Multiswarms, Exclusion, and Anti-Convergence in Dynamic Environments

Tim Blackwell and Jürgen Branke, *Member, IEEE*

***Abstract*—Many real-world problems are dynamic, requiring an optimization algorithm which is able to continuously track a changing optimum over time. In this paper, we explore new variants of particle swarm optimization (PSO) specifically designed to work well in dynamic environments. The main idea is to split the population of particles into a set of interacting swarms. These swarms interact locally by an exclusion parameter and globally through a new anti-convergence operator. In addition, each swarm maintains diversity either by using charged or quantum particles. This paper derives guidelines for setting the involved parameters and evaluates the multiswarm algorithms on a variety of instances of the multimodal dynamic moving peaks benchmark. Results are also compared with other PSO and evolutionary algorithm approaches from the literature, showing that the new multiswarm optimizer significantly outperforms previous approaches.**

***Index Terms*—Dynamic environments, genetic algorithms, optimization methods, particle swarm optimization (PSO).**

I. Introduction

PARTICLE SWARM OPTIMIZATION (PSO) is now established as an efficient optimization algorithm for static functions in a variety of contexts [35]. PSO is a population-based technique, similar in some respects to evolutionary algorithms (EAs), except that potential solutions (particles) move, rather than evolve, through the search space. The rules, or particle dynamics, which govern this movement, are inspired by models of swarming and flocking [27]. Each particle has a position and a velocity, and experiences linear spring-like [9] attractions toward two attractors:

1) the best position attained by that particle so far (particle attractor);
2) the best of the particle attractors in a certain neighborhood (neighborhood attractor);

where best is in relation to evaluation of an objective function at that position. The swarm attractor therefore enables information sharing between particles, while the particle attractors serve as individual particle memories.

The optimization process is iterative. In each iteration, the acceleration vectors of all the particles are calculated based on the position of the corresponding attractors. Then, this acceleration is added to the velocity vector, the updated velocity is constricted so that the particles progressively slow down, and this new velocity is used to move the individual from the current to the new position. A more detailed introduction to PSO is provided in Section II-A.

While most of the optimization problems discussed in the scientific literature are static, many real-world problems are dynamic, i.e., they change over time. In such cases, the optimization algorithm has to track a moving optimum as closely as possible, rather than just find a single good solution. It has been argued [13] that EAs may be a particularly suitable candidate for this type of problems, and over the past decade, a large number of EA variants for dynamic optimization problems have been proposed. For an overview, the reader is referred to [12]–[14], [25], and [32].

Recently, the application of PSO to dynamic problems has also been explored [1], [2], [16], [18], [21], [23], [24], [31], [35], [39]. Similar to EAs, PSO needs to be adapted for optimal results on dynamic optimization problems. This is due to the following reasons.

1) *Outdated memory*: When the problem changes, the information stored in the memory, i.e., each individual's local best solution and the corresponding fitness, may no longer be true and may actually be misguiding the search.
2) *Diversity loss and linear collapse*: If the swarm is converging, the attractors will be close to the optimum position and the swarm will be shrinking at a rate determined by the constriction factor and by the local environment at the optimum. For functions with spherical symmetric local neighborhoods, a theoretical analysis and an experimental verification suggest that the rate of shrinkage (and hence diversity loss) is scale invariant [3], [4], [8]. If the optimum shifts within the collapsing swarm, then reoptimization will be efficient. However, if the optimum shift is significantly far from the swarm, the low velocities of the particles will inhibit tracking, and the swarm can even oscillate about a false attractor and along a line perpendicular to the true optimum (linear collapse) [6].

Various adaptations to PSO have been suggested to tackle the difficulties mentioned above. Most of them assume that the time of a change in the environment is known to the algorithm, or can be detected, e.g., by a reevaluation of the objective function at one or several of the attractors [16], [23].

The problem of outdated memory is usually solved by either simply setting each particle's memory position to its current position (i.e., erasing the memory), or by reevaluating every memory position and setting it to either the old memory or current particle position, whichever is better [16].

The approaches to counterbalance the effect of diversity loss can be grouped into three categories: The approaches of the first

Manuscript received June 18, 2004; revised February 8, 2005.

T. Blackwell is with the Department of Computing, Goldsmiths College, University of London, London SE14 6NW, U.K. (e-mail: t.blackwell@gold.ac.uk).

J. Branke is with Institute AIFB, University of Karlsruhe, Karlsruhe D-76128, Germany (e-mail: branke@aifb.uni-karlsruhe.de).

Digital Object Identifier 10.1109/TEVC.2005.857074

category introduce diversity after the problem changed. Hu and Eberhart [23] list a number of these which all involve randomization of the entire, or part of, the swarm. This is either in response to function change, or at some predetermined interval. The problem of approaches in this category is that randomization implies loss of information gathered so far, and it seems difficult to determine the right amount of randomization. Too much will resemble restart, while too little does not solve the problem of convergence.

The second category attempts to maintain diversity throughout the run. For PSO, this may be achieved by integrating a sort of repulsion. In static environments, Krink *et al.* [29] proposed ways to handle collisions of spatially extended particles to avoid premature convergence. Parsopoulos and Vrahatis [34] use repulsion to keep particles away from already detected optima in an attempt to detect new ones. For dynamic environments, Blackwell and Bentley [5], [6] introduced charged PSO (CPSO), where some mutually repelling particles orbit a nucleus of neutral particles. This nucleus is, in fact, a conventional PSO swarm. The picture is reminiscent of classical pictures of the atom [6], although the orbits are chaotic rather than elliptical. The idea is that the charged subswarm maintains population diversity, at least within the spatial extent of the charged orbits, so that function change can be quickly (and automatically) registered, and the swarm can adapt. Meanwhile, the neutral swarm can continue to explore the neighborhood of the optimum in increasing detail. CPSO has been applied to a number of unimodal and bimodal test functions of high change frequency and spatial severity, and has been shown to work well, outperforming conventional PSO [1]. In [2], the authors have simplified the idea and replaced the charged particles by quantum particles that basically move to random positions around the swarm's global best. Another approach to maintain diversity is to replace the usual global neighborhood by a more local neighborhood. This reduces, at least temporarily, the pressure to move toward the global best, and thus allows to sustain diversity for a longer time. Li and Dam [31] tested this idea with a grid-like neighborhood structure. A similar approach is adopted by Janson and Middendorf [24] who use a hierarchical neighborhood structure, which has been shown to outperform the standard PSO in particular on unimodal dynamic problems.

In the final category, Blackwell and Branke [2] introduced a version of a multiswarm PSO, with the aim of maintaining a multitude of swarms on different peaks. This approach has been inspired by multipopulation EA approaches like the self-organizing scouts developed by Branke [13], which have shown to give excellent results on the tested problems. In the multiswarm approach, a part of the population clusters around any local optimum it may discover, and remains close to this optimum for further exploration. The remainder of the population continues to search for new local optima, and the process is repeated if any more local optima are found. This technique is expected to work well for a class of dynamic functions consisting of several peaks, where the dynamism is expressed by small changes to the peaks locations, heights, and widths. These have been argued to be representative of real world problems [11]. To track the optimum in such an environment, the algorithm has to be able to follow a moving peak, and to jump to another peak when the peak heights change in a way that makes a previously nonoptimal peak the highest peak. In order to allow each subswarm to track its peak, in [2], the multiswarm idea has also been combined with the quantum particle idea to sustain diversity within a swarm. Another multiswarm approach has recently been proposed by Parrott and Li [33]. There, the number and size of swarms is adjusted dynamically by a speciation mechanism called clearing [36], originally proposed for finding several optima in multimodal landscapes. While it also splits up the swarm into several subswarms, no additional diversity mechanisms have been included.

In this paper, we elaborate on our previous multiswarm PSO, adding a new operator (anti-convergence), providing guidelines for suitable parameter settings, and evaluating the approach on a variety of problem instances.

The multiswarm idea has also been proposed for purposes other than tracking dynamic environments. For example, the nichePSO approach of Brits *et al.* [15] is aimed at multimodal functions. It creates a two particle subswarm from a particle and its nearest spatial neighbor, if the variance in that particles fitness is less than a threshold. Subswarms may merge, and they also absorb particles from the main swarm. Although nichePSO was able to optimize successfully some static multimodal benchmark functions, it is not adaptable to the dynamic problem in an obvious way. This is because the algorithm, as the authors point out, depends on a very uniform distribution of particles in the search space, and on a training phase. The already mentioned multiswarm approach with clearing has also been used in [30] for detecting several optima in multimodal environments. In [38], different swarms are used to optimize different parts of a solution cooperatively. A two swarm approach for min–max optimization was proposed in [37]. Kennedy [26] suggested to cluster particles in each iteration into subswarms, and have them determine their global best separately. Finally, multiswarm variants have also been suggested in a nonoptimization context [7].

The paper is structured as follows. Section II provides a brief introduction on PSO, and then describes in detail the proposed multiswarm PSO. Also, in that section, some basic guidelines on parameter settings are suggested. Then, in Section III, the multiswarm approach is empirically evaluated on the moving peaks benchmark and compared with alternative approaches from the literature. This paper concludes with a summary and some ideas for future work.

II. PSO and Multiswarms

In this section, we will first provide a more formal introduction to PSO, then discuss some general aspects of multiswarms, and finally describe our multiswarm algorithm. For a more extensive treatment of PSO techniques, the reader is referred to [17], [19], and [20].

A. Particle Swarm Optimization (PSO)

As has already been explained in the introduction, particles move through the search space driven by attraction to their per-

sonal best found solution so far, and the best found solution in the neighborhood.

