Particle Swarms for Dynamic Optimization Problems

Tim Blackwell[1], Jürgen Branke[2], and Xiaodong Li[3]

[1] Department of Computing
Goldsmiths College, London, UK
t.blackwell@gold.ac.uk
[2] Institute AIFB
University of Karlsruhe, Karlsruhe, Germany
branke@aifb.uni-karlsruhe.de
[3] School of Computer Science and Information Technology
RMIT University, Melbourne, Australia
xiaodong.li@rmit.edu.au

1 Introduction

Many practical optimization problems are dynamic in the sense that the best solution changes in time. An optimization algorithm, therefore, has to both find and subsequently track the changing optimum. Examples include the arrival of new jobs in scheduling, changing expected profits in portfolio optimization, and fluctuating demand.

Clearly, if the changes in the problem instance are radical, the best one can do is to repeatedly solve the optimization problem from scratch. However, in most practical applications the changes are gradual. If this is the case, it should be possible to speed up optimization after a problem change by utilizing some of the information on the fitness landscape gathered during the optimization process so far. In recent years, appropriately modified evolutionary algorithms (EAs) have been shown to achieve this successfully; see, e.g., [11, 23]; the focus of this chapter is to present similar advances within the context of particle swarm optimization.

Particle swarm optimization (PSO) is a versatile population-based optimization technique, in many respects similar to evolutionary algorithms. Basically, particles "fly" above the fitness landscape, while a particle's movement is influenced by its attraction to its neighborhood best (the best solution found by members of the particle's social network), and its personal best (the best solution the particle has found so far). PSO has been shown to perform well for many static problems [32], and is introduced in more detail in Section 2.

The application of PSO to dynamic problems has been explored by various authors [9, 7, 14, 17, 19, 20, 21, 32, 29, 38]. The overall consequence of

this work is that PSO, just like EAs, must be modified for optimal results in dynamic environments. There are two main difficulties that need to be addressed:

1. **Outdated memory:** If the problem changes, a previously good solution stored as neighborhood or personal best may no longer be good, and will mislead the swarm towards false optima if the memory is not updated.
2. **Diversity loss:** In normal operation, the swarm contracts around the best solution found during the optimization. As has been demonstrated, the time taken for a partially converged swarm to re-diversify, find the shifted peak, and then re-converge is quite deleterious to performance [4].

A number of adaptations have been applied to PSO in order to solve these difficulties; memories can be refreshed or forgotten and swarms may be re-diversified through randomization, repulsion, and dynamic information exchange and with the use of multi-populations. An account of these adaptations, and a summary of how PSO can detect change (this is especially important when change is unpredictable), is presented in Section 3. A more detailed review of our own work on PSO algorithms, the species PSO and the multi-swarm PSO, is described and extended in Section 4. These approaches are empirically tested and compared in Section 5. The chapter concludes with a summary and some ideas for future work.

2 Particle Swarm Optimization

Optimization with particle swarms (see Chap. 2 for a detailed introduction) has two major ingredients, the particle dynamics and the particle information network. The particle dynamics are derived from swarm simulations in computer graphics [34], and the information sharing component is inspired by social networks [18, 25]. These ingredients make PSO a robust and efficient optimizer of real-valued objective functions (although PSO has also been successfully applied to combinatorial and discrete problems). PSO is an accepted computational intelligence technique, sharing some qualities with evolutionary computation [1]. For an introduction to PSO see also Chap. 2 of this book.

In PSO, population members (particles) move over the search space according to

$$\mathbf{v}(t+1) = \mathbf{v}(t) + \mathbf{a}(t+1) \tag{1}$$

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \mathbf{v}(t+1) \tag{2}$$

where $\mathbf{a}$, $\mathbf{v}$, $\mathbf{x}$, and t are acceleration, velocity, position and time (iteration counter) respectively.

