hamming distance for genotypes with a binary representation. The smaller the Euclidean (or the Hamming) distance between two individuals, the more similar they are.

The definition of species also depends on another parameter r_s, which denotes the radius measured in Euclidean distance from the center of a species to its boundary. The center of a species, the so-called species seed, is always the fittest individual in the species. All particles that fall within the r_s distance from the species seed are classified as the same species.

input : L_{sorted} - a list of all particles sorted in decreasing fitness values
output: S - a list of all dominating particles identified as species seeds

begin
  $S = \emptyset$;
  while *not reaching the end of* L_{sorted} **do**
    Get best unprocessed $p \in L_{sorted}$;
    $found \leftarrow$ FALSE;
    for *all* $s \in S$ **do**
      if $d(s,p) \leq r_s$ **then**
        $found \leftarrow$ TRUE;
        break;
      end
    end
    if *not* $found$ **then**
      let $S \leftarrow S \cup \{p\}$;
    end
  end
end

Fig. 4. Algorithm for determining species seeds.

A. Determining Species Seeds From the Population

The algorithm for determining species seeds, introduced by Petrowski in [31] and also Li *et al.* in [24], is adopted here. By applying this algorithm at each iteration step, different species seeds can be identified for multiple species and then used as the *lbest* for different species accordingly. Fig. 4 summarizes the steps for determining the species seeds.

The algorithm (as given in Fig. 4) for determining the species seeds is performed at each iteration step. The algorithm takes as an input L_{sorted}, a list containing all particles sorted in decreasing order of fitness. The species seed set S is initially set to $\emptyset$. All particles are checked in turn (from best to least-fit) against the species seeds found so far. If a particle does not fall within the radius r_s of all the seeds of S, then this particle will become a new seed and be added to S. Fig. 5 provides an example to illustrate the working of this algorithm. In this case, applying the algorithm will identify s_1,, s_2, and s_3 as the species seeds. Note also that if seeds have their radii overlapped (e.g., s_2 and s_3 here), the first identified seed (such as s_2) will dominate over those seeds identified later from the list L_{sorted}. For example, s_2 dominates s_3, therefore p should belong to the species led by s_2.

B. Choosing Neighborhood Bests

Since a species seed is the fittest particle in a species, other particles within the same species can be made to follow the species seed as the newly identified neighborhood best (*lbest*). This allows particles within the same species to be attracted to positions that make them even fitter. Because species are formed around different optima in parallel, making species seeds the new neighborhood bests provides the right guidance for particles in different species to locate multiple optima.

Since species seeds in S are sorted in the order of decreasing fitness, the more highly fit seeds first get allocated with particles before the less fit seeds in S. This also helps the algorithm to locate the global optima before local ones.

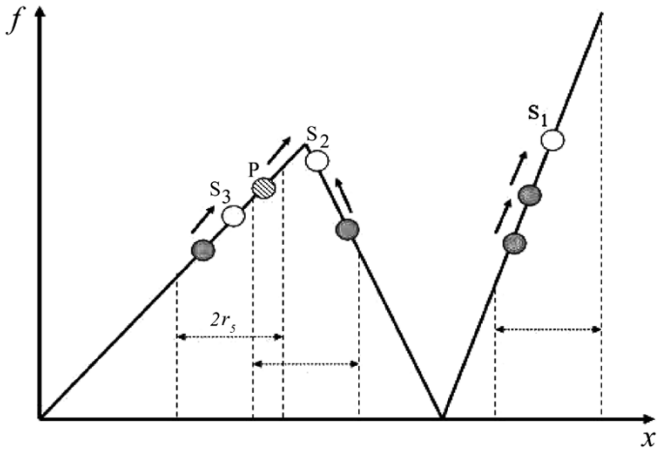

Fig. 5. Example of how to determine the species seeds from the population at each iteration step. s_1, s_2, and s_3 are chosen as the species seeds. Note that p follows s_2.

C. Complexity

The complexity of the algorithm for determining species seeds (Fig. 4) can be estimated based on the number of evaluations of Euclidean distances between two particles that are required. Assuming there are N individuals sorted and stored in list L_{sorted}, the **while** loop steps through $L_{\text{sorted}}N$ times to see if each individual is within the radius r_s of the seeds in S. If S currently contains i number of seeds, then at best the **for** loop is executed only once when the particle considered is within r_s of the first seed compared; and at worst the **for** loop is executed i times when the particle falls outside of r_s of all the seeds on S. Therefore, the number of Euclidean distance calculations required for this procedure, $T(N)$, can be obtained by the following [24]:

$$N \leq T(N) \leq \sum_{i=1}^{N}(i-1) = \frac{N(N-1)}{2} \tag{4}$$

which shows that the worst time complexity of the procedure is $O(N^2)$ when there are N species seeds. However, it is important to note that a much tighter upper bound, $\bar{N}_s \cdot N$, can be derived, where $\bar{N}_s$ is the upper bound of the number of species that will be found at each iteration. $\bar{N}_s$ can be estimated according to the size of the search space and r_s. Typically, $\bar{N}_s \ll N$. In such a case, the above procedure takes roughly $\bar{N}_s \cdot N$ Euclidean distance calculations at each iteration. Since $\bar{N}_s$ is determined by r_s but not the population size N, this procedure in fact has a linear time complexity of $O(N)$.

Note that the above procedure for determining species seeds assumes that all particles must have been sorted in decreasing fitness values at each iteration. The number of comparisons involved in this sorting procedure follows a complexity of $O(N \log N)$. If the sorting procedure is taken into account, SPSO's complexity would be $O(N \log N)$.

The k-means clustering-based PSO as suggested by Kennedy [19] has a complexity of $O(K \cdot N \cdot \text{Iter})$, where K is the prespecified number of clusters, N the population size, and Iter the number of iterations required before the cluster centers become stabilized. The main drawback with the k-means clus-

```
input : L_sorted - A list of all particles sorted in decreasing fitness values
output: S - A list of all dominating particles identified as species seeds
begin
    initialSize = L_sorted.size() ;
    for i = 1 to initialSize do
        if L_sorted[i].fitness = L_sorted[i].seedFitness then
            Remove the i-th particle in L_sorted
        end
        ;
    end
    Add all seeds in S back to L_sorted ;
    while L_sorted.size() < initialSize do
        Randomly generate a new particle p⃗ ;
        Add p⃗ to L_sorted ;
    end
end
```

Fig. 6. Algorithm for replacing redundant particles.

tering technique is the difficulty in predetermining appropriate K and Iter values.

The average complexity of SPSO and k-means based PSO are difficult to calculate since they are problem dependent. However, it can be said that the key advantage of the procedure for determining species seeds is that it provides a natural and effective mechanism in determining *lbest* which allows SPSO to locate multiple optima efficiently. In contrast, the cluster centers identified via the k-means clustering technique might be misguiding the swarm population most of the time (see Fig. 2).

V. Species-Based PSO (SPSO)

Once the species seeds have been identified from the population, we can then allocate each seed to be the *lbest* to all the particles in the same species at each iteration step. The SPSO accommodating the algorithm for determining species seeds described above can be summarized in the following steps.

- Step 1) Generate an initial population with randomly generated particles.
- Step 2) Evaluate all particle individuals in the population.
- Step 3) Sort all particles in descending order of their fitness values (i.e., from the best-fit to least-fit ones).
- Step 4) Determine the species seeds for the current population (see Fig. 4).
- Step 5) Assign each species seed identified as the *lbest* to all individuals identified in the same species.
- Step 6) Replace redundant particles in species (see Fig. 6 in the next section).
- Step 7) Adjusting particle positions according to (1) and (2).
- Step 8) Go back to Step 2), unless the termination condition is met.

Comparing with Kennedy's clustering-based PSO [19] (also discussed in Section II-A), SPSO improves in the following two aspects.

- SPSO only requires calling the procedure for determining species once in order to determine the species seeds, which are used as substitutes for neighborhood bests. In contrast, identifying the cluster centers in Kennedy's PSO would normally require a prespecified number of iterations. Note that this number can be difficult to determine in advance, in order to have the cluster centers stabilized.
- SPSO provides a natural and effective mechanism in identifying species seeds as neighborhood bests, thereby providing a better guidance to the particle population in locating multiple optima. In SPSO, an identified species seed is always the best-fit individual in that species, however, this is not always the case in Kennedy's clustering-based PSO.

A. Removing Redundant Particles in Species

One potential problem with regard to efficiency (as well as to locating a complete set of global optima) of SPSO is that toward the end of a run, it is quite common that some particles within a species may have converged to one optimum before other species reach convergence on other optima. Any additional particles that have converged on the same global optimum (one of multiple global optima needed to be located) are considered to be *redundant* since they do not contribute further to the improvement of convergence. We can make better use of these redundant particles by replacing them with randomly generated particles so that other parts of the search space can be better explored. Since our objective is to locate multiple optima, the substitution of redundant particles by randomly generated particles should also increase the likelihood of SPSO in finding more optima than otherwise. Fig. 6 shows the pseudocode of the procedure for removing additional duplicated particles when particles are identified as having the same fitness with the species seed within the same species. Once all duplicated particles are removed from L_{sorted}, the species seeds are then first added back to L_{sorted}. If there are more space on L_{sorted}, then add randomly generated new particles to L_{sorted} until its size is resumed to its initial size.

The procedure shown in Fig. 6 for replacing redundant particles in species can be called before the normal PSO updating algorithm given in Fig. 3.

B. Performance Measurements

We assume that the global optima of all test functions are known *a priori*. This allows us to focus on our two main goals in this research, that is to measure how accurately the SPSO locates multiple optima, as well as its convergence speed. More specifically, the performance of the SPSO can be measured in the following two ways.

1) Accuracy: The model is run for a number of fixed iteration steps, and the *accuracy*, which measures the closeness of the fittest solutions from all species to all the known optima, is recorded in the final iteration step. Note that each known optimum corresponds only to a different species seed which is the fittest particle in that species.

2) Convergence Speed: In this mode, an expected accuracy level is prespecified and the number of evaluations required to achieve the expected accuracy is recorded.