A particle i is defined by its position $\vec{x}_i$, the position of its personal best solution found so far, $\vec{p}_i$, and its velocity $\vec{v}_i$. Furthermore, each particle knows the best solution found so far by any of its "neighbors." Different particle topologies are explored in [24] and [28], but the standard neighborhood is global (*gbest*), i.e., all particles know the position of the globally best solution found so far $\vec{p}_g$. In the beginning, particle positions and velocities are generated randomly. The algorithm then proceeds iteratively, updating first all velocities, and then all particle positions as follows:

$$\vec{v}_i = \chi\left[\vec{v}_i + c_1\vec{\epsilon_1}\cdot(\vec{p}_g - \vec{x}_i) + c_2\vec{\epsilon_2}\cdot(\vec{p}_i - \vec{x}_i)\right] \quad (1)$$

$$\vec{x}_i = \vec{x}_i + \vec{v}_i. \quad (2)$$

The constriction factor $\chi < 1$ acts like friction, slowing the particles, so that finer exploration is achieved. c_1 and c_2 control the relative attraction to the global best and personal best found solutions, respectively. Finally, $\vec{\epsilon_1}$ and $\vec{\epsilon_2}$ are vectors of random variables drawn with uniform probability from [0, 1].

The overall algorithm is summarized as Algorithm 1 (unless stated otherwise, throughout this paper, we assume maximization problems without loss of generality).

Algorithm 1 Standard PSO Algorithm
FOR EACH particle i
  Randomly initialize $\vec{v}_i$, $\vec{x}_i = \vec{p}_i$
  Evaluate $f(\vec{p}_i)$
  $\vec{p}_g := \arg\max\{f(\vec{p}_i)\}$
REPEAT
  FOR EACH particle i
    Update particle position $\vec{x}_i$
    according to Eqs. (1) and (2)
    Evaluate $f(\vec{x}_i)$
    //Update personal best
    IF $f(\vec{x}_i) > f(\vec{p}_i)$ THEN
      $\vec{p}_i := \vec{x}_i$
    //Update global best
    IF $f(\vec{x}_i) > f(\vec{p}_g)$ THEN
      $\vec{p}_g := \arg\max\{f(\vec{p}_i)\}$
UNTIL termination criterion reached

B. Multiswarms—General Considerations

The fundamental idea of the proposed approach is to divide up the swarm into a number of subswarms, with the aim to position each of those subswarms on different, promising peaks of the landscape. However, simply breaking up the neighborhoods and dividing up the global swarm into a number of independent swarms is unlikely to be effective since the swarms would not interact (the dynamics governing the position and velocity updates of a particle in a particular swarm are specified by parameters belonging to that swarm only).

In this paper, two forms of swarm interaction are applied: *exclusion* and *anti-convergence*. Exclusion is a local interaction between colliding swarms, preventing swarms from settling on the same peak. Anti-convergence is an information sharing interaction among all swarms in the multiswarm algorithm, with the aim of allowing new peaks to be detected. Additionally, as a diversity preserving technique, each swarm has some charged or quantum particles. While the authors have used exclusion and quantum swarms before [2], anti-convergence is a completely new operator. We here motivate each development in turn.

1) Exclusion: If a swarm is divided into a number of subswarms, it may happen that particles from different swarms cluster around a single peak. This is undesirable since the motivation behind a multiswarm approach is to position different swarms on different peaks. Should one of the watched peaks become optimal, a swarm will already be present to take advantage. We therefore require diversity among the swarms. This requirement is called *swarm diversity*, as opposed to particle diversity which refers to the diversity of particles within a swarm [3]. Swarm diversity might arise from repulsions between particles from different swarms, but this would not prevent equilibriums where the attractions toward the peak are in balance with the repulsions between competing swarms. No single swarm is able to move closer to the peak and optimization would cease.

In order to prevent this, we use a simple competition among swarms that are close to each other. The winner is the swarm with the best function value at its swarm attractor. The loser is expelled and reinitialized in the search space; the winner remains. Swarms can be considered to be close to each other when their swarm attractors (best solutions found so far by each swarm) are within an exclusion radius r_{excl}. Exclusion thus constitutes local interaction among colliding swarms.

2) Anti-Convergence: As each swarm converges to a peak, neutral particles collapse inwards in a shrinking nucleus, while charged particles orbit chaotically around the peak [4]. The spatial extent of the neutral subswarm is therefore a suitable criterion for swarm convergence; swarm convergence is thus defined as the case of the neutral swarm size being less than a convergence radius r_{conv}, and multiswarm convergence occurs when all swarms have converged. But if the number of swarms is less than the number of peaks in the fitness landscape, and all swarms have converged (to different peaks, because exclusion is in force), the system will have lost its peak-detection capability. If one of the unwatched peaks becomes promoted to optimal due to an increase in peak height, or a completely new peak appears, the converged swarms may not be able to respond. Therefore, we advocate here the use of an anti-convergence operator. Whenever all swarms have converged, anti-convergence expels the worst swarm from its peak and reinitializes it in the search space. As a result, there is at least one swarm watching out for new peaks. Anti-convergence is a global interaction because it is assumed that all swarms are aware of each other's convergence status; it is information sharing between *swarms*. By contrast, the updating of the swarm attractor is information sharing between *particles* in the same swarm.

3) Charged and Quantum Swarms: In order to allow a swarm to follow a moving peak, particle diversity within a swarm is necessary. In CPSO, particle diversity is maintained by an interparticle Coulomb repulsion between charged particles. A measure of this diversity is the spatial size of the

charged swarm. However, the charged swarm size fluctuates greatly due to the chaotic nature of the orbits [8]. This might have drawbacks in multi-CPSO, where it is desired to find swarms on and around localized optima. Furthermore, CPSO suffers from quadratic complexity, arising from the Coulomb repulsions which are calculated between all pairs of particles in the charged subswarm.

It is probable that the success of CPSO over PSO in the dynamic context is only due to the increased diversity around the contracting PSO swarm, and that the exact dynamics are not important. In which case, simple randomization of the charged particles in a region surrounding the neutral swarm may be sufficient, superseding the need for expensive $O(N^2)$ computations, where N is the number or particles. This is the basis of the quantum swarm first proposed in [2].

Quantum swarm optimization (QSO) builds on the atomic picture of CPSO, and uses a quantum analogy for the dynamics of the charged particles. In the quantum model of the atom, the orbiting electrons are replaced by a quantum cloud. This cloud is actually a probability distribution governing where the electron will be found upon measurement. The electron does not follow a classical trajectory in between measurements. The probability distribution depends on the energy (and various other quantum numbers) of the electron. For example, electron positions in the lowest energy state of the hydrogen atom are distributed according to $p(r)dr = (const)r^2e^{-2r}$, where r is the distance from the nucleus, and $p(r)dr$ the probability that the electron is observed at a location with distance r [22]. By stretching the quantum analogy still further, a measurement corresponds to a function evaluation. At this point the charged particles are randomized within a ball of radius r_{cloud} centered on the swarm attractor. This provides a very simple update rule and corresponds to a uniform (and very unphysical) probability distribution. The velocity of the charged particle is irrelevant (it is indeterminate in the quantum picture) and the charged particles are not repelled from other charged particles or attracted to any attractor. However, any good location that they do find by virtue of their random positioning around the swarm attractor may still be useful to the neutral swarm due to information sharing. The multi-QSO proposed in this paper is an assembly of QSO swarms. The only interaction between these swarms occurs when swarms collide and the exclusion principle is applied, or due to the anti-convergence mechanism described above.

C. Proposed Multiswarm Algorithm

Based on the considerations above, the proposed multiswarm algorithm works as follows. After initialization, the algorithm iterates through a main loop with five stages: test for convergence, test for exclusion, test for function change, particle update, and attractor update. Test for convergence checks whether all swarms have converged and if so, marks the worst swarm for randomization (set init-bit to TRUE). Similarly, exclusion tests all pairs of swarms whether they are too close and if so, marks the inferior swarm for randomization. If a change in the environment is detected, all particle bests are reevaluated and any randomization is canceled. Then, depending on whether the swarm is to be randomized or not, the particles are reinitialized or simply updated. These stages are described in Algorithm 2.

Note that each particle gains an additional index n indicating the swarm it belongs to, i.e., particle ni is particle i of swarm n. Particle update rules depend on whether the particle is of neutral, charged, or quantum type and are defined as follows.

1) For neutral particles

$$\vec{v}_{ni} = \chi[\vec{v}_{ni} + c_1\vec{\epsilon}_1\cdot(\vec{p}_{ng} - \vec{x}_{ni}) + c_2\vec{\epsilon}_2\cdot(\vec{p}_{ni} - \vec{x}_{ni})] \quad (3)$$

$$\vec{x}_{ni} = \vec{x}_{ni} + \vec{v}_{ni}. \quad (4)$$

2) For quantum particles

$$\vec{x}_{ni} \in B_n(r_{\text{cloud}}). \quad (5)$$

3) For charged particles

$$\vec{v}_{ni} = \chi\left[\vec{v}_{ni} + c_1\vec{\epsilon}_1\cdot(\vec{p}_{ng} - \vec{x}_{ni}) + c_2\vec{\epsilon}_2 \cdot(\vec{p}_{ni} - \vec{x}_{ni})\right] + \vec{a}^{+}_{ni} \quad (6)$$

$$|\vec{v}_{ni}| \le \vec{v}_{\text{clamp}} \quad (7)$$

$$\vec{x}_{ni} = \vec{x}_{ni} + \vec{v}_{ni} \quad (8)$$

$$\vec{a}^{+}_{ni} = \sum_{l=1\ldots N^{+}_{n}, l\neq i} \frac{Q_k Q_1}{|\vec{x}_{ni} - \vec{x}_{nl}|^3}(\vec{x}_{ni} - \vec{x}_{nl}) \quad (9)$$

$$\left|a^{+}_{ni}\right| \le a_{\text{clamp}}. \quad (10)$$

Apart from the variables already defined in Section II-A, the following notation has been used: neutral and quantum particles have charge $Q_k = 0$ and charged particles have the same charge $Q_k = Q > 0$. The interactions between particles of the same swarm are parameterized by the usual PSO parameters (neutral and charged particles), Q (charged particles), and the radius r_{cloud} of the d-dimensional ball B_n centered on $\vec{p}_{ng}$ (quantum particles). The Coulomb repulsion, unlike earlier CPSO implementations ([1]–[8]), has an unprotected denominator. In principle, very large accelerations can occur if two charged particles are very close. In order to prevent singularities, $|\vec{a}^{+}_{nk}|$ is clamped to a very large number, a_{clamp}, cf. (10) [in practice, the Java constant $\text{Double.MAX\_VALUE} = 1.8\cdot 10^{308}$ is used), and $|\vec{v}_{nk}|$ is clamped to the dynamic range X (cf. (7)]. Charged particle velocity clamping tames the large fluctuations that can occur with Coulomb forces [8]. Choosing X for $\vec{v}_{\text{clamp}}$ ensures that any displaced charged particle will not stray too far from the search space X^d. Note that neutral particle clamping is not necessary since constriction ensures convergence [17].