A particle's acceleration $\mathbf{a}$ is primarily governed by attraction to two solutions: its personal best and its neighborhood best. Particles possess a memory of the best (with respect to an objective function) location that they

have visited in the past, their *personal best* or *pbest*, and of its fitness. In addition, particles have access to the best location of any other particle in their neighborhood, usually denoted as *neighborhood best* or *gbest*. Naturally, these two locations will coincide if the particle has the best local best in its neighborhood. Several neighborhood topologies have been tried, with the fully connected network remaining a popular choice for unimodal problems. With this neighborhood structure, every particle will share information with every other particle in the swarm so that there is a single *gbest* global best attractor representing the best location found by the entire swarm.

The particles experience a linear or spring-like attraction, weighted by a random number, towards each attractor. Convergence towards a good solution will not follow from these dynamics alone; the particle movement must progressively contract. This contraction is implemented by Clerc and Kennedy with a constriction factor χ, $\chi < 1$, [16]. For our purposes here, the Clerc-Kennedy PSO will be taken as the canonical swarm; χ replaces other energy draining factors such as decreasing 'inertial weight' and velocity clamping.

Overall, the acceleration of particle i in Eq.1 is given by

$$\mathbf{a}_i = \chi[c\boldsymbol{\epsilon_1} \cdot (\mathbf{p}_g - \mathbf{x}_i) + c\boldsymbol{\epsilon_2} \cdot (\mathbf{p}_i - \mathbf{x}_i)] - (1 - \chi)\mathbf{v}_i \tag{3}$$

where $\boldsymbol{\epsilon_1}$ and $\boldsymbol{\epsilon_2}$ are vectors of random numbers drawn from the uniform distribution $\mathcal{U}[0, 1]$, $c > 2$ is the spring constant and $\mathbf{p}_i$, $\mathbf{p}_g$ are personal and neighborhood attractors. This formulation of the particle dynamics emphasizes constriction as a frictional force, opposite in direction and proportional to velocity. Clerc and Kennedy derive a relation for $\chi(c)$: standard values are $c = 2.05$ and $\chi = 0.729843788$. The complete PSO algorithm for maximizing an objective function f is summarized as Algorithm 2.

3 Addressing the Challenges in Dynamic Environments

As has been mentioned in Section 1, in dynamic environments, PSO suffers from outdated memory and lost diversity. This section summarizes the approaches proposed in the literature to address these challenges. Because many of these approaches explicitly react to a change in the landscape, we start by discussing ways to detect a change.

3.1 Detecting a Change

In many applications, a change is known to the system, e.g., in scheduling, when a new job arrives and has to be integrated into the schedule. If the time of a change is not known, it has to be detected. In the literature, this is usually done by simply re-evaluating one or more solutions, and concluding that a change has occurred if the fitness value of at least one of these solutions has changed [14, 20]. How many solutions and which ones to re-evaluate

Algorithm 2 Canonical PSO

```
FOR EACH particle i
    Randomly initialize v_i, x_i = p_i
    Evaluate f(p_i)
    g = arg max f(p_i)
REPEAT
  FOR EACH particle i
      Update particle position x_i according to Eqs. 1, 2 and 3
      Evaluate f(x_i)
      //Update personal best
      IF f(x_i) > f(p_i) THEN
          p_i = x_i
      //Update global best
      IF f(x_i) > f(p_g) THEN
          p_g = arg max f(p_i)
UNTIL termination criterion reached
```

may depend on the particular application. Usually, the best solution found so far is re-evaluated, which prevents the algorithm from converging around a previously good solution which is no longer good.

In [22], a change in the environment is detected from observing the algorithm behavior, which has the advantage of also working in noisy environments when the above re-evaluation scheme might lead to false alarms.

3.2 Memory Update

If the environment changes, the particle memory (namely the best location visited in the past, and its corresponding fitness) may no longer be valid, with potentially disastrous effects on the search.

This problem is typically solved in one of two ways: re-evaluating the memory or forgetting the memory [14]. In the latter, each particle's memory is simply set to the particle's current position, and the global best is updated making sure that $\mathbf{p}_g = \arg\max f(\mathbf{p}_i)$.

3.3 The Problem of Lost Diversity

If the environment changes after the swarm has converged to a peak, it takes time for the population to re-diversify and re-converge, making it slow in tracking a moving optimum.