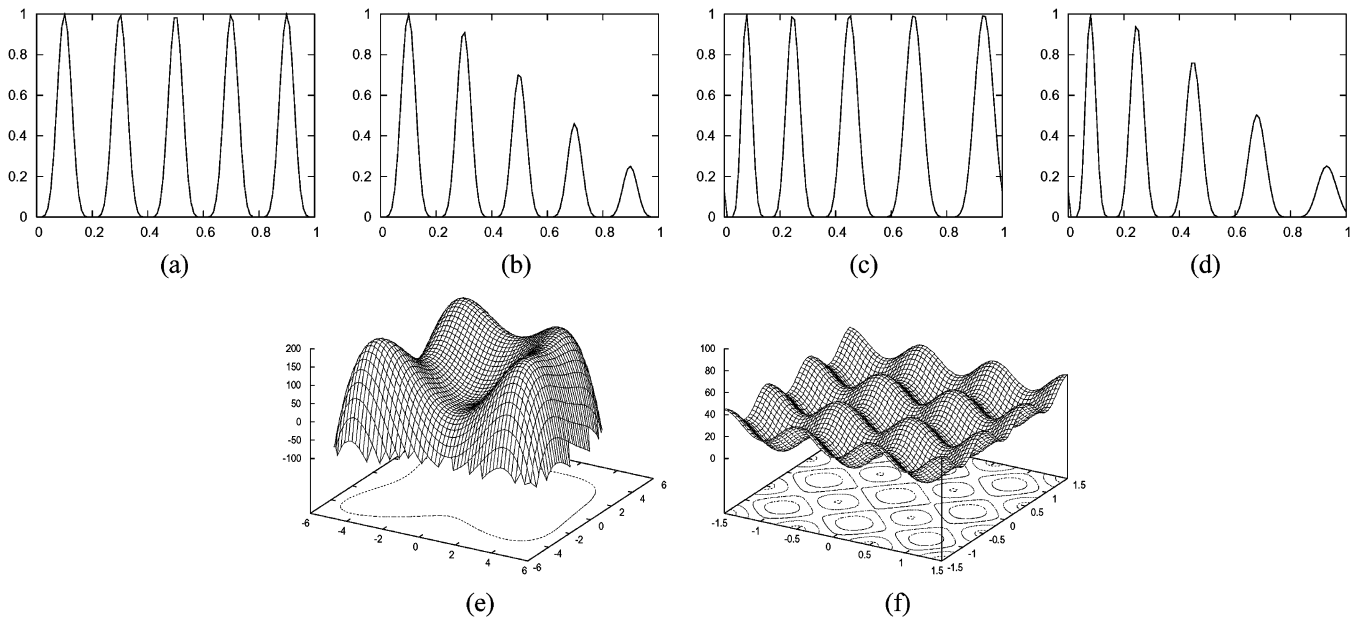

Fig. 7. Test functions. (a) F1. (b) F2. (c) F3. (d) F4. (e) F5: Himmelblau's function. (f) F6: Rastrigin function.

In addition, we also measure the performance in terms of success rate, i.e., the percentage of runs in which all global optima are successfully located.

Accuracy is calculated by taking the average of the fitness differences between all known global optima to their closest species seeds

$$\text{accuracy} = \frac{1}{|\text{opts}|} \sum_{j=1}^{|\text{opts}|} |\text{fit}(\text{opt}_j) - \text{fit}(\text{seed}_j)| \qquad (5)$$

where $|\text{opts}|$ gives the number of known global optima. A pair of opt_j and seed_j represents that for each optimum opt_j, there is correspondingly a closest species seed seed_j to opt_j (since normally opt_j is within the radius r_s of seed_j). This seed_j can be identified from S, the set of species seeds. If opt_j is not within r_s of any species seeds of S (which means opt_j is not found by SPSO), then $\text{fit}(\text{seed}_j)$ is simply set to 0, resulting in $|\text{fit}(\text{opt}_j) - \text{fit}(\text{seed}_j)|$ being equal to $|_{\text{fit}}(\text{opt}_j)|$, in which case the overall accuracy value will be degraded. Since a species seed is always the fittest individual in its species, (5) should give an accurate indication of how closely the algorithm identifies all the global optima. Note that suboptima of a test function are not taken into consideration in computing accuracy. Since our goal is to find a complete set of global optima, i.e., all the global optima of interest in a function, we allow the algorithm to run for 2000 iterations before termination to see if all known global optima are found.

To measure convergence speed at a required level of accuracy, we only need to check set S, which contains the species seeds identified so far. These species seeds are dominating individuals sufficiently different from each other, however, they could be individuals with high as well as low fitness values (see Fig. 5). We can decide if a global optimum is found by checking each species seed in S to see if it is close enough to a known global optimum. An expected accuracy acceptance threshold $(0 < \epsilon \leq 1)$ is defined to detect if the solution is close enough to a global optimum, and the following condition must be satisfied:

$$\forall x \in S_{\text{opt}} \exists y \in S\text{: } \min\{||x - y||\} \wedge |\text{fit}(x) - \text{fit}(y)| \leq \epsilon \quad (6)$$

where s_{opt} is a set of all known global optima of a multimodal function, and S is a set of identified species seeds (each should correspond closely to an optimum toward the end of a run). $\min\{||x-y||\}$ returns the closest pair of a global optimum (from s_{opt}) and a species seed (from S). Equation (6) states that for each global optimum x there must exist a species seed y such that the fitness difference between x to its closest species seed y is not greater than ϵ. This condition must be satisfied or the maximum number of iterations allowed is reached before a run can be terminated.

C. Test Functions

The five test functions suggested by Beasley *et al.* [1] and the Rastrigin function (with different dimensions) were used to test SPSO's ability to locate a single or multiple maxima.

As shown in Fig. 7, F1 has five evenly spaced maxima with a function value of 1.0. F2 has five peaks decreasing exponentially in height, with only one peak as the global maximum. F3 and F4 are similar to F1 and F2 but the peaks are unevenly spaced. F5 Himmelblau's function has two variables x and y, where $-6 \leq x, y \leq +6$. This function has four global maxima at approximately $(3.58, -1.86), (3.0, 2.0), (-2.815, 3.125)$, and $(-3.78, -3.28)$.

F6 Rastrigin function (where $-1.5 \leq x_i \leq 1.5, i = 1, \ldots, 10$) has one global minimum (which is (0, 0) for $\text{dimension} = 2$), and many local minima. The number of minima increases depending on the dimension. There are only nine minima (including the global minimum) for dimension 2 [Fig. 7(f)], however, this increases to 27, 81, 243, 729 minima for dimension 3, 4, 5, and 6, respectively. F6 with a dimension of 2, 3, and up to 10 variables were used to test SPSO's ability

TABLE I
TEST FUNCTIONS

Function	Range	Comments
$F1(x) = sin^6(5\pi x)$	[0, 1]	5 global optima with an equal height
$F2(x) = exp\left(-2log(2)\cdot\left(\frac{x-0.1}{0.8}\right)^2\right)\cdot sin^6(5\pi x)$	[0, 1]	1 global optimum and 4 local optima
$F3(x) = sin^6(5\pi(x^{3/4} - 0.05))$	[0, 1]	5 global optima unevenly spaced
$F4(x) = exp\left(-2log(2)\cdot\left(\frac{x-0.08}{0.854}\right)^2\right)\cdot sin^6(5\pi(x^{3/4} - 0.05))$	[0, 1]	1 global optimum and 4 local optima unevenly spaced
$F5(x, y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2$	[-6, 6]	4 global optima with an equal height
$F6(\vec{x}) = \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i) + 10)$	[-1.5, 1.5]	1 global optima and many local optima

TABLE II
RESULTS ON ACCURACY AFTER 2000 ITERATIONS SPSO (AVERAGED OVER 50 RUNS). NICHEPSO RESULTS WERE QUOTED FROM [7]. BOTH SPSO AND NICHEPSO HAVE ACHIEVED 100% SUCCESS RATE

Function	Num. of global optima	r_s	Accuracy (SPSO) (mean and std err)	Accuracy (NichePSO) (mean and std err)
F1	5	0.05	0.00 ± 0.00	7.68E-05 ± 3.11E-05
F2	1	0.05	4.00E-17 ± 2.26E-17	9.12E-02 ± 9.09E-03
F3	5	0.05	3.20E-14 ± 3.20E-14	5.95E-06 ± 6.87E-06
F4	1	0.05	1.72E-07 ± 0.00	8.07E-02 ± 9.45E-03
F5	4	2.00	2.19E-09 ± 2.19E-09	4.78E-06 ± 1.46E-06

in dealing with functions with numerous local minima and of higher dimensions.

D. Experiment Setups

For the test functions used in this paper, since the exact number of global optima and the distances between them are known, the species radius r_s was set normally to a value smaller than the distance between two closest global optima. By doing this, SPSO should be able to sufficiently distinguish two different optima.

For PSO parameters in (1) and (2), φ_1 and φ_2 were set to 2.05, hence a constriction factor χ of 0.729844 was obtained using (3) (see also PSO with Type 1" constriction in [12] and [22]). $\pm V_{\max}$ was set to be the lower and upper bounds of the allowed variable ranges (as suggested in [20, p. 342]).

E. Results on Convergence Accuracy

To test the accuracy of SPSO, a swarm population size of 30 was used for all the above test functions. SPSO was run 50 times, with each run terminated if 2000 iteration steps is reached. Success rate was measured by the percentage of runs (out of 50) locating all the global optima within the 2000 iteration steps, for the minimum of an expected accuracy $\epsilon = 0.0001$. r_s was set to 0.05 for F1 to F4, and 2.0 for F5, respectively. To make a fair comparison with NichePSO [7], we tried to use a setup that is as close as possible to that of NichePSO.

NichePSO's performance was measured in the same way as SPSO in terms of *accuracy* (i.e., how close the best particles in all subswarms are to the actual global optima) and *success rate*. Table II shows that though both SPSO and NichePSO have achieved 100% success rate, SPSO produced a better accuracy than NichePSO for all five test functions.

SPSO also compared well over these five functions to sequential niched GA (SNGA), proposed by Beasley *et al.* [1], though SNGA used a slightly different setting and performance metrics. SNGA used a population size of 100 and a sequence of multiple runs in order to locate all the optima, and it achieved a success rate of 99% on F1, 90% on F2, 100% on F3, 99% on F4, and 76% on F5, respectively (see [1, Table IV]).

Formation of different species in SPSO discourages interaction between particle species located on different optima, hence once different species converge on their corresponding optima, they would stay converged.

F. Results on Convergence Speed

To test the convergence speed of SPSO, we set the expected accuracy (i.e., the accuracy threshold) ϵ to 0.0001. The number of function evaluations required for finding all the global optima at the expected accuracy ϵ are averaged over 50 runs. A run is only terminated if either the required accuracy for all the global optima or the maximum of 2000 iteration steps is reached. Two population sizes were used, SPSO-s30 with a population size of 30, and SPSO-s50 with a population size of 50. The reason for doing so is to show that population size has a significant impact on SPSO's performance efficiency, that is, the number of evaluations can be substantially smaller if an appropriate population size is chosen.