The local swarm interaction, exclusion, is parameterized by the exclusion radius r_{excl}. When the swarm attractors of two swarms (the swarm attractors of converging swarms are invariably very close to the swarm center of mass [8]) are within r_{excl}, we assume that the swarms are overlapping and competing for the same peak. The response to this condition is to reinitialize the worse performing swarm (as determined by $f(\vec{p}_{ng})$) at that time in X^d. Exclusion can be turned off by setting r_{excl} to zero.

A swarm has converged if its neutral size $|S_n|$ is less than a convergence parameter r_{conv}. The swarm size is defined as the largest component separation between any two particles, stated mathematically as $|S_n| = \max_{kl}\max_j|(\vec{x}_{nk} - \vec{x}_{nl})*\vec{e}_j|$, where $\vec{e}_j$ is the unit vector in direction j. This diversity measure has been used to analyze the convergence of the neutral swarm around a single peak [8]. A scaling law for diversity loss as a function of iteration time t, $|S_n| = C\alpha^t$, C, α constants, $\alpha < 1$, has been motivated by theoretical and empirical studies.

If all swarms have converged, the worst swarm (with respect to $f(\vec{p}_{ng})$) is reinitialized in X^d. Note that anti-convergence can be turned off by setting r_{conv} to zero.

Multiswarms are composed of either neutral and charged particles or neutral and quantum particles. A convenient representation of the multiswarm *configuration* is $M(N+N^+)$ or $M(N+N^q)$, where N, N^+ and N^q are the numbers of neutral, charged and quantum particles in each swarm, and M is the total number of swarms in the multiswarm. $M(N+N^+)$ and $M(N+N^q)$ also evaluate to the total number of particles. In the following sections of this paper, where an actual configuration is specified, e.g., $10(5+5)$, the second group of particles is either charged or quantum; charged in the case of multiswarm optimization with CPSO swarms (mCPSO), or quantum for multi-QSO (mQSO).

In summary, the multi-swarm is a colony of M interacting swarms. Neutral particles in any swarm always follow a pure PSO position and velocity update rule. Charged classical particles obey neutral dynamics, but are also mutually repelled from other charged particles in their own swarm. A quantum particle, on the other hand, does not follow a classical rule; simply, upon measurement (a function evaluation), it is to be found in $B_n(r_{\text{cloud}})$. Particles within a swarm share information via the updated memory p_{ng}. Swarms interact both locally (exclusion) and globally (anti-convergence). A number of parameters have been introduced to specify these interactions, namely r_{excl}, r_{conv}, and either Q or r_{cloud} depending on whether charged swarms or quantum swarms are used. The next section presents some analytical arguments for bounds on these parameters.

Algorithm 2 Multiswarm Algorithm

```
//Initialization
FOR EACH particle ni
  Randomly initialize v_ni, x_ni = p_ni
  Evaluate f(p_ni)
FOR EACH swarm n
  p_ng := arg max{f(p_ni)}
  //Marker for randomization
  init[n] := FALSE
REPEAT
  //Anti-Convergence
  IF all swarms have converged THEN
    //Remember to randomize worst swarm
    init[worst swarm] := TRUE
  //Exclusion.
  FOR EACH PAIR of swarms n, m
    IF swarm attractor p_ng is within r_excl
    of p_mg THEN
      IF f(p_ng) ≤ f(p_mg) THEN
        init[n] := TRUE
      ELSE
        init[m] := TRUE
  FOR EACH swarm n
    //Test for Change
    Evaluate f(p_ng).
    IF new value is different from last
    iteration THEN
      Reevaluate each particle attractor.
      Update swarm attractor.
      //Cancel randomization
      init[n] := FALSE
    FOR EACH particle i of swarm n
      IF (init[n] = TRUE) THEN
        randomize particle
      ELSE
        //Update Particle
        Apply equations (3)-(9) depending
        on particle type.
      //Update Attractor
      Evaluate f(x_ni).
      IF f(x_ni) > f(p_ni) THEN
        p_ni := x_ni.
      IF f(x_ni) > f(p_ng) THEN
        p_ng := x_ni
    //Randomization complete
    init[n] := FALSE
UNTIL number of function evaluations
performed > max
```

D. Reflections on Parameter Settings

1) Particle Diversity: Assume a swarm has been sitting on a peak for a while and the neutral nucleus has shrunk to a size much smaller than the peak shift distance s. When the peak does shift, a charged or quantum particle is more likely to be close to the new optimum if s is commensurate with the average charged or quantum swarm size $\langle |S^{+,q}| \rangle$. This size is a measure of the average particle diversity. If s is much smaller than the average particle diversity, then it would be unlikely to find a charged or quantum particle close to the shifted peak. Conversely, if s is much larger than the average particle diversity, the swarm is not diverse enough to rapidly cope with the change [8]. We therefore guess that the optimal average charged or quantum particle displacement from the swarm center (assumed to be virtually coincident with p_g) is $0.5s$, leading to $\langle |S^{+,q}| \rangle = s$.

Predictions for the optimal values of the two parameters which determine the diversities of the charged and quantum swarms follow from this condition. The relationship between Q and $\langle |S^+| \rangle$ follows an empirical law $\langle |S^+| \rangle = AQ^k$, where A and k are constant for a particular configuration $1(N+N^+)$. This yields

$$Q = \left(\frac{s}{A}\right)^{\frac{1}{k}}. \tag{11}$$

This empirical law is derived from a series of experiments on the $d = 5$ static sphere function, dynamic range $X = 100$, with charged swarms of various configurations. [Note that these experiments differ slightly from earlier work on charged swarm size [8], since the charged particles interact through the clamped Coulomb repulsion and velocity instead of reduced repulsion when particles are within a distance of a core radius, cf. (10) and (7).] The equations of the best fit power law for the dependence of $|S^+|$ on Q are reported in Table I.

A quantum particle will be found, on average, at a displacement of $0.5r_{\text{cloud}}$ from the swarm center, so $\langle |S^q| \rangle = r_{\text{cloud}}$ and

TABLE I
CHARGED SWARM SIZE DEPENDING ON Q AND SWARM CONFIGURATION

Configuration $N + N^+$	Equation of best fit power law	Correlation Coefficient
$5 + 5$	$\lvert S^+\rvert = 4.9Q^{0.60}$	1.0
$10 + 10$	$\lvert S^+\rvert = 8.0Q^{0.58}$	1.0
$15 + 15$	$\lvert S^+\rvert = 10.3Q^{0.62}$	1.0
$20 + 20$	$\lvert S^+\rvert = 12.2Q^{0.62}$	1.0

this leads to a very simple prediction for the quantum diversity parameter

$$r_{\text{cloud}} = s. \tag{12}$$

2) Multiswarm Cardinality, M: The motivation for the introduction of multiswarms is to let each swarm sit on a different peak, so that if peak heights change and a previously inferior peak becomes optimal, a swarm is already there. Ideally, all peaks of the landscape would be covered by a swarm; we expect therefore that the multiswarm will perform well when the number of swarms is equal to the number of (significant) peaks in the landscape, i.e.,

$$M = p. \tag{13}$$

When $p < M$, there will be a number of swarms that will always be reinitialized as they become excluded from an occupied peak. These swarms cannot contribute much to the optimization, but they consume many function evaluations at the expense of swarms sitting on the peaks, doing the actual work. Thus, we expect performance to deteriorate. On the other hand, if $p > M$, there are more peaks than swarms, and anti-convergence will mitigate against performance deterioration due to low swarm diversity.

3) Exclusion: Since the idea behind exclusion is to ensure that only one swarm sits on a peak, the optimal value for r_{excl} can be estimated by considering the average diameter of the peak basin of attraction. Assuming that all p peaks are evenly distributed in X^d, the linear diameter of the basin of attraction of any peak, d_{boa} can be used to predict an optimal value for r_{excl}:

$$r_{\text{excl}} = 0.5 d_{\text{boa}}. \tag{14}$$

The basin diameters themselves can be estimated by fitting p peaks evenly into X^d to give, on average, $d_{\text{boa}} = X/p^{1/d}$.

4) Anti-Convergence: In general, we expect anti-convergence (or some other mechanism to keep at least one swarm patrolling) to be beneficial when $M < p$. If r_{conv} is too large, it is likely that the same swarm will be repeatedly reinitialized because it will never have the chance to converge. This might degrade performance because the poorer swarms will never be reinitialized, even when their peaks are comparably tiny. Alternatively, if r_{conv} is too small, the multiswarm will wait a long time before reinitialization, which again will degrade performance because one of the many unwatched peaks might have become optimal during this wait. On the whole, if r_{excl} is comparable to d_{boa}, we would expect $r_{\text{conv}} \leq r_{\text{excl}}$ since a converging swarm will be close to the peak center, inside the basin of attraction.