The diversity loss was examined in [3] based on the swarm diameter $|S|$, defined as the largest distance, along any axis, between any two particles. When a change occurs and the new optimum location is within the collapsing swarm, there is a good chance that a particle will find itself close to the new optimum within a few iterations and the swarm will successfully track the

moving target. The swarm as a whole has sufficient diversity. However, if the new optimum is outside the swarm's expansion, the low velocities of the particles (which are of order $|S|$) will inhibit re-diversification and tracking, and the swarm can even oscillate about a false attractor and along a line perpendicular to the true optimum, in a phenomenon known as linear collapse [6]. [4] uses these considerations to examine under what conditions a swarm can track a single moving optimum.

Because of the problem of convergence, either a diversity increasing mechanism should be invoked at change (or at predetermined intervals), or sufficient diversity has to be ensured at all times [8]. There are four principle mechanisms proposed in the literature for either re-diversification or diversity maintenance: randomization [20], repulsion [6], dynamic networks [21, 37] and multi-populations [30, 7]. They will be discussed in turn in the following.

3.4 Re-diversification

Hu and Eberhart [20] study a number of re-diversification mechanisms. They all involve randomization of the entire or part of the swarm after a change. Since randomization implies information loss, there is a danger of erasing too much information and effectively re-starting the swarm. On the other hand, too little randomization might not introduce enough diversity to cope with the change. In the multi-swarm approaches (see below), it has been suggested to always keep one swarm searching, and randomize the least-fit swarm whenever all swarms have converged [8]. The way quantum particles are used in this chapter, as described in Section 4.1, can also be seen as a form of local re-diversification.

3.5 Repulsion

A constant degree of swarm diversity can be maintained at all times through some type of repulsive mechanism. Repulsion can be either between particles, or from an already-detected optimum. For example, Krink et al. [36] study finite-size particles as a means of preventing premature convergence. The hard sphere collisions produce a constant diversification pressure. Alternatively, Parsopoulos and Vrahatis [32] place a repeller at an already-detected optimum, in an attempt to divert the swarm and find new optima. Neither technique, however, has been applied to the dynamic scenario.

An example of repulsion that has been tested in a dynamic context is the atom analogy [6, 5, 9, 7]. In this model, a swarm is comprised of 'charged' and 'neutral' particles. The charged particles repel each other, leading to a cloud of charged particles orbiting a contracting, neutral, PSO nucleus (as shown in Figure 1). Charge enhances diversity in the vicinity of the converging PSO subswarm, so that optimum shifts within this cloud should be traceable. Good tracking (outperforming canonical PSO) has been demonstrated for unimodal dynamic environments of varying severities [4].

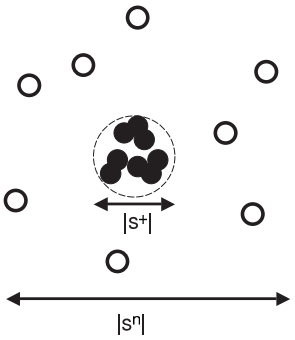


Fig. 1. An example of a swarm containing both neutral and charged particles. A solid circle denotes a neutral particle, whereas a hollow circle denotes a charged particle. $|S^+|$ and $|S^n|$ denote the size of the charged and neutral swarms, respectively.

In [7, 8], the charged particle idea has been simplified to the quantum particle, which does not follow the usual particle movement laws, but instead is re-generated in each iteration at a random position in a vicinity around the swarm's global best. Quantum particles have been shown to be easier to control, to be computationally faster and to perform better than charged particles in [7, 8]. More details on quantum particles are discussed in Section 4.1.

3.6 Dynamic Network Topology

Adjustments to the information-sharing topology can be made with the intention of reducing, maybe temporarily, the desire to move towards the global best position, thereby enhancing population diversity. Li and Dam [37] use a grid-like neighborhood structure, while Janson and Middendorf [21] apply a tree-like structure. Both papers report improvements over unmodified PSO for unimodal dynamic environments. In the latter approach, the particles can change places in the hierarchy, In [22], it has been additionally suggested to break up the tree into sub-trees after a change, so that they can independently search for a new optimum for a while, until they are joined again.