Table III shows the required number of evaluations for convergence for SPSO with a population size of 30 and 50, respectively (at a required accuracy of 0.0001). It can be noticed that

TABLE III
RESULTS ON CONVERGENCE SPEED AT THE REQUIRED ACCURACY WITH 100% SUCCESS RATE (AVERAGED OVER 50 RUNS)

Function	Num. of global optima	ϵ	r_s	Num. of evals.(SPSO-s30) (mean and std err)	Num. of evals.(SPSO-s50) (mean and std err)
F1	5	0.0001	0.05	4116.00 $\pm$ 494.83	1134.00$\pm$ 216.76
F2	1	0.0001	0.05	930.60 $\pm$ 202.30	587.00 $\pm$ 138.13
F3	5	0.0001	0.05	4990.80 $\pm$ 478.85	1068.00 $\pm$ 175.89
F4	1	0.0001	0.05	1224.60 $\pm$ 271.76	733.00 $\pm$ 179.06
F5	4	0.0001	2.00	10135.80 $\pm$ 925.45	3987.00 $\pm$ 453.54

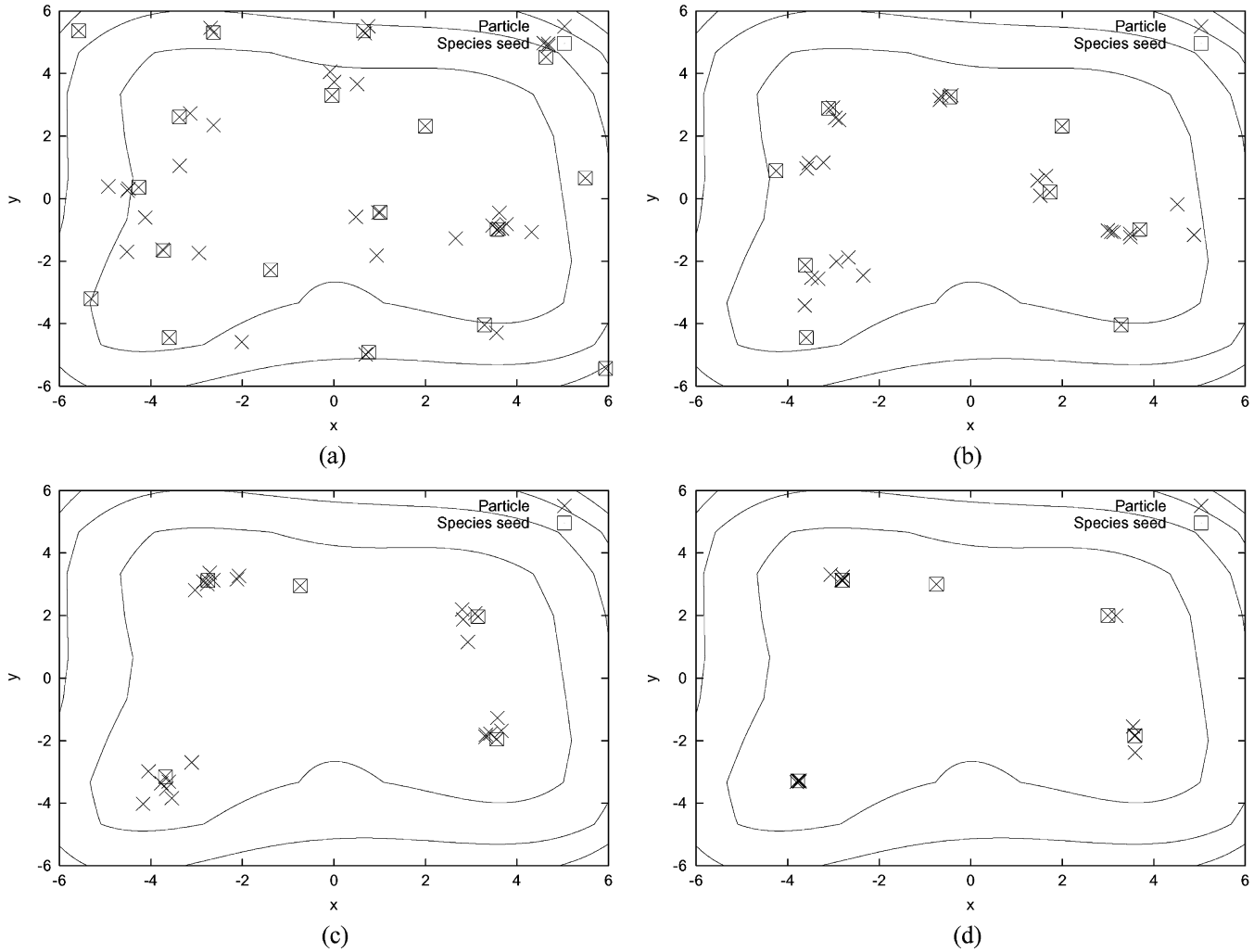

Fig. 8. Snapshot of a simulation run of SPSO on F5—Step 1, 4, 10, and 34. (a) Step 1. (b) Step 4. (c) Step 10. (d) Step 34.

a smaller number of evaluations was required for SPSO with a population size of 50.

Fig. 8 shows that on F5, in a SPSO simulation run with a population size of 50, many species seeds were identified by SPSO initially as expected. Over the successive iteration steps, these species were merged to form new species groups around the four optima. Eventually, the species seeds in four species groups reached the required accuracy of 0.0001 to the four optima at step 34.

G. Sensitivity to Population Size

Population size has a significant effect on the efficiency of SPSO. Using a small population size is not always a good idea when the task is to locate multiple optima. On the other hand, if the population size is too large, then the computation may become very expensive.

Fig. 9 shows the effect of using varying population sizes for SPSO. It can be observed that to use the smallest number of evaluations in order to reach the required accuracy 0.0001, SPSO should use a population size of around 70. We can also note that the standard error is reduced as the population size is increased to around 90.

H. Sensitivity to Niche Radius

SPSO uses the specified niche radius r_s to determine species and species seeds, therefore it is understandable that r_s plays

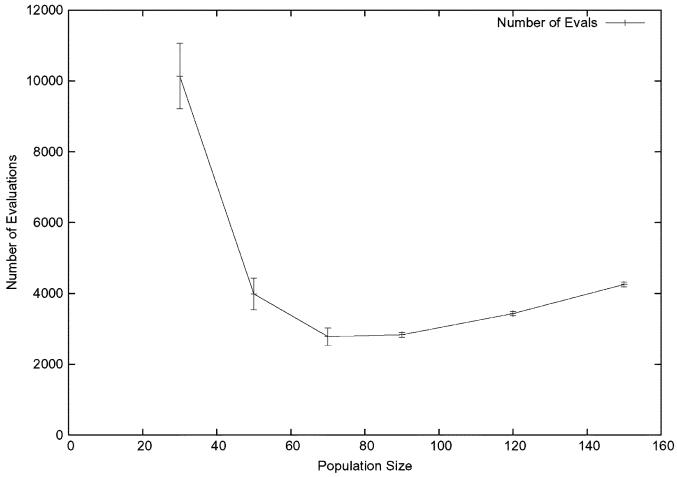

Fig. 9. Average number of evaluations in finding all known global optima (with the expected accuracy of 0.0001) over 50 runs for F5, with one standard error bar.

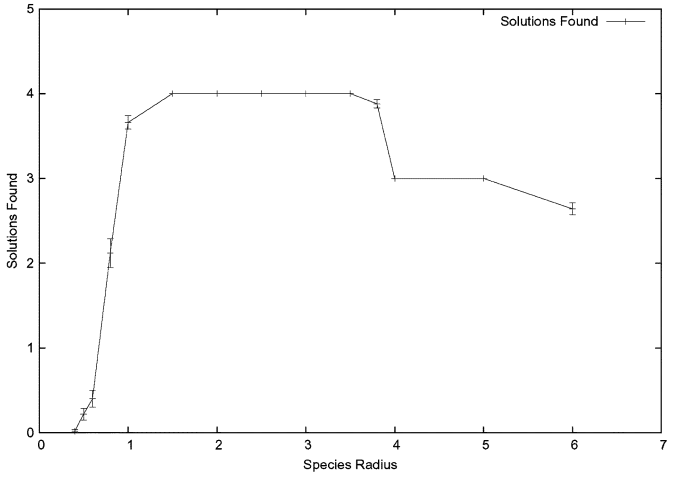

Fig. 10. Average number of global optima found over 50 runs for F5, with one standard error bar.

a critical role in the performance of SPSO. If r_s is too small, many small isolated species could be formed at each generation. A potential problem with this is that small isolated particles species tend to prematurely converge very quickly. PSO relies much on the interactions among particles in order to make further progress, so if there are not sufficient numbers of particles in each species, the species will only converge to a local optimum and become stagnant there. However, if r_s becomes too large, it is possible that SPSO will include more than one optimum within a niche radius r_s. If this happens, SPSO will have difficulty in distinguishing different optima within a species. When r_s is large enough to cover the entire variable range, SPSO degenerates to a standard PSO, which is commonly used for locating a single global optimum.

F5 was chosen to test SPSO's sensitivity to r_s. A population size of 50 was used. r_s was allowed to be varied from 0.5 to 4.0. SPSO was run for a maximum of 1000 generations. Fig. 10 shows the results. A noticeable fact is that SPSO is more sensitive to smaller r_s values than larger r_s values. When the r_s value is smaller than 1.1, the number of optima that can be found by SPSO is rapidly reduced. However, it is interesting to see that when using larger r_s values, SPSO still managed to find three or four optima. When r_s was increased to 6, although SPSO was expected to only distinguish two distant optima using the r_s value, it was able to occasionally find one additional optimum due to the fact that replacing the redundant particles by randomly generated new particles helps to better explore the search space.

I. Scalability of SPSO

F6 Rastrigin function was used to test how well SPSO performs on problems with higher dimensions and a large number of optima. Deb and Goldberg proposed a method to compute the value of niche radius, r, in a n-dimensional space where there exist p global optima [13]

$$r = \frac{\sqrt{\sum_{k=1}^{n} \left(x_k^u - x_k^l\right)^2}}{2\sqrt[n]{p}} \tag{7}$$

where x_k^l and x_k^u are the lower and upper bounds on the kth dimension of the variable vector of n dimensions. Although this method assumes that the number of global optima is known and they are evenly distributed in search space, we can use (7) as the basis to calculate the r_s values depending on the number of dimensions. Note that in (7) if p is fixed, then r gets larger as the number of dimensions increases. In the case of F6, there is only one global optimum, so the r_s values should be larger for higher dimensions.