A lower bound for r_{excl} can be estimated by using the empirical result for the size of the contracting neutral swarm as a function of iteration t, $\lvert S(t)\rvert = C\alpha^t$, for constants $C \approx \lvert S(0)\rvert$ and $\alpha \approx 0.92$ (cf. [8, eq. (22)]). This scaling law is postulated to hold good for a neutral swarm converging toward a locally spherical symmetric optimum, which should hold if a neutral swarm is converging on a peak. In the following, we assume (optimistically, since we want to derive a lower bound) that the swarm immediately locates the new optimum after a change, and furthermore, that the neutral swarm size shortly after a peak shift $\lvert S(0)\rvert$ is close to s (i.e., the cloud of neutral particles spans the region between the old and new optimum location). That means, after a change, the swarm starts to contract around the new optimum, starting from $\lvert S(0)\rvert = s$. The number of iterations the PSO has to converge on the new peak can be calculated as $K_{\text{iter}} = (K - (M(N + N^{+,q} - 1)))/(M(1 + N + N^{+,q}))$, where K is the number of function evaluations between shifts, M is the number of swarms, and $N + N^{+,q}$ is the number of particles in each swarm. This estimation assumes that there will be $N + N^{+,q}$ function evaluations during the particle updates in a single iteration, followed by M function evaluations at the *test for change* stage. Furthermore, right after a change, $M(N+N^{+,q}-1)$ additional evaluations are required to reevaluate all particles. Then, the smallest a converging neutral swarm can shrink to is $s \cdot 0.92^{K_{\text{iter}}}$, which leads to the following prediction for the upper and lower limits of r_{conv}:

$$s \cdot 0.92^{K_{\text{iter}}} < r_{\text{conv}} \leq r_{\text{excl}}. \tag{15}$$

III. EXPERIMENTS

A. Experimental Framework

For performance evaluation, we used the publicly available moving peaks benchmark (MPB) [10]. It consists of a number of peaks, of varying heights and widths, moving by a fixed shift length s in random directions. Unless stated otherwise, the parameters have been set as follows: the search space has five dimensions $X^5 = [0, 100]^5$, there are $p = 10$ peaks, the peak heights vary randomly in the interval [30, 70], and the peak width parameters vary randomly within [1, 12]. The peaks change position every $K = 5000$ evaluations by a distance of $s = 1$ in a random direction, and their movements are uncorrelated (the MPB coefficient $\lambda = 0$). These parameter settings are summarized in Table II. They correspond to [10, Scenario 2] and have been chosen to facilitate comparisons with the self-organizing-scouts approach [14], a state-of-the-art EA for dynamic optimization problems. The termination condition for each experiment is 100 peak changes, corresponding to 500 000 function evaluations (or 413 900 function evaluations for the comparisons with Jansen and Middendorf [24]).

Scenario 2 actually specifies a family of benchmark functions, since the initial location, initial height and width of the peaks, and their subsequent development is determined by a pseudorandom number generator. All our results are based on

TABLE II
STANDARD SETTINGS FOR THE MOVING PEAKS BENCHMARK

Parameter	Setting
Number of peaks p	10
Number of dimensions d	5
Peak heights	$\in [30, 70]$
Peak widths	$\in [1, 12]$
Evals between changes	5000
Change severity s	1.0
Correlation coefficient λ	0

averages over 50 runs, where each run uses a different random number seed for the optimization algorithm, as well as the MPB. The primary performance measure is the offline error [14] which is the average over, at every point in time, the error of the best solution found since the last change of the environment. This measure is always greater or equal to zero and would be zero for perfect tracking.

The dynamics of the multiswarm models discussed is governed by three parameters r_{excl}, r_{conv} and Q, or r_{cloud}, respectively. One of the aims of the series of experiments is to investigate the sensitivity of the algorithm to these parameter settings. Another aim is to investigate the effect of different multiswarm configurations $M(N+N^+)$ and $M(N+N^q)$. In order to compare performance to the results from evolutionary techniques reported in [11], unless stated otherwise, the total population size (total number of particles), N_{pop} was fixed at 100 particles. The standard PSO parameters $c_{1,2} = 2.05$ and $\chi = 0.729\,843\,788$ have been well tested by many authors and have been shown to lead to convergence for noninteracting particles [17], and for interacting particles close to symmetric optima [8]. These standard PSO values were used for all experiments. The common "gbest" particle connectivity was also used, corresponding to a single neighborhood with a single swarm attractor. Initialization of the swarms involves placing the particles randomly in X^d with random velocities, also constrained to lie in X^d. Because we expect anti-convergence to have an effect only when there are more peaks than swarms, anti-convergence has been switched off unless stated otherwise.

B. Effect of Varying Multiswarm Configuration

The first set of experiments examines the effect of the multiswarm configuration on performance with the MPB and with the standard settings as mentioned above. Some of the experiments are similar to those reported in our earlier EvoSTOC paper [2], but differ in that we use the new clamped, intraswarm Coulomb repulsion and multiswarm parameter settings corresponding to our considerations in Section II-D. Besides, we have slightly modified the structure of the algorithm such that a swarm can be randomized at most once per iteration.

Many different configurations of 100 particles are possible. The number of swarms, M, can range from 1 (where the multiswarms reduce to single swarm PSO, CPSO, and QSO) to 100 (where the concept of a swarm is lost, since a lone particle cannot exchange information through the updating of p_{ng}). As we have argued, optimal configurations are likely to lie in between these extremes, with M being close to the number of peaks. As far as possible, symmetrical configurations (equal numbers of neutral and charged/quantum particles in each swarm) of 2–50 swarms were tested, along with the extremes. However, there are no configurations which have the same number of particles in each swarm for M in the range 11–19. In order to include a multiswarm in this range, the $14(4+3)$ configuration has been used, despite the fact that the total number of particles of that configuration is 98.

TABLE III
OFFLINE ERROR ± STANDARD ERROR FOR DIFFERENT CONFIGURATIONS

Configuration	mCPSO	mQSO
$2(25+25)$	10.19 ± 0.35	9.75 ± 0.33
$3(17+16)$	7.04 ± 0.26	6.73 ± 0.25
$4(13+12)$	4.94 ± 0.18	4.77 ± 0.18
$5(10+10)$	3.74 ± 0.14	3.71 ± 0.15
$10(10+0)$	2.32 ± 0.06	2.32 ± 0.06
$10(0+10)$	2.13 ± 0.08	1.93 ± 0.08
$10(5+5)$	2.05 ± 0.07	1.75 ± 0.06
$14(4+3)$	2.29 ± 0.07	1.93 ± 0.06
$20(3+2)$	2.89 ± 0.07	2.35 ± 0.07
$25(2+2)$	3.27 ± 0.08	2.69 ± 0.07
$50(1+1)$	15.35 ± 0.41	3.96 ± 0.09

The empirical law for the charged swarm size for a $(5+5^+)$ swarm is $|S^+| = 4.9Q^{0.60}$ (see Table I). Demanding that $|S^+| = s = 1.0$ implies [from (11)] $Q = 0.071$. The same demand for the quantum swarm leads to a parameter choice $r_{\text{cloud}} = s = 1.0$ (12), and (14) suggests $r_{\text{excl}} = 31.5$ since $d_{\text{boa}} = X/p^{1/d} = 100/10^{1/5} = 63.1$. Anti-convergence has been switched off for all runs. As will be shown later, it might have improved the results for $M < 10$, but would not fully compensate for the effect of an insufficient number of swarms.

The results of the experiments are reported in three groups: $M \in [2:50]$, the extreme cases $M = 1$ and $M = 100$, and some experiments where swarm interaction has been switched off. At the end of the section, we also compare convergence curves for some of the configurations.

1) Multiswarms: The effect of varying $M \in [2:50]$ on the offline error can be seen in Table III. A visualization is provided also in Fig. 1.

The swarm diversity (as opposed to the particle diversity within a swarm) is increased by increasing the number of swarms. The results show that performance improves with increasing swarm diversity, reaching an optimum at ten swarms. This confirms our expectation of $M = p$ being the optimal setting for parameter M. Increasing the number of swarms beyond ten reduces performance for two reasons: First, the additional swarms inevitably climb peaks that are already occupied, only to be randomized by exclusion. Thus, they

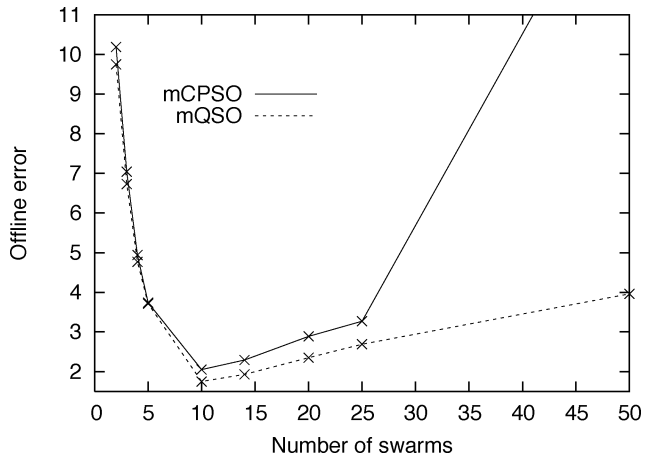

Fig. 1. Influence of number of swarms on offline error. For ten swarms, configuration $10(5+5)$ is plotted.

do not contribute to the optimization. Second, since we keep the total population size fixed, the swarms have less and less particles. In particular in the CPSO configuration, a 50-swarm, consisting of 50 of the smallest possible swarms (i.e., two particles) performs rather badly because there is no diversity in the charged swarm—a charged particle can only be repelled from other charged particles in the same swarm, and there are no other charged particles in this swarm. This is in contrast to the $50(1+1)$ multi-QSO, where particle diversity is still possible and the result is markedly better than the equivalent CPSO configuration. Notice how the quantum interaction is consistently and substantially better than the charged interaction, and this is independent of the configuration.

It is also interesting to compare performance among the $M = 10$ multiswarms. The interacting PSO's $10(10+0)$ configuration already performs much better than a single swarm. However, when comparing the results with the $10(0+10)$ configurations, it is evident that diversity within a swarm is also important. A multiswarm where each swarm consists of only charged particles performs better than purely neutral swarms, and using only quantum swarms is the best of the pure configurations. The combination of neutral and charged or quantum particles, however, performs best (see $10(5+5)$ configurations), because the charged or quantum particles help to track changes, but only the neutral particles can converge and rapidly improve a good solution.