A division of the swarm into subswarms is intuitively helpful in dynamic multi-modal environments, where several promising regions of the search space can be tracked simultaneously. This is the core idea of the speciation PSO (SPSO) proposed in [27, 30], and the multi-swarm PSO (MPSO) proposed in [7, 8]. SPSO uses a fixed swarm size and dynamically divides the swarm into subswarms based on a technique known as clearing [33]. MPSO, on the other hand, has a set of swarms of predetermined size, and adjusts the number of such swarms during the run, thereby also changing the overall population size. These two approaches are examined in more detail below.

4 Multi-swarms and Speciation

The authors of this chapter have independently proposed two different approaches which divide the swarm into a number of subswarms: the multi-swarm PSO (MPSO, [7, 8]) and the speciation-based PSO (SPSO, [27, 31, 28]). The motivation for both approaches is that in a dynamic environment, it is useful to maintain information about several promising regions of the search space. This proposition was already the motivation behind the self-organizing scouts approach [11] and has recently been confirmed also in [12]. By dividing up the swarm, the subswarms may simultaneously track different promising regions of the search space. This is particularly helpful if the environment changes in a way that makes a previous local optimum the new global optimum. If one of the subswarms was tracking the local optimum or a nearby region, the new global optimum is immediately found.

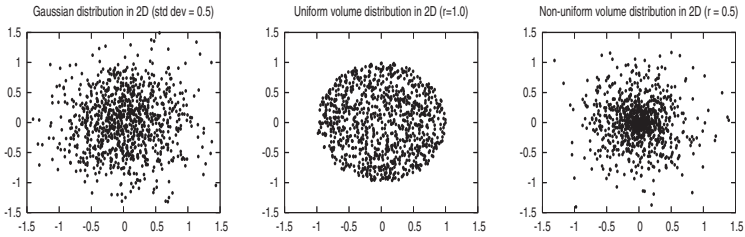
In this section, we describe the previously proposed MPSO and SPSO in detail and also present some new extensions. The approaches are then compared empirically in Section 5. Because both approaches now use quantum particles, these are discussed first.

4.1 Quantum Particles

The quantum particles have been proposed in [7] as a means to maintaining a certain level of diversity within a swarm. They have been inspired by atomic models. In a classical atom, a number of electrons orbit, at various distances, a small ball of nucleons. The picture is altered in the quantum atom; the electrons do not orbit in deterministic paths but are distributed in a probability 'cloud' around the nucleus. The PSO atom consists of a nucleus of normal PSO particles moving under the normal update rules. Typically this nucleus will be shrinking in size as it converges on an optimum. The nucleus is surrounded by quantum particles. Quantum particles do not follow PSO dynamics but are placed at positions around the center of the nucleus, defined by $\mathbf{p}_g$, according to a probability distribution. As a result, they do not converge, but maintain a constant level of diversity.

Different probability distributions are conceivable. It is reasonable to assume that the quantum probability distribution should be spherically symmetric, i.e., shells of constant density centered on $\mathbf{p}_g$. In the following, three different distributions are considered: The Gaussian distribution, the uniform volume distribution (UVD), and a non-uniform volume distribution (NUVD); see Figure 2 for examples in a two-dimensional space. The UVD in d dimensions can be generated as follows [15]:

1. Generate a point $\mathbf{x_i}$ from $\mathcal{N}[0,1]$ for $1 \leq i \leq d$.
2. Calculate the distance of x_i to the origin $dist = \sqrt{\sum_{i=1}^{i=d} x_i^2}$.
3. Select a value u from $\mathcal{U}[0,1]$.
4. The new point will be $r_{cloud} \cdot \mathbf{x_i} \frac{\sqrt[d]{u}}{dist}$.



(a) Gaussian (b) UVD (c) NUVD

Fig. 2. 1,000 sampling points for a Gaussian distribution, a uniform volume distribution (UVD), and a non-uniform volume distribution (NUVD), in two dimensions.

r_{cloud} is a parameter defining the radius of the distribution.