Table IV presents the results on F6. Note that r_s is calculated according to (7), and population size for each dimensional case is simply set to be four times the number of known optima for F6 with up to five-dimensions as suggested by Brits [9]. A constant population size of 972 (the same as for dimension 5) was used for 6, 8, 9, and 10 dimensions, since the population size would be too large if it was set to be four times of the number of optima.

In this experiment, the same parameter settings were used as those by the NichePSO [9]. On this Rastrigin function, SPSO compared better than NichePSO which only achieved 100% success rate for dimension of 1 and 2, 97.45% and 97.08% for dimension of 3 and 4, respectively. It can be noted that SPSO also has increasing difficulty to converge to the required accuracy, as the dimension is increased to more than 7. This is because SPSO is essentially designed to encourage forming species depending on the local feedback in the search space. The higher dimension and the presence of a large number of local optima of the Rastrigin function would require SPSO to have an increasingly larger population in order to locate the global optimum.

J. Summary for Static Multimodal Function Optimization

By using the concept of species, we have developed a PSO which allows the swarm population to be divided into different species adaptively, depending on the feedback obtained from the fitness landscape at each iteration step during a run. Particles from each identified species follow a chosen neighborhood best to move toward a promising region of the search space. Multiple species are able to converge toward different optima in parallel, without interference across different species. In a classic

TABLE IV
SPSO'S SUCCESS RATES ON F6 RASTRIGIN FUNCTION (OVER 50 RUNS). r_s IS BASED ON (7)

Dimension	2	3	4	5	6	7	8	9	10
Number of optima	9	27	81	243	729	2187	6561	19683	59049
Population size	36	108	324	972	972	972	972	972	972
Species radius r_s	2.12	2.60	3.00	3.35	3.67	3.97	4.24	4.5	4.74
Success rate	100%	100%	100%	100%	100%	100%	94%	70%	62%

GA algorithm, crossover carried out over two randomly chosen fit individuals often produces a very poor offspring (imagining the offspring are somewhere between two fitter individuals from two distant peaks). In contrast, SPSO seems to be able to alleviate this problem effectively.

Tests on a suite of widely used multimodal test functions have shown that SPSO can find all the global optima for all test functions with one or two dimensions reliably (with 100% success rate), and with good accuracy ($< 0 \cdot 0001$), however, SPSO seemed to show increasing difficulty to converge as the dimension of the Rastrigin function was increased to more than 7. Although increasing the population size can help resolve this problem to some extent, a larger population size will incur more computational cost. Comparison of SPSO's results with other published works has demonstrated that SPSO is comparable or better than an existing evolutionary algorithms (SNGA) and a NichePSO for handling multimodal function optimization, not only with regard to accuracy and success rate, but also on convergence speed.

VI. THE DYNAMIC SPSO-DSPSO

Now that it has been shown that the SPSO can locate multiple optima, it is necessary to modify it for operation in dynamic environments. The SPSO already finds multiple optima in parallel rather than sequentially, a necessary requirement for operating in a dynamic environment, and provides a natural mechanism for particles to join and exit species and for species to join and split during a run. However, for success in a changing fitness landscape two further attributes are necessary:

1) the SPSO must be capable of tracking moving optima once located;
2) part of the population must be distributed throughout the search space in order to locate new optima or, in strongly dynamic environments, to relocate moved optima.

In order to track dynamic optima, every iteration each particle's pbest fitness value is reevaluated at its recorded pbest position before it is compared with the particle's current fitness. Particles thus use current fitness information when determining their movement, while maintaining spatial knowledge of an earlier environmental state. It is intended that this strategy will give the swarm an improved ability to track peaks relative to methods which reset the pbest location rather than reevaluate fitness values. This is a simple strategy for dealing with a dynamic environment and in a periodically or infrequently changing environment performing the extra evaluation every iteration is unnecessary. In a continuously varying environment, this extra evaluation is warranted although it doubles the number of fitness evaluations performed by the algorithm. As the extra evaluations do not alter the results obtained, it was decided to avoid the additional difficulties of implementing a scheme such as triggered updates [10], [17] needed to eliminate these evaluations.

To encourage the swarm to search the solution space and to prevent convergence of particles at known optima a maximum species population parameter $p_{\max}$ was introduced. Only the best $p_{\max}$ candidate members will be allocated as members of a species. Lower fitness candidate members that would cause the species population to exceed $p_{\max}$ are reinitialized at random positions in the solution space and are not allocated to a new species until the following iteration. 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. A drawback of this scheme is that the particles of a converged species with velocities at or near zero would not automatically increase their velocities again when optima move—they would need to be provided with new information from a particle passing within r_s to raise velocities and begin exploring the changed environment.

A. Dynamic Environment

Morrison and De Jong's DF1 dynamic test function generator [14] has been used to generate environments such as that shown in Fig. 1. This generator is capable of creating a specified number of optima in two dimensions using the equation

$$f(X,Y) = \max_{i=1,N}[H_i - R_i \cdot \sqrt{(X-X_i)^2 + (Y-Y_i)^2}] \quad (8)$$

where N gives the number of optima and the ith optimum is specified by its position (X_i, Y_i), height H_i, and slope R_i, with height and slope in the ranges

$$H_i \in [H_{\text{base}}, H_{\text{base}} + H_{\text{range}}] \quad (9)$$

$$R_i \in [R_{\text{base}}, R_{\text{base}} + R_{\text{range}}]. \quad (10)$$

Optima vary their positions, shapes, and heights. Heights cycle between a maximum and minimum value creating a sawtooth profile when height is graphed against iterations for an optimum. The fitness at a point on the surface is assigned the maximum height of all optima at that point; the optima with the greatest height at a point is then said to be visible at that point. While the number of optima present cannot be altered once the environment has been specified, the combination of changing heights and positions causes optima to temporarily obscure each other so that they vanish and reemerge.

The dynamism of the environment can be specified by a parameter A in the range $[0.0, 4.0]$, used as input to a logistics function

$$\Delta_t = A \cdot \Delta_{t-1} \cdot (1 - \Delta_{t-1}) \quad (11)$$

where the output Δ_t is in the range $(0, 1)$. Using as input a value of A less than 1.0 produces a static environment (once initial transient values have been worked through), while increasing A beyond 1.0 produces increasingly dynamic environments with a pseudorandom or chaotic series of values produced as A approaches its upper limit of 4.0. Δ is then multiplied by a scaling factor S to adapt the output to the environmental space and used as the step size for changes in the environment.

The direction of each step is randomly chosen for spatial coordinates, with steps that put a peak out of range reflecting at boundaries. The direction of change in height and slope variables is initially randomly chosen and continued in that direction until it exceeds the range, at which time the direction of change is reversed.

B. Performance Measurement

Measuring the effectiveness of an EA 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. Hence, in order to measure both the ability to track visible optima generally and the best optimum specifically, two measures of error were recorded at each iteration:

1) the best global error since last change e'_{g_t}, where e_{g_t} is the error of the fittest particle relative to the global optimum for that iteration s
$$e_{g_t} = 1 - \frac{f_{g_t}}{h_{g_t}} \tag{12}$$
with f_{g_t} the best fitness for any particle in the population and h_{gt} the height of the global optimum at time t; e'_{g_t} is the best e_{g_t} since the last change in the environment
$$e'_{g_t} = \min\{e_{g_\tau}, e_{g_{\tau+1}}, \ldots, e_{g_t}\} \tag{13}$$
where τ is the last iteration at which a change to the environment occurred;
2) the best average of best "local" errors since the last environment change across all N visible optima e'_{avg_t} calculated from e_{avg_t}
$$e_{\mathrm{avg}_t} = \frac{1}{N}\sum_{i=1}^{N} e'_{i_t} \tag{14}$$
where e'_{i_t} is calculated as per (12) and (13) from the best fitness f_{i_t} of a particle at time t on peak i of height h_{i_t} so that e'_{avg_t}
$$e'_{\mathrm{avg}_t} = \min\{e_{\mathrm{avg}_\tau}, e_{\mathrm{avg}_{\tau+1}}, \ldots, e_{\mathrm{avg}_t}\}. \tag{15}$$

From these, the offline errors $e'_{g(\mathrm{offline})}$ and $e'_{\mathrm{avg(offline)}}$ are computed, defined as per Branke [5]
$$e'_{g(\mathrm{offline})} = \frac{1}{T}\sum_{t=1}^{T} e'_{g_t} \tag{16}$$
and
$$e'_{\mathrm{avg(offline)}} = \frac{1}{T}\sum_{t=1}^{T} e'_{\mathrm{avg}_t} \tag{17}$$
where T is the number of iterations so far.

Average local error has been used in order to measure the ability of the swarm to track multiple peaks in addition to the global optimum.

For the purpose of determining f_{i_t} when calculating e_{avgt}, a particle is considered to be on an optimum if the optimum is visible at the location of a particle. If no particle is on an optimum the error for the optimum is calculated as the magnitude of the difference between the optimum's height and the fitness of the closest particle within the species radius of the optimum's apex. If there is neither a particle on an optimum nor within the species radius of the optimum's apex, that optimum is not included in the average of errors across optima e_{avgt}. Error has been assigned in this fashion due to the lack of an obvious "fair" measure of error applying to optima that barely protrude above the surrounding environment and are visible for a relatively small volume of the solution space.

C. Dynamic Environment Experimental Setup

The purpose of the experiments performed was twofold. First, it was necessary to determine empirically the effect of the population maximum $p_{\max}$ and species radius r_s on the DSPSO algorithm's performance in different environments. Second, the DSPSO's performance needs to be compared to that of existing algorithms. Random search was used to assess when the algorithms' ability to find a moving optimum ceased. The atomic swarm [2], [3] was selected as a PSO variant with good performance in dynamic environments. The atomic swarm uses an additional term in the particle swarm velocity update (1) to introduce a repulsive force between particles
$$\vec{a}_i = \sum_{j=1} \frac{\vec{Q}_i \vec{Q}_j}{r_{ij}^3}\vec{r}_{ij}, \quad r_{\mathrm{core}} < r_{ij} < r \tag{18}$$
and (1) becomes
$$v_{\mathrm{id}}(t) = \chi(v_{\mathrm{id}}(t-1) + \varphi_1 r_1(p_{\mathrm{id}} - x_{\mathrm{id}}(t-1)) + \varphi_2 r_2(p_{gd} - x_{\mathrm{id}}(t-1))) + a_{\mathrm{id}} \tag{19}$$
where Q_i is the charge of particle i, $\vec{r}_{ij} = \vec{x}_i - \vec{x}_j, r_{ij} = |\vec{r}_{ij}|$ and r and r_{core} are the upper and lower limits of distance between particles for repulsion to occur. Note that as the atomic swarm algorithm is designed to track only a single peak rather than multiple peaks, simultaneously it is unfair to compare its performance with the DSPSO algorithm on the e'_{avg} results; these results have been included for the atomic swarm algorithm for completeness only.