2) Extreme Cases: Table IV shows the results for the extreme configurations with $M = 1$ and $M = 100$. Note that the $1(100+0)$ configuration corresponds to the standard PSO, and the $1(50{+}50^{+})$ and $1(50{+}50^{q})$ are just single swarm CPSO and QSO, respectively. The poor offline errors for the single swarms prove the efficacy of the multiswarm approach in this environment. There is no significant difference between the $100+0$ and $50+50$ configurations. This is surprising since the charged and quantum swarms have greater particle diversity and it would be expected that this would lead to better peak tracking, and hence better overall optimization. Also, charged or quantum particles have shown to be beneficial in the multiswarm configurations. However, a single large population has some inherent diversity anyway, and a shift severity of $s = 1.0$ for MPB is rather

TABLE IV
OFFLINE ERROR ± STANDARD ERROR FOR EXTREME CONFIGURATIONS

Configuration	mCPSO	mQSO
$1(100+0)$	16.40 ± 0.54	16.40 ± 0.54
$1(0+100)$	16.66 ± 0.53	16.81 ± 0.52
$1(50+50)$	16.65 ± 0.53	16.16 ± 0.52
$100(1+0)$	25.30 ± 0.77	25.30 ± 0.77
$100(0+1)$	25.95 ± 0.90	7.87 ± 0.20

mild. Overall, it seems that further increasing particle diversity in single swarms becomes important only for severely dynamic mono-modal environments [1]. The single swarms rapidly converge to a single peak after initialization, and there is not enough diversity at these parameter settings for the swarm to move away from this peak and search for other, possibly better, peaks. But on the other hand, diversity of even the neutral swarms seems sufficient to track the small movements of that peak. Hence, all single swarms are essentially equivalent. At $M = 100$, the concept of a swarm is lost since there can be no inter swarm interaction and information sharing. A single particle only has the memory of its best position visited in the past. Apart from the $100(0+1)$ QSO result, it is apparent that a single swarm, with local information exchange through p_g is preferable to 100 particles that can share information only through exclusion. This is perhaps because information exchange is guaranteed with the PSO and CPSO mechanism which assumes that all particles in a swarm are able to update their knowledge of p_g instantaneously. Conversely, information exchange through exclusion is local and can only happen when two swarms collide. The small discrepancy between $100(1+0)$ PSO and $100(0+1)$ CPSO is surely due to clamping. The interactions are identical because a lone charged particle experiences no Coulomb repulsion and so is subject to neutral particle dynamics.

The performance of the $100(0+1)$ multi-QSO is remarkable given the simplicity of the model. These single quantum particles are each repositioned randomly in a hyper sphere of radius 1.0 around the best position they have found, and interact through exclusion. Once more, the poor optimization potential of single charged particle swarms $100(0+1)$ CPSO is evident.

3) Many-Swarms: We call a multiswarm with $r_{\text{excl}} = 0.0$ (i.e., interaction switched off) a "many-swarm." A many-swarm configuration is denoted by $M*1(N+N)$. Looking at Table V and comparing the data with the corresponding entries from Table III, it becomes evident that exclusion is very important—it is the pressure that prevents two or more swarms optimizing the same peak, and hence promotes swarm diversity. Notice that ten noninteracting swarms is still better than one big one (cf. the $1(100+0)$ and $10(10+0)$ results). Since the many swarms are equivalent to a single swarm with different information sharing topologies, these results also illustrate the drawbacks of a single swarm with global neighborhood topology, presumably due to its diversity quenching effect.

4) Convergence: The convergence of the offline error over time is compared in Fig. 2 for standard PSO $1(100+0)$, many-swarm PSO $10*(10+0)$, multiswarm PSO with only neutral

TABLE V
OFFLINE ERROR ± STANDARD ERROR WHEN EXCLUSION OPERATOR IS SWITCHED OFF

Configuration	mCPSO	mQSO
$10 * 1(5+5)$	9.51 ± 0.70	9.38 ± 0.73
$10 * 1(10+0)$	9.89 ± 0.73	9.89 ± 0.73
$10 * 1(0+10)$	9.54 ± 0.67	8.81 ± 0.71

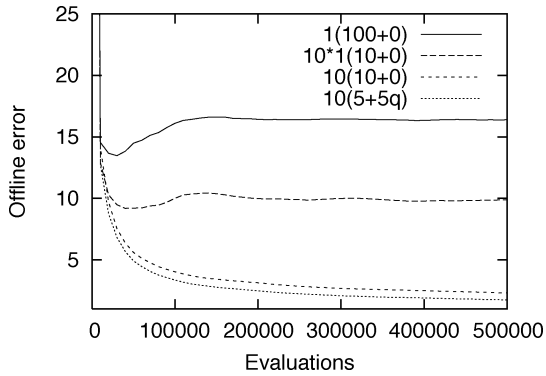

Fig. 2. Convergence of offline error over time, for different configurations.

particles $10(10+0)$, and mQSO $10(5+5^q)$. As can be seen, all approaches converge very quickly (the randomly generated initial population has an offline error of approximately 168). However, the standard PSO is soon trapped on a peak and the offline error increases again as another peak becomes optimal. The many-swarm suffers a similar fate, but performs better because it usually has swarms sitting on more than one peak. The multiswarms clearly do not suffer from that problem due to the exclusion parameter that prevents several swarms from sitting on a single peak. Finally, the offline error is further reduced by replacing some neutral particles with quantum particles.

C. Effect of Varying the Shift Severity

These experiments compare the different approaches with ten swarms, $10(10+0)$ multi-PSO, $10(5+5^+)$ multi-CPSO, $10(5+5^q)$ multi-QSO, and the $10*1(10+0)$ many-swarm on environments of increasing shift severity. Parameter settings are standard as above. According to our considerations in Section II-D, we vary the diversity parameter settings depending on the shift severity $s : Q = (s/A)^{1/k} = (s/4.9)^{1/0.6}$, and $r_{\text{cloud}} = s$.

The results are shown in Table VI. Interestingly, in the static environment ($s = 0$), multi-PSO, multi-CPSO, and multi-QSO all perform equally well, which indicates that the diversity preserving techniques are not harmful in a static environment. Naturally, overall performance degrades with increasing shift length, as the peaks are more and more difficult to track. As predicted, for $s > 0$, the increased diversities of the CPSO and QSO swarms enable better tracking of a peak as the shift severity worsens. Results are very consistent, with multi-QSO performing best, followed by multi-CPSO, and finally, the multi-PSO with only neutral particles. The many-swarm PSO without exclusion performs significantly worse than all others, which stresses again the importance of the exclusion operator.

TABLE VI
OFFLINE ERROR ± STANDARD ERROR FOR VARYING SHIFT SEVERITY

s	$10 * 1(10+0)$	$10(5+5^+)$	$10(5+5^q)$	$10(10+0)$
0.0	8.80±0.72	1.18 ±0.07	1.18 ±0.07	1.18±0.07
1.0	9.89±0.73	2.05±0.07	1.75±0.06	2.32±0.06
2.0	10.83 ± 0.69	2.80 ± 0.07	2.40 ± 0.06	3.37 ± 0.20
3.0	11.63 ± 0.64	3.57 ± 0.08	3.00 ± 0.06	4.24 ± 0.09
4.0	11.73 ± 0.57	4.18 ± 0.09	3.59 ± 0.10	5.08 ± 0.11
5.0	11.83 ± 0.62	4.89 ± 0.11	4.24 ± 0.10	5.90 ± 0.12
6.0	12.27 ± 0.61	5.53 ± 0.13	4.79 ± 0.10	6.62 ± 0.16

TABLE VII
OFFLINE ERROR ± STANDARD ERROR DEPENDING ON PARAMETERS Q AND r_{cloud}

Q	mCPSO	r_{cloud}	mQSO
0.0	2.32 ± 0.06	0.0	4.04 ± 0.09
0.001	2.34 ± 0.07	0.01	2.61 ± 0.07
0.01	2.23 ± 0.09	0.1	1.99 ± 0.08
0.071	2.05 ± 0.07	1.0	1.75 ± 0.06
0.1	2.04 ± 0.08	5.0	1.99 ± 0.06
1.0	2.40 ± 0.07	10.0	2.10 ± 0.06

D. Sensitivity With Respect to Within-Swarm Diversity Parameters Q and r_{cloud}

Table VII shows the impact of the parameters Q and r_{cloud} on the performance of multi-CPSO and multi-QSO, respectively. Standard settings were used for all other parameters. Configuration was $10(5+5^+)$ and $10(5+5^q)$.

For $Q = 0$, charged particles behave just like neutral particles, and the result is just the same as if a standard multi-PSO would have been used. For QSO, on the other hand, setting $r_{\text{cloud}} = 0$ renders the quantum particles useless, and performance is rather bad. The best performing parameter settings for MQSO are $Q = 0.071$ and $r_{\text{cloud}} = 1.0$ just as predicted. For mCPSO, $Q = 0.071$ and $Q = 0.1$ perform best (difference is not significant). In any case, it can be seen that the performance is not very sensitive to the examined parameters. Even selecting a value 1/10 or ten times its optimal setting performs better than not using the diversity mechanisms.

E. Effect of Varying the Exclusion Parameter r_{excl}

The effect of varying the exclusion radius r_{excl} is summarized in Table VIII and visualized in Fig. 3. As with the other parameters, the performance of our multiswarm PSO is not very sensitive to r_{excl}. The optimal setting of 30.0 is slightly smaller than the 31.5 suggested by (14), but good results are achieved for $1.0 \leq r_{\text{excl}} \leq 40$. Overall, the effect of setting r_{excl} a little too large seems to have a more severe impact than setting r_{excl} a little too small, thus one should probably always stay on the safe side and below the setting suggested by (14).

TABLE VIII
INFLUENCE OF r_{excl} ON PERFORMANCE OF $10(5 + 5^{+})$ mCPSO AND $10(5 + 5^{q})$ mQSO. RESULTS REPORT ON OFFLINE ERROR ± STANDARD ERROR

r_{excl}	mCPSO	mQSO
0.0	9.51 ± 0.70	9.38 ± 0.73
1.0	2.50 ± 0.12	2.20 ± 0.10
10.0	2.22 ± 0.10	2.05 ± 0.10
20.0	2.08 ± 0.08	1.78 ± 0.07
30.0	1.98 ± 0.07	1.72 ± 0.06
31.5	2.05 ± 0.07	1.75 ± 0.06
40.0	2.25 ± 0.09	1.93 ± 0.08
50.0	2.76 ± 0.12	2.31 ± 0.09

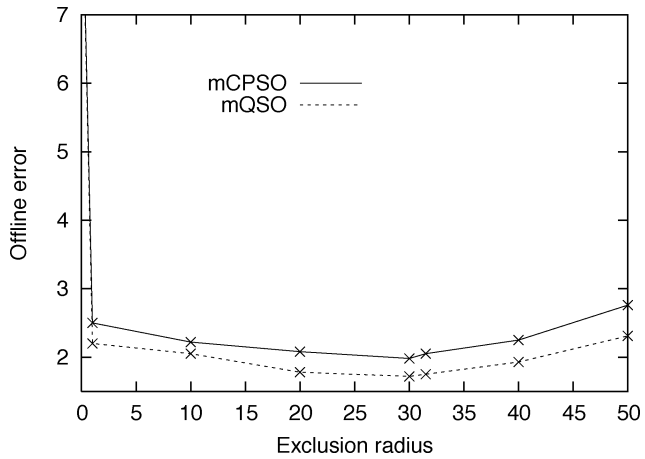

Fig. 3. Influence of exclusion on offline error.

TABLE IX
INFLUENCE OF r_{conv} ON PERFORMANCE OF $10(5 + 5^{+})$-CPSO AND $10(5 + 5^{q})$-QSO. RESULTS REPORT ON OFFLINE ERROR ± STANDARD ERROR

r_{conv}	mCPSO	mQSO
0.0	3.68 ± 0.11	3.65 ± 0.11
0.01	3.68 ± 0.11	3.65 ± 0.11
0.1	3.68 ± 0.11	3.65 ± 0.11
1.0	2.82 ± 0.07	2.68 ± 0.05
10.0	2.75 ± 0.05	2.59 ± 0.06
20.0	2.63 ± 0.06	2.52 ± 0.06
22.9	2.65 ± 0.06	2.50 ± 0.06
30.0	2.62 ± 0.06	2.50 ± 0.06
40.0	2.60 ± 0.06	2.50 ± 0.06
50.0	2.64 ± 0.06	2.50 ± 0.06
100.0	2.80 ± 0.06	2.65 ± 0.06

TABLE X
COMPARISON OF SINGLE SWARM OPTIMIZATION OF A SINGLE MOVING PEAK. RESULTS REPORT ON OFFLINE ERROR ± STANDARD ERROR

Configuration	CPSO	QSO
$1(100 + 0)$	0.87 ± 0.04	0.87 ± 0.04
$1(0 + 100)$	1.18 ± 0.04	0.37 ± 0.02
$1(50 + 50)$	0.73 ± 0.03	0.47 ± 0.02

F. Effect of Varying Anti-Convergence Parameter r_{conv}

The last parameter introduced by our approach is r_{conv}, determining when anti-convergences is triggered. Since exclusion is predicted to be advantageous only when the number of peaks is larger than the number of swarms ($p > M$), this series of experiments was run with $p = 50$ and $M = 10$.

Note that with a different number of peaks, our considerations in Section II-D suggest a modification of the exclusion radius according to (14), we therefore select $r_{\text{excl}} = 22.9$.

With $K_{\text{iter}} = (5000 - 90)/110 = 44.64$, based on the considerations in Section II-D, we would expect that setting r_{excl} below 0.024 will have no effect since a neutral swarm cannot shrink below this size, so the criterion for anti-convergence is never attained. Accordingly, (15) suggests $0.024 \leq r_{\text{conv}} \leq 22.9$.

The empirical results in Table IX clearly show that the lower bound is not tight, as no change in performance can be observed for $r_{\text{conv}} \leq 0.1$, some five times larger than predicted. We assume that this is because the swarm does not immediately find the new optimum, and K_{iter} thus overestimates the number of iterations for converging. Also, the upper bound of (15), $r_{\text{conv}} \leq r_{\text{excl}} = 22.9$ seems low, since there is no significant difference in performance for $r_{\text{conv}} \leq 50$. Nevertheless, the range specified by (15) is a good guide, and as it seems, setting r_{conv} to the upper limit r_{excl} works well in practice in our cases.

The best results are at $r_{\text{conv}} = 40.0$ and $r_{\text{conv}} \in [22.9, 50]$, with offline errors of 2.60 ± 0.06 and 2.50 ± 0.06 for mCPSO and mQSO, respectively. The importance of anti-convergence is illustrated by the fact that the result at $r_{\text{conv}} = 0.0$ is significantly worse than the optimal results with anti-convergence.

G. Experiments on a Single Peak

To isolate the particle diversity features of charged particles and quantum particles, and to demonstrate their benefit even in simple dynamic environments, this section shows some results of experiments with a single swarm on a single moving peak, shift severity $s = 1.0$ and a change every 5000 evaluations. The results are summarized in Table X. Clearly, CPSO and QSO in their default $1(50 + 50)$ configurations outperform the standard PSO without particle diversity feature. As in previous tests, QSO outperforms CPSO. If all particles are charged, performance is significantly worse than the neutral swarm. It seems that the purely charged swarm has a problem with convergence, and the peak shifts are too small to outweigh this disadvantage with the benefit of increased particle diversity. On the other hand, a pure quantum swarm, being basically a local optimizer, performs very well on the single peak environment.

TABLE XI
ABILITY OF A $10(5+5)$ MULTISWARM TO OPTIMIZE ENVIRONMENTS WITH VARYING NUMBERS OF PEAKS. THE TABLE SHOWS OFFLINE ERROR $\pm$ STANDARD ERROR BOTH WITH AND WITHOUT ANTI-CONVERGENCE

	with anti-convergence		without anti-convergence	
p	mCPSO	mQSO	mCPSO	mQSO
1	4.93 ± 0.17	5.07 ± 0.17	4.93 ± 0.17	5.07 ± 0.17
2	3.36 ± 0.26	3.47 ± 0.23	3.36 ± 0.26	3.47 ± 0.23
5	2.07 ± 0.08	1.81 ± 0.07	2.07 ± 0.08	1.81 ± 0.07
7	2.11 ± 0.11	1.77 ± 0.07	2.11 ± 0.11	1.77 ± 0.07
10	2.08 ± 0.07	1.80 ± 0.06	2.05 ± 0.07	1.75 ± 0.06
20	2.64 ± 0.07	2.42 ± 0.07	2.95 ± 0.08	2.74 ± 0.07
30	2.63 ± 0.08	2.48 ± 0.07	3.38 ± 0.11	3.27 ± 0.11
40	2.67 ± 0.07	2.55 ± 0.07	3.69 ± 0.11	3.60 ± 0.08
50	2.65 ± 0.06	2.50 ± 0.06	3.68 ± 0.11	3.65 ± 0.11
100	2.49 ± 0.04	2.36 ± 0.04	4.07 ± 0.09	3.93 ± 0.08
200	2.44 ± 0.04	2.26 ± 0.03	3.97 ± 0.08	3.86 ± 0.07

TABLE XII
EXPERIMENT WITH A TEN-DIMENSIONAL MPB. THE TABLE REPORTS OFFLINE ERROR $\pm$ STANDARD ERROR FOR DEFAULT PARAMETER SETTINGS ($r_{\text{excl}} = r_{\text{excl}} = 39.7$, $r_{\text{cloud}} = s = 1.0$, $Q = 0.071$, $M(N+N) = 10(5+5)$) AND DEVIATIONS THEREOF

	Offline error	
Parameter setting	mQSO	mCPSO
default	4.25 ± 0.14	4.49 ± 0.15
$r_{excl} = 47.7$	4.32 ± 0.16	4.45 ± 0.13
$r_{excl} = 31.8$	4.70 ± 0.18	4.43 ± 0.18
$r_{conv} = 47.7$	4.25 ± 0.14	4.49 ± 0.15
$r_{conv} = 31.8$	4.25 ± 0.14	4.48 ± 0.15
$r_{cloud} = 1.2$	4.36 ± 0.15	n/a
$r_{cloud} = 0.8$	4.25 ± 0.14	n/a
$Q = 0.085$	n/a	4.55 ± 0.15
$Q = 0.057$	n/a	4.55 ± 0.14
$12(4+4)$	4.60 ± 0.15	4.86 ± 0.14
$8(6+6)$	4.17 ± 0.15	4.40 ± 0.14

H. Effect of Varying the Number of Peaks

These experiments investigate how our approach scales with the number of peaks. Also, they validate the intuitive idea that anti-convergence is relevant only when there are more peaks than swarms. The number of peaks is varied between 1 and 200, and tests are performed with and without anti-convergence for the $10(5+5)$ configuration. Q and r_{cloud} are set to standard values, r_{excl} is determined by (14), and r_{conv} is set to r_{excl} for runs with anti-convergence.

As can be seen in Table XI, increasing the number of peaks has only a relatively small impact on the offline error. Note that when the number of peaks is increased, two aspects of the problem change: the problem becomes harder, because there are more local optima and it is increasingly difficult to watch over so many peaks. On the other hand, the standard error even of random solutions is reduced, because the MPB fitness is the maximum over all the peaks, a value which increases monotonically with an increasing number of peaks. The multiswarm approach seems to be able to cope well with environments consisting of a large number of peaks. However, decreasing the number of peaks significantly below the number of swarms seems to cause the offline error to increase, compare for example, the 4.93 or 5.07 performance on the single peak with configuration $10(5+5)$ to the offline error of 0.73 or 0.47 achieved with configuration $1(50+50)$ (Table X). The reason is that there are always a number of swarms that cannot converge on a peak due to exclusion, thus they are basically useless, while using up precious function evaluations.