The experiments were run in a 2-D environment with range $[-1.0, 1.0]$ for 1000 iterations. Peaks' initial positions were assigned randomly. Output from the logistics function was scaled by a factor of 0.4 to reduce the step sizes and dynamism controlled by assigning A a value in $\{1.2, 3.3, 3.9\}$, giving, respectively, a constant step size of 0.067, a repeating series of step sizes of 0.329 and 0.192 and a chaotic series of step sizes. It should be noted that although the average output of the logistic function for $A = 3.3$ (approximately 0.65) is greater than the average output for $A = 3.9\,(0.53)$, while the maximum possible output arising from $A = 3.99\,(0.998)$ is greater than that when $A = 3.3\,(0.825)$. It is therefore not possible to unequivocally state whether the dynamism for $A = 3.99$ should be considered less than or greater than that for $A = 3.3$ without a formal definition of "dynamism." The height and shape parameters of the

TABLE V
EXPERIMENTAL PARAMETER VALUES

Experiment	A	No. of particles	No. of Optima	Update Interval	p_{max}	r_s
1	{1.2,3.3,3.99}	60	3	10	{2,5,10,20,40,60}	0.1
2	{1.2,3.3,3.99}	60	3	10	15	{0.1,0.2,0.5,1.0}
3	1.2	60	3	10	{2,5,10,20,40,60}	{0.1,0.2,0.5,1.0}
4	{1.2,3.3,3.99}	60	3	{1,10,100}	15	0.1
5	1.2	60	{1,3,10,100}	10	15	0.1

peaks were limited to $H_i \in [1 \cdot 0; 5 \cdot 0]$ and $R_i \in [1 \cdot 0; 5 \cdot 0]$ with initial values of H_i and R_i evenly distributed throughout the range of allowed values.

PSO parameters were set in line with Clerc's values for constriction [12]-φ_1 and φ_2 were set to 2.05 and the constriction factor χ to 0.729 844. Velocity was not constrained. The population size was set to 60 to allow for tracking multiple peaks. Electrostatic parameters for the atomic swarm were otherwise set to be analogous to those given by [3] scaled for the difference in environment dimensions. Hence, on 1/2 the particles the charge Q was set to 0.5 (1/4 of the side length of the environment as per [3]) and to 0.0 on the remaining particles. The upper limit of distance for repulsion r is set to $2\sqrt{2}$ to include all of the environment and the lower cutoff r_{core} set to 0.03 (approximately 1/64th of the side length as per [3]).

Five experiments were performed as follows.

1) To test the effect of the population maximum $p_{\max}$ on the ability of the DSPSO to track moving peaks the DSPSO algorithm was run with values of $p_{\max}$ in $\{2, 5, 10, 20, 40\ \ 60\}$ were combined with A in $\{1.2, 3.3, 3.9\}$. The total population was set at 60 particles and the species radius r_s set to 0.1. The environment consisted of three peaks updating every ten iterations.
2) To test the effect of the species radius r_s on the ability of the DSPSO to track moving peaks the DSPSO algorithm was run with values of r_s in $\{0.1, 0.2, 0.5, 1.0\}$ were combined with A in $\{1.2, 3.3, 3.9\}$. The total population was set at 60 particles and the population max $p_{\max}$ set at 15. The environment consisted of three peaks updating every ten iterations.
3) The relationship between $p_{\max}$ and species radius r_s was investigated by assigning $p_{\max}$ and r_s the values from experiments 1 and 2. A population of 60 was used with an environment of three peaks updating every ten iterations and A set to 1.2.
4) The ability of the DSPSO relative to the atomic swarm and random search to cope with different levels of dynamism was tested by running the three algorithms in environments with A in $\{1.2, 3.3, 3.99\}$ and the update interval in $\{1, 10, 100\}$. DSPSO parameters $p_{\max}$ and r_s were set to 15 and 0.1, respectively. A population of 60 particles was used.
5) The ability of DSPSO relative to atomic swarms and random search to cope with large numbers of peaks was investigated by using environments with 1, 3, 10, or 100 peaks and A set to 1.2. DSPSO parameters were those used for Experiment 3.

The different parameter combinations for the experiments are summarized in Table V. Thirty runs were performed for each combination of settings and the results averaged over the runs. Optima and particles were assigned random initial positions for each run and initial optima heights evenly distributed between the minimum and maximum optima heights in H_i.

D. Results

1) Varying Species Population Maximum $p_{\max}$ and A: Results are provided in Fig. 11. Examining the results shows that a lower population maximum gives better results for greater levels of dynamism on the environments investigated, while the average local error e'_{avg} and global error e'_g obtained with $p_{\max}$ set to the total particle population is greater than any error obtained with $p_{\max}$ set to fewer particles. Average error is at a minimum with $p_{\max} = 10$ for $A = 1.2$ and $p_{\max} = 2$ for $A = 3.99$ and $A = 3.3$. Minimum global errors e'_g are obtained with $p_{\max}$ values greater than those which provide the lowest e'_{avg} errors for the same A values.

2) Varying Species Radius r_s and A: Results are provided in Fig. 12. The global error e'_g is shown to be fairly insensitive to the r_s setting on the investigated environment with slight improvements for the high-dynamism cases at $r_s = 0.5$ and at the low-dynamism case for $r_s = 0.2$. Average error is at a minimum with $r_s = 0.2$ for all A investigated.

3) Varying Species Population Maximum $p_{\max}$ and Speciation Radius r_s: As shown in Fig. 13, the global error e'_g remains insensitive to r_s across the range of $p_{\max}$ values except at the extremes, where a large r_s value of 1.0 gives best performance for $p_{\max} = 60$ and a small r_s value of 0.1 gives best performance for $p_{\max} = 2$. Performance is best for $p_{\max} = 20$ across the range of r_s values. Average errors across peaks are generally less for the $r_s = 0.1$ and $r_s = 0.2$ cases.

Performance on e'_{avg} suffers as r_s increases and as $p_{\max}$ moves away from a value of 10.

4) Comparison of DSPSO, Atomic Swarm, and Random Search at Different Levels of Dynamism: Results, averaged first over the iterations of each run then averaged across the 30 runs, are shown in Tables VI–VIII for values of A of 1.2, 3.3 and 3.99, respectively. Initial optima and particle locations for each run are randomly generated, while optima heights are evenly distributed between the minimum and maximum allowable heights. Snapshots of the particles of the DSPSO algorithm for a sequence of iterations are shown in Fig. 14 for $A = 3.99$ with an update interval of ten iterations. The case shown features two visible optima with the third optima obscured. The particles can be seen to congregate around the

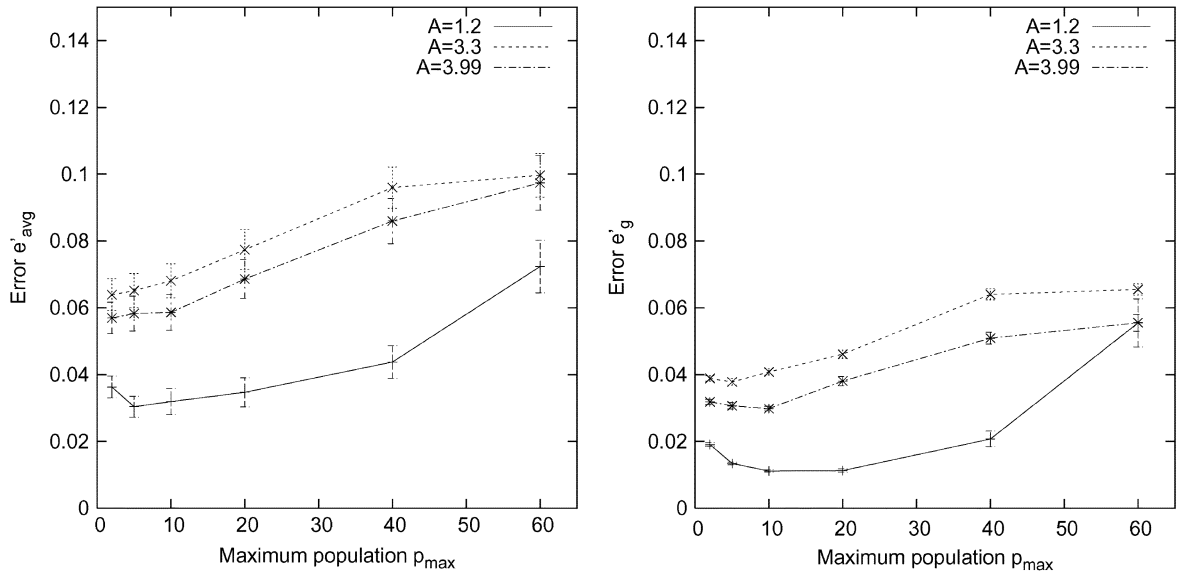

Fig. 11. Average minimum error e'_{avg} (left) and global error e'_g (right) across optima versus $p_{\max}$ with varying dynamism. The error bar represents one standard error.