The multiswarms perform best when the number of peaks is equal or slightly smaller than the number of swarms, both with and without anti-convergence. This is in line with the intuition of associating a swarm with each peak. When $p < M$, because there are always swarms reinitialized due to exclusion, we would expect anti-convergence to have little or no effect and this is evident in the results. Anti-convergence slightly worsens optimization at $p = 10$ because further randomization of a swarm when all ten swarms have converged onto the ten peaks is not helpful. However, as the modality of the environment increases, anti-convergence offers an increasingly significant advantage. At $p = 200$, the offline error of mQSO with anti-convergence is 41% smaller than without. mQSO is consistently better than mCPSO independent of anti-convergence and for all $p > 2$.

I. Effect of Varying the Dimensionality

This section shall demonstrate that our general guidelines for setting parameters are also applicable for higher dimensions. For a MPB with standard settings but with 10 instead of 5 dimensions, our guidelines suggest the following parameter settings: $r_{\text{excl}} = r_{\text{conv}} = 39.7$, $r_{\text{cloud}} = s = 1.0$, $Q = 0.071$, $M(N+N) = 10(5+5)$. Table XII compares the resulting offline error with the offline error when slightly different parameter settings would have been chosen ($\pm$ 20%). As can be seen, the performance is very robust with respect to parameter settings, all results range between 4.17 and 4.70 for mQSO, and 4.40 and 4.86 for mCPSO. The default parameter settings are among the best, with 4.25 for mQSO and 4.49 for mCPSO, and the difference to the best found parameter settings is not statistically significant. These results confirm the general robustness of the approach, and show that our guidelines for parameter settings are appropriate also for higher dimensional problems.

J. Comparisons With Other Approaches

In the above, we have already compared the multiswarm approach with the standard PSO, and with the use of many-swarm neighborhood structure. In this section, the proposed multiswarm QSO is compared with some other approaches from the literature.

TABLE XIII
PERFORMANCE OF STANDARD PSO WITH RE-INITIALIZATION OF A FRACTION ρ OF THE POPULATION, DEPENDING ON SHIFT SEVERITY s

	Shift severity s		
Algorithm	0.0	1.0	10.0
$N = 30, \rho = 0\%$	16.41 ± 0.54	17.47 ± 0.53	19.76 ± 0.51
$N = 30, \rho = 10\%$	14.31 ± 0.51	14.81 ± 0.45	15.85 ± 0.49
$N = 30, \rho = 50\%$	12.77 ± 0.49	12.92 ± 0.39	13.38 ± 0.44
$N = 30, \rho = 90\%$	12.60 ± 0.46	12.55 ± 0.45	14.40 ± 0.38
$N = 100, \rho = 0\%$	16.17 ± 0.58	16.40 ± 0.54	18.23 ± 0.54
$N = 100, \rho = 10\%$	13.93 ± 0.58	13.81 ± 0.50	14.38 ± 0.45
$N = 100, \rho = 50\%$	11.82 ± 0.46	13.14 ± 0.47	13.54 ± 0.44
$N = 100, \rho = 90\%$	11.74 ± 0.47	13.18 ± 0.47	14.24 ± 0.39
QSO $10(5 + 5^q)$	1.18 ± 0.07	1.75 ± 0.06	6.96 ± 0.15

TABLE XIV
PERFORMANCE OF STANDARD PSO WITH REINITIALIZATION OF A FRACTION ρ OF THE POPULATION, DEPENDING ON THE NUMBER OF PEAKS p

	Number of peaks p		
Algorithm	1	10	200
$N = 30, \rho = 0\%$	1.17 ± 0.04	17.47 ± 0.53	18.09 ± 0.47
$N = 30, \rho = 10\%$	1.16 ± 0.04	14.81 ± 0.45	9.98 ± 0.16
$N = 30, \rho = 50\%$	1.17 ± 0.04	12.92 ± 0.39	8.46 ± 0.12
$N = 30, \rho = 90\%$	1.46 ± 0.05	12.55 ± 0.45	8.00 ± 0.11
$N = 100, \rho = 0\%$	0.87 ± 0.04	16.40 ± 0.54	17.81 ± 0.53
$N = 100, \rho = 10\%$	0.89 ± 0.04	13.81 ± 0.50	9.57 ± 0.14
$N = 100, \rho = 50\%$	1.00 ± 0.04	13.14 ± 0.47	8.15 ± 0.10
$N = 100, \rho = 90\%$	1.59 ± 0.06	13.18 ± 0.47	8.06 ± 0.11
QSO10$(5 + 5^q)$	5.07 ± 0.17	1.75 ± 0.06	2.26 ± 0.03

1) Comparison With Partial Reinitialization: One straightforward approach to make PSO more suitable for dynamic environments is to randomize a certain number of particles after a change [23]. In the tests reported below, we use this mechanism in combination with reevaluating each particle attractor after a change, just as we did for our multiswarm approaches in order to prevent outdated memory information misguiding the search. In [23], Hu and Eberhart use a population size of 30, and because we wanted to keep as many parameters as possible equal, we tested population sizes of 30 and 100, randomizing $\rho \in \{0\%, 10\%, 50\%, 90\%\}$ of the particles after a change. As test scenarios, we looked at MPB with different change severities $s \in \{0, 1.0, 10.0\}$ and different numbers of peaks $p \in \{1, 10, 200\}$.

Table XIII summarizes the results on different change severities, with all other settings corresponding to our default settings. For comparison, the result of the $10(5 + 5^q)$ QSO are also included. As can be seen, the population size seems to have only a marginal effect. However, randomizing a fraction of the particles can help significantly, as it reintroduces diversity and avoids getting stuck on a peak that is no longer optimal. For the considered environment, the higher the percentage of randomized particles, the better the performance, except for the very strong shift severity of $s = 10.0$, where 50% randomization performed best. This deviates from the findings of Hu and Eberhart [23], where a low to medium randomization strategy performed best. The difference can be explained by the different test environments: while our problem has ten peaks, the one considered by Hu and Eberhart is unimodal. Obviously, if there are several peaks with the possibility that another peak becomes the optimal one, the information about the location of the previous optimum is less valuable, and higher randomization performs better. Independent of the parameter settings, QSO performs an order of magnitude better than randomization.

The results on different numbers of peaks assuming a shift severity of $s = 1.0$ are reported in Table XIV. In accordance to Hu and Eberhart's results, on the single peak, a randomization fraction of about $\rho = 10\%$ performs best, at least for population size of 30 (cf. Table XIV). A larger population performs better in combination with low randomization ($\rho < 10\%$), while with high randomization, a smaller population size seems slightly better. This is not surprising as both a large population size as well as randomization are means of increasing diversity and can, to same extent, substitute each other. Also, with larger randomization, quick convergence becomes important, which is facilitated with smaller population sizes. On the 10 and 200 peak environment, QSO $10(5 + 5^q)$ again performs much better than either of the randomization approaches. On the single moving peak, QSO $10(5 + 5^q)$ suffers from the fact that it cannot utilize more than one population, which significantly decreases performance as has already been discussed in Section III-H. On the other hand, if it is assumed that the environment is unimodal, and the number of swarms is chosen according to (13), then QSO $1(50 + 50^q)$ with a single population again performs much better than either of the randomization approaches (0.47 ± 0.02, cf. Table X).

2) Comparison With Hierarchical Swarms: Jansen and Middendorf [24] propose an adaptation of PSO to dynamic environments by introducing a dynamic and hierarchic neighborhood structure. This is used to maintain some particle diversity, useful in dynamic environments. The approach has been tested on a MPB with a 40 particle swarm and 100 iterations between changes. There are 41 evaluations per iteration (each particle, and the global best to check for change) and an additional 39 evaluations if change is detected. So there are 4139 evaluations between function changes. Other parameters were 50 peaks, and a shift severity of $s = 3.0$.

Jansen and Middendorf [24] report an offline error of around 8.5 for the hierarchical swarm after 8000 iterations (=80 peak shifts), which is not improved further until the end of the run at iteration 30 000.

For comparison, we run a similar experiment with our $10(5 + 5^q)$ multiswarm approach. As suggested in Section II-D, we set our parameters as follows: $r_{\text{cloud}} = 3.0$, $Q = 0.441$, and $r_{\text{excl}} = r_{\text{conv}} = 22.9$. After 100 peak shifts, the offline error is 3.52, with a standard error of 0.06. Compared with the offline

error >8.0 of the hierarchical swarm, this corresponds to a reduction of 54%.

3) Comparison With Self-Organizing Scouts (SOS): The self-organizing scouts (SOS) approach is a state-of-the-art EA approach to dynamic optimization problems [14]. Since our multiswarms are inspired by SOS, it is interesting to compare the two approaches. In [13], results of SOS are reported for a number of different MPB instances. On the standard instance with 10 peaks, 5 dimensions, $s = 1.0$, and a change every 5000 function evaluations (which has also been our standard setting), SOS achieves an offline error of 4.01, while mQSO in its $10(5+5)$ configuration achieves 1.75 (cf. Table III), which corresponds to a reduction by 56%! For $s = 3.0$ and otherwise identical parameter settings, for SOS an offline error of 6.54 has been reported, while according to Table VI, mQSO has an error of only 3.13, corresponding to a reduction by 56%. And with 200 peaks and $s = 1.0$, SOS yields an offline error of 3.62, while mQSO produces an error of only 2.63, a reduction of 38%.

So far, we can only guess why our PSO-based multiswarm approach works so much better than a similar EA-based multipopulation approach on the problems we used for testing. It might be due to the better local optimization capability of PSO.

IV. Conclusion

In a dynamic environment, it is important that the optimization algorithm is able to continuously track the moving optimum over time. In this paper, we extend and explore a PSO approach from [2] to tackle dynamic environments. The approach has been motivated by the multipopulation approaches in the evolutionary computing area, which try to keep a number of subpopulations "watching" over, and following, promising peaks of the fitness landscape. Following that idea, it was suggested to divide up the swarm into several subswarms and to incorporate the following three diversity operators.