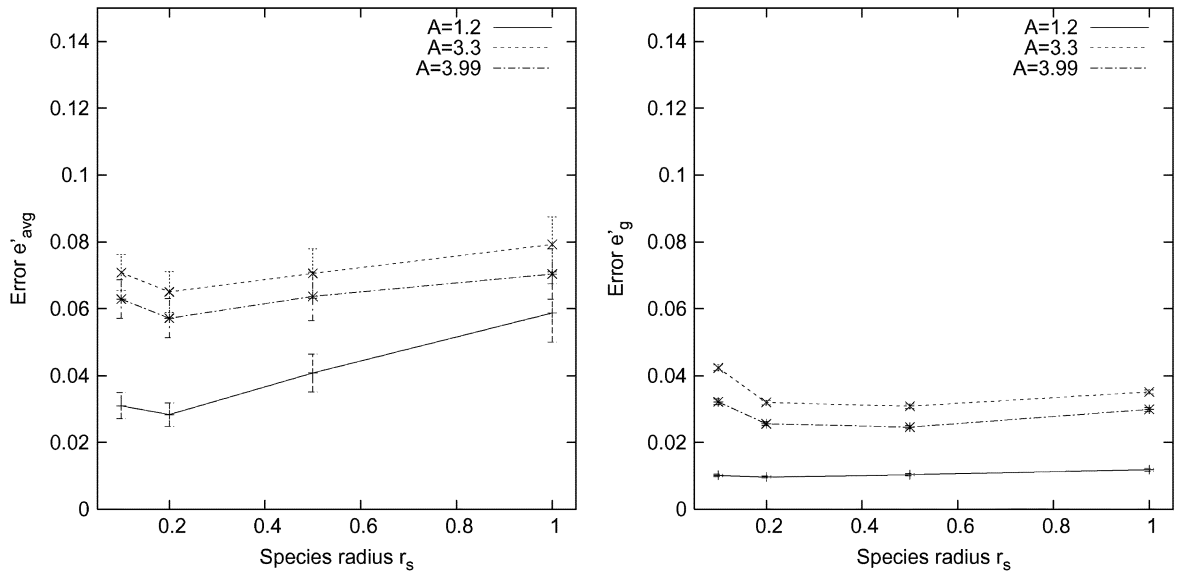

Fig. 12. Average minimum error e'_{avg} (left) and global error e'_g (right) across optima versus r_s with varying dynamism. The error bar represents one standard error.

visible optima with a number of particles distributed over the remaining solution space. The transition from Fig. 14(c) to (d) shows the peaks moving and losing their attendant swarms. Figs. 14(e) and (f) show the swarms relocating and exploiting the two peaks.

Graphs of error $e'_{g(\text{offline})}$ versus iteration for the three algorithms and $e'_{\text{avg}(\text{offline})}$ versus iteration for DSPSO and random search are given in Figs. 15 and 16. These show that, in the runs with an update interval of 100 [Figs. 15(c) and 16(c)], the global error $e'_{g(\text{offline})}$ of the atomic swarm and DSPSO algorithms are similar before the first environmental update but the error of the atomic swarm algorithm increases as the runs progress before stabilizing. A similar relationship can be seen between the DSPSO algorithm and random search for the cases with $A = 3.99$ and updates every interval [Fig. 16(a)]—$e'_{g(\text{offline})}$ is similar for the first 100 iterations before that for DSPSO increases, and then stabilizes between iteration 200 and 400.

Broadly, results are as expected—as A was increased the DSPSO and atomic swarm algorithms found it increasingly difficult to track the global optima; increasing the update interval results in better tracking; the offline error e_{offline} decreases initially then stabilizes. Random search results decrease as the update interval is increased and stay approximately constant as peak motion is increased by increasing A.

In all cases, DSPSO outperforms the atomic swarm algorithm. Random search only outperforms DSPSO in the highest dynamism cases with an update interval of 1 and $A = \{3.3, 3.99\}$ and also outperforms atomic swarm for an update interval of 10 and $A = \{3.3, 3.99\}$ and performs as well as atomic swarm with an interval of 1 and $A = 1.2$.

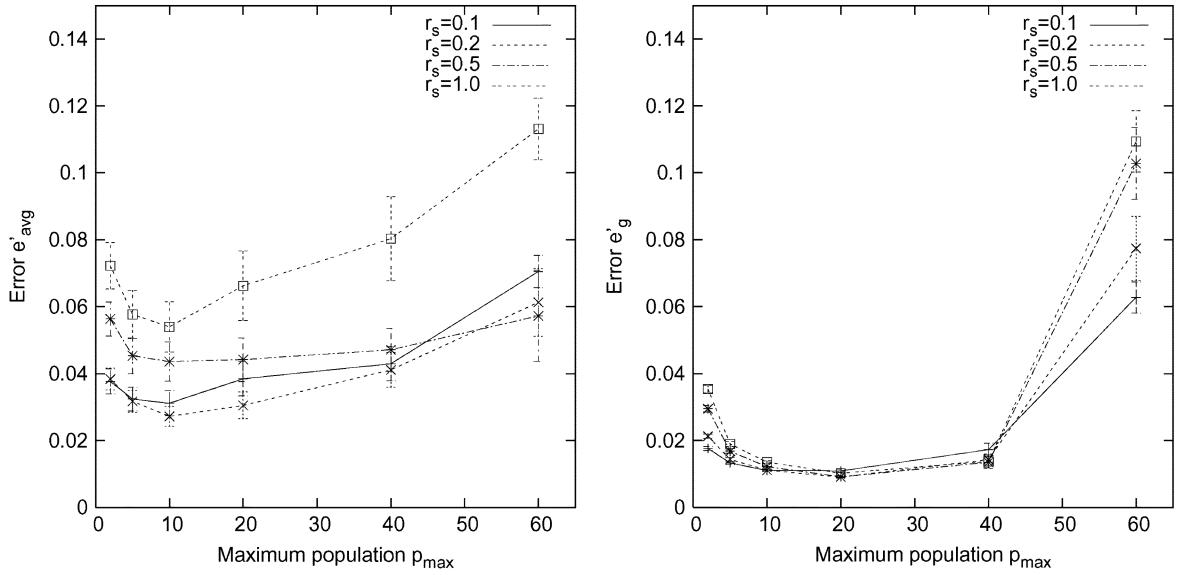

Fig. 13. Average local error e'_{avg} (left) and global error e'_g (right) across optima versus $p_{\max}$ with varying species radius r_s. The error bar represents one standard error.

TABLE VI
COMPARISON OF ALGORITHMS WITH $A = 1.2$

Update interval	Algorithm	Global error e'_g	SE of e'_g	Average local error e'_{avg}	SE of e'_{avg}
1	DSPSO	**0.058**	0.0016	**0.079**	0.0042
	Atomic swarm	0.091	0.0076	0.167	0.0159
	Random search	0.091	0.0033	0.118	0.0059
10	DSPSO	**0.011**	0.0004	**0.030**	0.0036
	Atomic swarm	0.035	0.0051	0.131	0.0186
	Random search	0.045	0.0014	0.062	0.0035
100	DSPSO	**0.004**	0.0004	**0.013**	0.0017
	Atomic swarm	0.008	0.0007	0.060	0.0096
	Random search	0.016	0.0010	0.023	0.0015

TABLE VII
COMPARISON OF ALGORITHMS WITH $A = 3.3$

Update interval	Algorithm	Global error e'_g	SE of e'_g	Average local error e'_{avg}	SE of e'_{avg}
1	DSPSO	0.124	0.0024	0.153	0.0061
	Atomic swarm	0.141	0.0028	0.191	0.0097
	Random search	**0.097**	0.0020	**0.124**	0.0053
10	DSPSO	**0.044**	0.0010	0.074	0.0057
	Atomic swarm	0.059	0.0047	0.124	0.0129
	Random search	0.050	0.0039	**0.066**	0.0031
100	DSPSO	**0.008**	0.0005	**0.022**	0.0026
	Atomic swarm	0.016	0.0020	0.054	0.0073
	Random search	0.018	0.0005	0.025	0.0014

As feared, converged DSPSO species would not explore the solution space until provided with new data by a passing particle but, as this happened rapidly, this did not pose a significant problem for the dynamic environment used.

5) Comparison of Algorithms With the Number of Optima Varying: As shown in Table IX, the e'_g and e'_{avg} errors increase for both the DSPSO and atomic swarm as the number of peaks increases (with the exception of the change from 10 to 100 peaks for atomic swarm on e'_{avg}). The DSPSO performs as well as the atomic swarm where a single peak is used and outperforms both the atomic swarm and random search in all other cases investigated. Random search outperforms atomic search for 100

TABLE VIII
COMPARISON OF ALGORITHMS WITH $A = 3.99$

Update interval	Algorithm	Global error e'_g	SE of e'_g	Average local error e'_{avg}	SE of e'_{avg}
1	DSPSO	0.111	0.0017	0.153	0.0080
	Atomic swarm	0.132	0.0049	0.199	0.0134
	Random search	**0.090**	0.0013	**0.125**	0.0066
10	DSPSO	**0.034**	0.0010	**0.062**	0.0054
	Atomic swarm	0.051	0.0039	0.119	0.0133
	Random search	0.047	0.0005	**0.064**	0.0032
100	DSPSO	**0.006**	0.0005	**0.018**	0.0022
	Atomic swarm	0.010	0.0009	0.055	0.0084
	Random search	0.018	0.0007	0.027	0.0018