1) *Quantum particles*. Standard PSO swarms converge over time, thereby losing diversity, and thus their ability to quickly react to a peak's move. Quantum particles appear at random positions, uniformly distributed around the swarm's global best. They ensure a swarm's particle diversity, and thus its adaptability. Besides quantum particles, we have also considered the earlier published charged PSO as an alternative throughout this paper.
2) *Exclusion*. The idea of multiple swarms is to distribute them over the different peaks in the landscape. Two or more swarms sitting on the same peak is a waste of capacity. Exclusion is a local interaction between swarms, aimed at ensuring swarm diversity: whenever two swarms are getting too close (one swarm's global best lies within a distance of r_{excl} from the other swarm's global best), the two swarms compete and the one with lower fitness is reinitialized.
3) *Anti-convergence*. In an environment, where there are possibly more peaks than swarms, it is necessary to keep constantly patrolling for new and better peaks. To this end, the paper proposed an anti-convergence operator, which reinitializes the worst of all swarms once all swarms have converged. Note that this operator constitutes a global swarm interaction.

For all additional parameters introduced by the above modifications, we have analytically derived guidelines for suitable settings.

The approach has been tested extensively on a variety of instances of the MPB, with varying shift severity and between 1 and 200 peaks. The results show that it performs well over a wide range of problem characteristics, that it is rather robust with respect to parameter settings, and that the parameter settings derived analytically are indeed appropriate.

Compared with previously published results on the MPB, namely, SOS [13] and a hierarchical PSO [24], our approaches cut the offline error approximately by half. Compared with randomization scheme [23], the improvement was even about an order of magnitude.

Overall, we can conclude that our approach is very suitable for dynamic environments, being robust and dramatically outperforming previous approaches on the same benchmark problem.

The only aspect where performance was below expectations is the case where the number of swarms is much larger than the number of peaks. This could perhaps be alleviated by making the number of swarms self-adaptive, allowing them to join and to split as needed. This is future work.

Acknowledgment

The authors would like to thank X. Yao and the three anonymous referees for their helpful comments.

References

[1] T. Blackwell, "Swarms in dynamic environments," in *Proc. Genetic and Evol. Comput. Conf.*, vol. 2723, E. Cantu-Paz, Ed., 2003, pp. 1–12.
[2] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," in *Applications of Evolutionary Computing*. ser. Lecture Notes in Computer Science, G. R. Raidl *et al.*, Eds. Berlin, Germany: Springer-Verlag, 2004, vol. 3005, pp. 489–500.
[3] T. M. Blackwell, "Particle swarms and population diversity I: Analysis," in *Proc. GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, J. Branke, Ed., 2003, pp. 9–13.
[4] ——, "Particle swarms and population diversity II: Experiments," in *Proc. GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, J. Branke, Ed., 2003, pp. 14–18.
[5] T. M. Blackwell and P. Bentley, "Don't push me! Collision-avoiding swarms," in *Proc. Congr. Evol. Comput.*, 2002, pp. 1691–1696.
[6] ——, "Dynamic search with charged swarms," in *Proc. Genetic Evol. Comput. Conf.*, W. B. Langdon *et al.*, Eds., 2002, pp. 19–26.
[7] T. Blackwell, "Swarm music: improvised music with multi-swarms," in *Proc. AISB Symp. Artif. Intell. Creativity Arts Sci.*, Aberystwyth, U.K., 2003, Society for the Study of Artificial Intelligence and the Simulation of Behavior, Univ. Wales, pp. 41–49.
[8] ——, "Particle swarms and population diversity," *Soft Computing*, vol. 9, no. 11, pp. 793–802, 2005.
[9] B. Brandstätter and U. Baumgartner, "Particle swarm optimization—Mass spring system analogon," *IEEE Trans. Magn.*, vol. 38, no. 2, pp. 997–1000, 2002.
[10] J. Branke. The moving peaks benchmark website. [Online]. Available: http://www.aifb.uni-karlsruhe.de/~jbr/movpeaks
[11] ——, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proc. Congr. Evol. Comput.*, vol. 3, 1999, pp. 1875–1882.
[12] ——, "Evolutionary approaches to dynamic environments—Updated survey," in *Proc. GECCO Workshop Evol. Algorithms for Dynamic Optimization Problems*, 2001, pp. 27–30.
[13] ——, *Evolutionary Optimization in Dynamic Environments*. Norwell, MA: Kluwer, 2001.

[14] J. Branke and H. Schmeck, "Designing evolutionary algorithms for dynamic optimization problems," in *Theory and Application of Evolutionary Computation: Recent Trends*, S. Tsutsui and A. Ghosh, Eds. Berlin, Germany: Springer-Verlag, 2002, pp. 239–262.
[15] R. Brits, A. Engelbrecht, and F. van den Bergh, "A niching particle swarm optimizer," in *Proc. Asia-Pacific Conf. Simulated Evol. Learning*, 2002, pp. 692–696.
[16] A. Carlisle and G. Dozier, "Adapting particle swarm optimization to dynamic environments," in *Proc. Int. Conf. Artif. Intell.*, 2000, pp. 429–434.
[17] M. Clerc and J. Kennedy, "The particle swarm: Explosion, stability, and convergence in a multi-dimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.
[18] J. P. Coelho, P. B. De Moura Oliveira, and J. E. Boaventura Cunha, "Nonlinear concentration control system design using a new adaptive particle swarm optimizer," in *Proc. Portuguese Conf. Autom. Control*, 2002, pp. 132–137.
[19] R. C. Eberhart and Y. Shi, "Special issue on particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, Jun. 2004.
[20] R. C. Eberhart and Y. Shi, *Swarm Intelligence*. San Mateo, CA: Morgan Kaufmann, 2001.
[21] ——, "Tracking and optimizing dynamic systems with particle swarms," in *Proc. Congr. Evol. Comput.*, 2001, pp. 94–100.
[22] A. French and E. Taylor, *An Introduction to Quantum Physics*. New York: Norton, 1978.
[23] X. Hu and R. Eberhart, "Adaptive particle swarm optimization: Detection and response to dynamic systems," in *Proc. Congr. Evol. Comput.*, 2002, pp. 1666–1670.
[24] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer for dynamic optimization problems," in *Applications of Evolutionary Computing*. ser. Lecture Notes in Computer Science, G. R. Raidl, Ed. Berlin, Germany: Springer-Verlag, 2004, vol. 3005, pp. 513–524.
[25] 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.
[26] J. Kennedy, "Stereotyping: Improving particle swarm performance with cluster analysis," in *Proc. Congr. Evol. Comput.*, 2000, pp. 1507–1512.
[27] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.
[28] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proc. Congr. Evol. Comput.*, 2002, pp. 1671–1676.
[29] T. Krink, J. Vesterstrom, and J. Rigel, "Particle swarm optimization with spatial particle extension," in *Proc. Congr. Evol. Comput.*, 2002, pp. 1474–1479.
[30] X. Li, "Adaptively choosing neighborhood bests using species in a particle swarm optimizer for multimodal function optimization," in *Proc. Genetic Evol. Comput. Conf.*, vol. 3102, K. Deb *et al.*, Eds., 2004, pp. 105–116.
[31] X. Li and K. H. Dam, "Comparing particle swarms for tracking extrema in dynamic environments," in *Proc. Congr. Evol. Comput.*, 2003, pp. 1772–1779.
[32] R. Morrison, *Designing Evolutionary Algorithms for Dynamic Environments*. Berlin, Germany: Springer-Verlag, 2004.
[33] D. Parrott and X. Li, "A particle swarm model for tracking multiple peaks in a dynamic environment using speciation," in *Proc. Congr. Evol. Comput.*, 2004, pp. 98–103.
[34] K. E. Parsopoulos and M. N. Vrahatis, "On the computation of all global minimizers through particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 211–224, Jun. 2004.
[35] K. Parsopoulos and M. Vrahatis, "Recent approaches to global optimization problems through particle swarm optimization," *Natural Comput.*, vol. 1, no. 2–3, pp. 235–306, 2002.
[36] A. Petrowski, "A clearing procedure its a niching method for genetic algorithms," in *Proc. Int. Conf. Evol. Comput.*, J. Grefenstette, Ed., 1996, pp. 798–803.
[37] Y. Shi and R. A. Krohling, "Co-evolutionary particle swarm optimization to solve min-max problems," in *Proc. Congr. Evol. Comput.*, 2002, pp. 1682–1687.
[38] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.
[39] X. Zhang, L. Yu, Y. Zheng, Y. Shen, G. Zhou, L. Chen, L. Xi, T. Yuan, J. Zhang, and B. Yang, "Two-stage adaptive PMD compensation in a 10 Gbit/s optical communication system using particle swarm optimization algorithm," *Optics Commun.*, vol. 231, pp. 233–242, 2004.

Tim Blackwell graduated with the Degree in physics from Bristol University, Bristol, U.K., in 1980 and the Ph.D. degree from Sussex University, Brighton, U.K., in 1984, for research in discrete quantum field theory, and the M.Sc. degree in computing from University College, London, U.K.

He moved to London in 2000. His M.Sc. thesis on the application of swarms to music led to a number of papers and he returned to research in 2003 with a Lectureship at Goldsmiths College. He was a Research Assistant at Imperial College London and the University of Glasgow before leaving academia to spend time developing musical interests. He is the Goldsmiths Principal Investigator for the EPSRC funded Extended Particle Swarms Project. He is also the Principal Investigator for the EPSRC Live Algorithms for Music Research Network; this network of computer and cognitive scientists and musicians is investigating autonomous music machines. Needless to say, it is expected that some of these machines will exploit the intelligence of swarms.

Jürgen Branke received the Diploma and the Ph.D. degree from the University of Karlsruhe, Karlsruhe, Germany, in 1994 and 2000, respectively.

He has been active in the area of nature-inspired optimization since 1994. After receiving the Ph.D. degree, he worked as a Scientist at Icosystem, Inc., before he returned to the University of Karlsruhe, where he currently holds the position of a Research Associate. He has written the first book on evolutionary optimization in dynamic environments and has published numerous articles in the area of nature-inspired optimization applied in the presence of uncertainty, including noisy and dynamic environments and the search for robust solutions. His research interests also include the design and optimization of complex systems, multiobjective optimization, parallelization, and agent-based modeling.