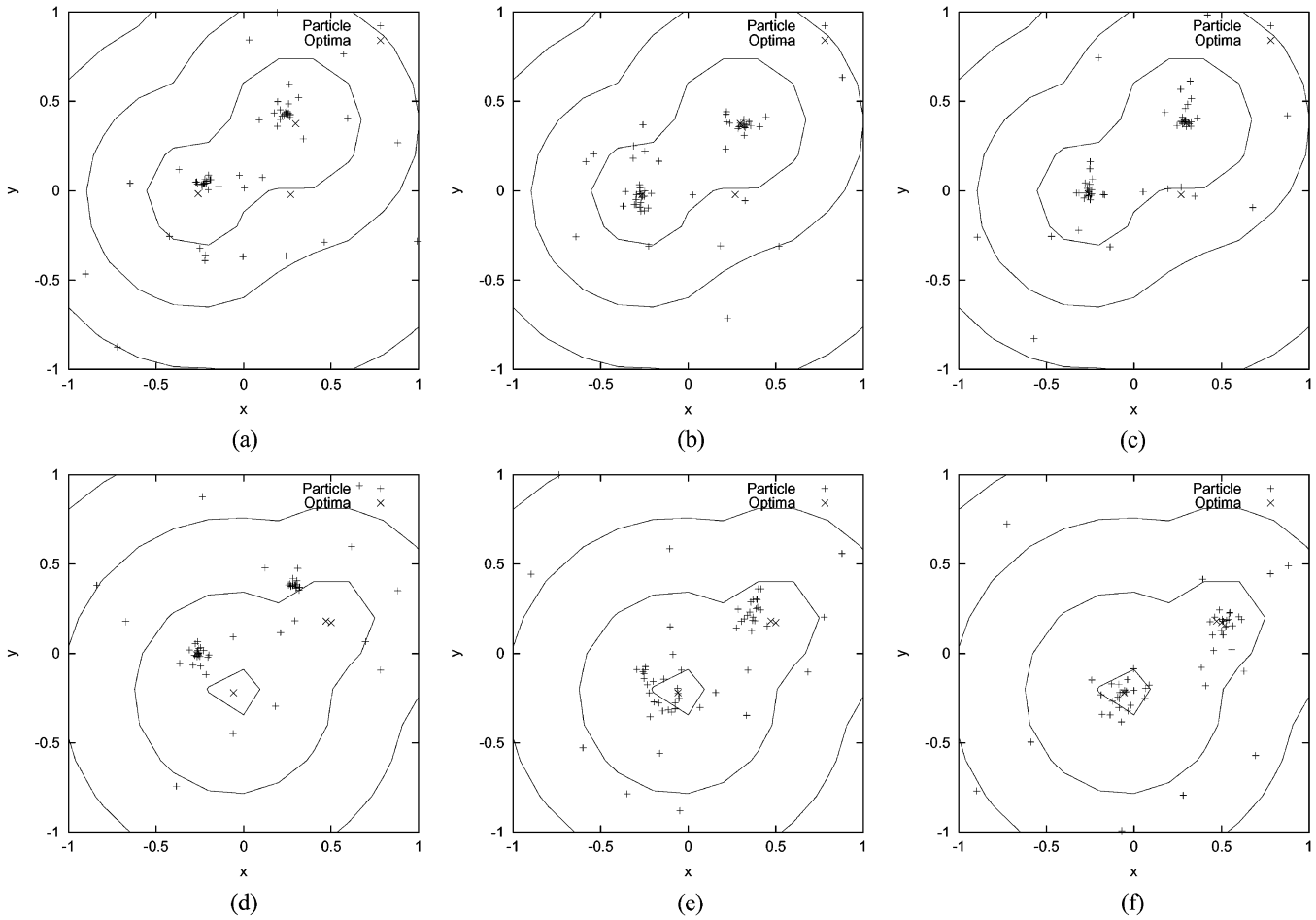

Fig. 14. Snapshots from a run with three optima (note that one is obscured), $A = 3.99$ and an update interval of ten iterations. (a) Iteration 340, immediately after environment update. (b) Iteration 342, two iterations after environment update. (c) Iteration 349, semiconverged ten iterations after last environment update. (d) Iteration 350, immediately after environment update. (e) Iteration 352, two iterations after environment update. (f) Iteration 359, semiconverged ten iterations after last environment update.

peaks and is close for 10 peaks. For the 100 peak instance the random search has a lower e'_{avg} error than the DSPSO.

E. Discussion of the Results for the DSPSO

The results indicate that the proposed DSPSO algorithm can successfully track optima in a dynamic 2-D environment. The DSPSO outperforms the atomic swarm algorithm in the dynamic multimodal environments tested. An atomic swarm is capable of tracking only one peak at a time and must change optima rapidly when the global optimum changes. The DSPSO is likely to already have a species exploiting the new global optimum giving it a distinct advantage, where the global optimum frequently switches optima as in the environments tested.

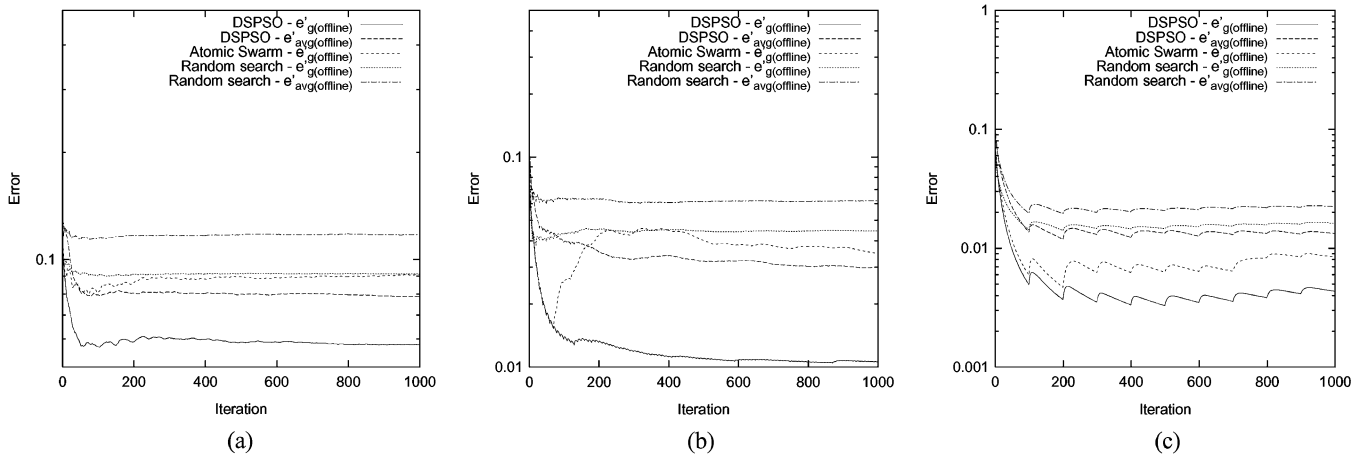

Fig. 15. Offline errors $e'_{g(\mathrm{offline})}$ and $e'_{\mathrm{avg}(\mathrm{offline})}$ versus iteration $\mathrm{A} = 1.2$, updates every (a) 1, (b) 10, and (c) 100 iterations.

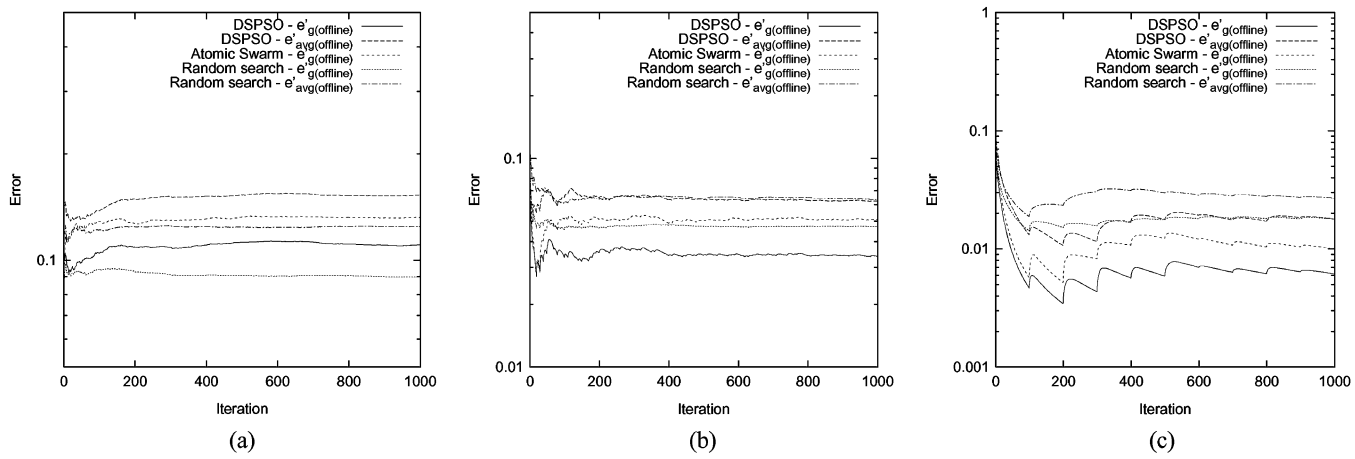

Fig. 16. Offline errors $e'_{g(\mathrm{offline})}$ and $e'_{\mathrm{avg}(\mathrm{offline})}$ verus iteration $A = 3.99$, updates every (a) 1, (b) 10, and (c) 100 iterations.

The graphs of runs with environment updates every 100 iterations [Figs. 15(c) and 16(c)] demonstrate this, showing the atomic swarm performing similarly to the DSPSO before the first update but worse after. The DSPSO and atomic swarm achieved very similar e'_g values, where the environment featured a single peak which shows that, in some environments at least, the DSPSO does not gain its ability to track multiple optima by sacrificing the ability to track a single optimum.

Only at the greatest levels of dynamism tested, $A = \{3.3, 3.99\}$ and with the environment changing every iteration, does random search outperform the DSPSO. Intuitively, as the level of dynamism increases, the value of information describing the current environmental state for describing the new environmental state will approach zero, and hence at some level of dynamism random search will become the best-performing technique.

The results show how $p_{\max}$ can affect the swarm's searching and tracking abilities. A number of conclusions can be drawn from the data obtained. The smaller errors achieved with small $p_{\max}$ in high-dynamism cases indicate that these $p_{\max}$ settings provide better search abilities from more particles being reinitialized each iteration. However, the increase in error for the $A = 1.2$ instance as $p_{\max}$ decreases below ten shows that the smaller species resulting from low $p_{\max}$ are less successful at tracking optima due to their small size. In high-dynamism environments a lower $p_{\max}$ value resulted in lower errors. Comparing the effect of changing $p_{\max}$ values observed in environments updating every ten iterations with those observed in earlier work on the DSPSO [27], which examined the algorithm in a similar environment updating every iteration, also shows that larger $p_{\max}$ values give lower errors when the update interval is increased.

This improvement is due to the reduced need for exploration when optima move less frequently and the improved ability of larger species to converge on optima. The results show that the $p_{\max}$ parameter can be used to balance the tradeoff between tracking and search abilities. Generally, high-dynamism environments favor relatively small species populations which encourage exploration without being so small that species lose the ability to track individual optima effectively. Low-dynamism environments favor relatively large species to track a single optimum well but not so large that the total swarm will not search for or track multiple optima effectively. In practice, some knowledge of the environment at hand will be necessary to find the middle ground and set the parameter effectively.

The insensitivity to changes in r_s of the DSPSO algorithm's ability to find the global optimum is surprising. When the average local error e'_{avg} is considered as in Fig. 13 the

TABLE IX
COMPARISON OF ALGORITHMS WITH VARYING NUMBER OF PEAKS; $A = 1.2$ AND UPDATE INTERVAL $= 10$

Number of peaks	Algorithm	Global error e'_g	SE of e'_g	Average local error e'_{avg}	SE of e'_{avg}
1	DSPSO	**0.017**	0.0006	**0.017**	0.0006
	Atomic swarm	**0.017**	0.0006	**0.017**	0.0006
	Random search	0.088	0.0027	0.088	0.0027
3	DSPSO	**0.011**	0.0003	**0.032**	0.0041
	Atomic swarm	0.026	0.0006	0.124	0.0185
	Random search	0.045	0.0016	0.061	0.0034
10	DSPSO	**0.017**	0.0006	**0.045**	0.0053
	Atomic swarm	0.035	0.0013	0.126	0.0169
	Random search	0.039	0.0010	0.051	0.0025
100	DSPSO	**0.025**	0.0004	0.047	0.0042
	Atomic swarm	0.033	0.0004	0.080	0.0087
	Random search	0.028	0.0003	**0.039**	0.0022

$r_s = \{0.1, 0.2\}$ settings give better results than $r_s = \{0.5, 1.0\}$, while the global error e'_g is very similar. The difference in e'_{avg} is caused by the poor resolution of the DSPSO with large species radii—only one peak can be properly resolved and tracked within the species radius; other optima are ignored. That the global error e'_g does not suffer as a result of poor local optima tracking indicates that not just the number of optima covered is important; the space covered by all species also plays a role.

VII. CONCLUSION AND FUTURE WORK

This paper has given details of the SPSO algorithm, a particle swarm model which uses spatial speciation for locating multiple local optima in parallel, and adaptations to it to create the DSPSO algorithm for tracking multiple peaks in a dynamic environment. The results show that the speciation mechanism implemented with a r_s parameter allows the SPSO to locate multiple optima in parallel. The addition of a crowding mechanism implemented using $p_{\max}$ together with a method for updating the swarm's knowledge of the environment gives the swarm the ability to track multiple dynamic optima and adjust the balance of exploration and exploitation in a 2-D environment.

There are many areas for potential research. In the future, we will apply SPSO to large and more complex real-world multimodal optimization problems, especially problems where we have little (or no) prior knowledge about the search space. We also need to investigate how to best choose the species radius, for example, perhaps looking at how to adaptively choose the species radius based on the feedback obtained during the search.

The method used to update particle knowledge in the dynamic environment is simple and as noted earlier computationally inefficient in an infrequently updating environment. This could be improved by applying triggered information updates. Also, the 2-D environment used to test the DSPSO algorithm was simple. Further tests in more challenging environments are necessary, for example, one with more than two dimensions. Examining behavior in an environment with a small footprint, steep-sided global optimum, and large footprint, gently sloping local optima is also necessary to test the phenomena of particles disproportionately clustering around the global optima and of converged species remaining static when the environment changes until new data are provided. More research could look at self-adaptation of the user-defined $p_{\max}$ and r_s parameters to adjust to environments where the dynamism itself changes with time. Combining the DSPSO with algorithms such as the atomic swarm is also worth investigating–repulsion could be used within species to better track moving optima.

An improved and expanded set of metrics is needed to analyze behavior over the relevant objectives of the algorithm tracking the global optimum, tracking multiple local optima, and locating optima. As the ability to achieve each objective is improved by having more particles dedicated to it these objectives are in competition with each other. Tuning the algorithm's performance requires examining the objectives simultaneously and so that a beneficial balance can be found.

This paper has demonstrated that the SPSO and DSPSO algorithms are effective in some environments. Although this paper does not fully explore the possibilities of the algorithms, it is hoped that it may contribute to others' efforts to further research into evolutionary optimization in dynamic, multimodal environments.

ACKNOWLEDGMENT

The authors would like to especially thank J. Branke, Y. Jin, and the anonymous reviewers for comments that have greatly improved this paper.

REFERENCES

[1] D. Beasley, D. R. Bull, and R. R. Martin, "A sequential niche technique for multimodal function optimization," *Evol. Comput.*, vol. 1, no. 2, pp. 101–125, 1993.

[2] T. M. Blackwell, "Swarms in dynamic environments," in *Lecture Notes in Computer Science* Chicago, IL, 2003, vol. 2723, Proc. Genetic Evol. Comput. Conf., pp. 1–12.

[3] T. M. Blackwell and P. J. Bentley, "Dynamic search with charged swarms," in *Proc. Genetic Evol. Comput. Conf.*, New York, 2002, pp. 19–26.

[4] T. M. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," in *Proc. Appl. Evol. Comput.: EvoWorkshops 2004: EvoBIO, EvoCOMNET, EvoHOT, EvoISAP, EvoMUSART, and EvoSTOC*, vol. 3005, LNCS, Coimbra, Portugal, 2004, pp. 489–500.
[5] J. Branke, *Evolutionary Optimization in Dynamic Environments*. Norwell, MA: Kluwer , 2002.
[6] J. Branke, T. Kaubler, C. Schmidt, and H. Schmeck, "A multi-population approach to dynamic optimization problems," in *Adaptive Computing in Design and Manufacture 2000*. Berlin, Germany: Springer-Verlag, 2000, pp. 299–308.
[7] R. Brits, A. P. Engelbrecht, and F. van den Bergh, "A niching particle swarm optimizer," in *Proc. 4th Asia-Pacific Conf. Simulated Evol. Learning*, 2002, pp. 692–696.
[8] ——, "Solving systems of unconstrained equations using particle swarm optimization," in *Proc. IEEE Conf. Syst., Man, Cybern.*, Hammamet, Tunisa, 2002, pp. 102–107.
[9] ——, "Scalability of niche PSO," in *Proc. IEEE Swarm Intell. Symp.*, Indianapolis, IN, 2003, pp. 228–234.
[10] A. Carlisle and G. Dozier, "Adapting particle swarm optimization to dynamic environments," in *Proc. Int. Conf. Artif. Intell.*, Las Vegas, NV, 2000, pp. 429–434.
[11] M. Clerc, "The swarm and the queenL toward a deterministic and adaptive particle swarm optimization," in *Proc. Congr. Evol. Comput.*, 1999, pp. 1951–1957.
[12] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.
[13] K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proc. 3rd Int. Conf. Genetic Algorithms*, J. D. Schaffer, Ed.. San Mateo, CA, 1989, pp. 42–50.
[14] K. A. De Jong and R. W. Morrison, "A test problem generator for nonstationary environments," in *Proc. Congr. Evol. Comput.*, 1999, pp. 2047–2053.
[15] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proc. Congr. Evol. Comput.*, 2001, pp. 94–100.
[16] D. Fasulo, "An analysis of recent work on clustering algorithms," Univ. Washington, Seattle, WA, Tech. Rep. UW-CSE01-03-02, 1999.
[17] X. Hu and R. C. Eberhart, "Adaptive particle swarm optimization: Detection and response to dynamic systems," in *Proc. IEEE Congr. Evol. Comput.*, 2002, pp. 1666–1670.
[18] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments—A survey," *IEEE Trans. Evol. Comput.*, vol. 9, no. 3, pp. 303–317, Jun. 2005.
[19] J. Kennedy, "Stereotyping: Improving particle swarm performance with cluster analysis," in *Proc. Congr. Evol. Comput.*, 2000, pp. 1507–1512.
[20] J. Kennedy and R. Eberhart, *Swarm Intelligence*. San Mateo, CA: Morgan Kaufmann, 2001.
[21] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw. IV*, 1995, pp. 1942–9148.
[22] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proc. Congr. Evol. Comput.*, 2002, pp. 1671–1675.
[23] J. Kennedy and W. M. Spears, "Matching algorithms to problems: An experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator," in *Proc. IEEE Int. Conf. Evol. Comput.*, 1998, pp. 78–83.
[24] J. P. Li, M. E. Balazs, G. Parks, and P. J. Clarkson, "A species conserving genetic algorithm for multimodal function optimization," *Evol. Comput.*, vol. 10, no. 3, pp. 207–234, 2002.
[25] X. Li, "Adaptively choosing neighborhood bests using species in a particle swarm optimizer for multimodal function optimization," in *Proc. Genetic Evol. Comput. Conf.*, K. Deb *et al.*, Eds., 2004, pp. 105–116. Lecture Notes in Computer Science (LNCS 3102).
[26] R. W. Morrison, *Designing Evolutionary Algorithms for Dynamic Environments*. Berlin, Germany: Springer-Verlag, 2004.
[27] D. Parrott and X. Li, "A particle swarm model for tacking multiple peaks in a dynamic environment using speciation," in *Proc. Genetic Evol. Comput. Conf.*, 2004, pp. 98–103.
[28] K. E. Parsopoulos, V. P. Plagianakos, G. D. Magoulas, and M. D. Vrahatis, "Stretchingtechnique for obtaining global minimizers through particle swarm optimization," in *Proc. Particle Swarm Optimization Workshop*, Indianapolis, IN, 2001, pp. 22–29.
[29] K. E. Parsopoulos and M. N. Vrahatis, "Modification of the particle swarm optimizer for locating all the global minima," in *Proc. Int. Conf. Artif. Neural Netw. Genetic Algorithms*, Prague, Czech Republic, 2001, pp. 324–327.
[30] ——, "On the computation of all global minimizers through particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 211–224, Jun. 2004.
[31] A. Petrowski, "A clearing procedure as a niching method for genetic algorithms," in *Proc. 1996 IEEE Int. Conf. Evol. Comput.*, 1996, pp. 798–803.
[32] R. Thomsen, "Multimodal optimization using crowding-based differential evolution," in *Proc. 2004 Congr. Evol. Comput.*, vol. 2, 2004, pp. 1382–1389.
[33] R. K. Ursem, "Multinational GAs: Multimodal optimization techniques in dynamic environments," in *Proc. 2nd Genetic Evol. Comput. Conf.*, 2000, pp. 19–26.
[34] F. van den Bergh, "An analysis of particle swarm optimizers," Ph.D. dissertation, Dept. Comput. Sci., Univ. Pretoria, Pretoria, Gauteng, South Africa, 2002.
[35] J. S. Vesterstroem and J. Riget, "Particle swarms: Extensions for improved local, multi-modal, and dynamic search in numerical optimization," Master's thesis, Dept. Comput. Sci., University of Aarhus, Aarhus, Denmark, 2002.

Daniel Parrott received the B.E. (Honors) and Bachelor of Commerce degrees from Monash University, Victoria, Australia, and the M.S. degree in information technology from RMIT University, Melbourne, Australia.

He has been working in the Evolutionary Computation Group at RMIT University since 2003 and is interested in particle swarm optimization and genetic programming in particular.

Xiaodong Li (M'03) received the B.Sci. degree in information science from Xi'an University of Electronic Technology, Xi'an, China, in 1988, and the Dip.Com. and Ph.D. degrees in information science from the University of Otago, Dunedin, New Zealand, in 1992 and 1998, respectively.

He is currently with the School of Computer Science and Information Technology, RMIT University, Melbourne, Australia. His research interests include evolutionary computation (in particular, evolutionary multiobjective optimization and evolutionary optimization in dynamic environments), neural networks, complex systems, and swarm intelligence.

Dr. Li is a technical committee member of the IEEE Working Group on Swarm Intelligence, part of the Evolutionary Computation Technical Committee of the IEEE Computational Intelligence Society. He is also a member of SIGEVO (The ACM Special Interest Group on Genetic and Evolutionary Computation). He was the chief organizing chair for two special sessions on Swarm Intelligence, held as part of Congress on Evolutionary Computation 2003 and 2004.