The above procedure can be modified to generate other distributions by simply changing the distribution for u. For example, instead of selecting u from $\mathcal{U}[0,1]$ in Step 3, Gaussian $\mathcal{N}[0,\sigma]$ can be used with σ set to roughly $\frac{1}{3}$, since in a Gaussian distribution with 3σ away from the mean would cover 99% of all possible samples. This would create a distribution with higher density closer to mean. When changing the calculation of the new solution in Step 4 to $r_{cloud} \cdot \mathbf{x_i} \frac{u}{dist}$, one obtains the non-uniform volume distribution (NUVD) where the density decreases linearly with distance from the center.

Quantum particles are somehow related to the bare-bones PSO proposed by Kennedy, where each dimension of the new position of a particle is randomly selected from a Gaussian distribution with the mean being the average of $\mathbf{p_i}$ and $\mathbf{p_g}$ and the standard deviation σ being the distance between $\mathbf{p_i}$ and $\mathbf{p_g}$ [24]:

$$\mathbf{x_i} \leftarrow \mathcal{N}(\frac{\mathbf{p_i} + \mathbf{p_g}}{2}, ||\mathbf{p_i} - \mathbf{p_g}||) \tag{4}$$

Richer and Blackwell also reported work on PSO variants employing Gaussian distribution [35], as well as the more general Lévy distribution[4]. Algorithms employing a Lévy or Cauchy distribution, which both have a long fat tail, are more capable of escaping local optima than the Gaussian counterpart. Escape from local optima is profitable in circumstances in which a single global optimum must be found. In the context of tracking moving peaks in a multi-modal dynamic environment, we are more interested in finding multiple peaks in parallel so that when peaks move the optimizer still has a chance to relocate them. Distributions that can provide good sampling in an adjacent area of the peaks would be more suitable; hence the Gaussian distribution is preferred.

[4] The shape of the Lévy distribution can be controlled by a parameter α. For $\alpha = 2$ it is equivalent to Gaussian distribution, whereas for $\alpha = 1$ it is equivalent to the Cauchy distribution [35].

4.2 Multi-swarm PSO

The multi-swarm PSO (MPSO) was originally proposed in [7] and then extended in [8] and [2]. It maintains diversity on two levels: the swarm is divided into a number of subswarms which are forced to different areas of the search space (diversity between swarms), and each swarm has some quantum particles to ensure diversity within the swarm.

Basic Features

The MPSO proposed in [8] uses the following mechanisms:

- **Change Detection and Outdated Memory:** For change detection, in each iteration, a subswarm's global best is re-evaluated. If the fitness value has changed, a change is detected and all of the subswarm's personal best are re-evaluated before commencing.
- **Multiple Swarms:** The particles in MPSO are divided into a number of M independent subswarms. Each subswarm in MPSO has a fixed number of particles. Information sharing within a swarm is global, i.e., any good position found by any particle (neutral or quantum) is available to any other. It is known that a global information topology between particles produces better optimization of a uni-modal environment, whereas a local topology is preferred in the multi-modal situation. Even if the landscape is multi-modal, the role of the neutral PSO is to climb up a single peak. The diversity needed to find peaks stems from a dynamic interaction between separate swarms (exclusion, see below) rather than information transfer between social neighborhoods of a single swarm.
- **Exclusion:** Intuitively, several swarms sitting on the same local optimum are not very helpful; they should explore different promising regions of the search space. To this end, a so-called *exclusion* operator is employed. Swarm exclusion forbids two swarms moving to within r_{excl} of each other, where the distance between swarms is defined as the distance between their $\mathbf{p}_g$'s. When exclusion is invoked, the worse swarm, as judged by the current best values $f(\mathbf{p}_g)$, is randomized in the entire search space.
- **Anti-convergence:** Exclusion makes sure that the different swarms converge to different local optima. In order to be able to detect also new, emerging peaks, MPSO contains an additional mechanism termed *anti-convergence* in [8]: Whenever all swarms have converged, the least-fit swarm (as judged again by the current best values $f(\mathbf{p}_g)$) is randomized in the entire search space. Thereby, a swarm is considered as converged when its expansion, i.e., the radius of the smallest circle encompassing the neutral particles, is less than r_{conv}. Anti-convergence is particularly important if the number of swarms is significantly smaller than the number of local optima, in which case each swarm might converge to and get stuck on a different local optimum.

- **Quantum Particles:** Each swarm in MPSO consists of a number N^0 of neutral particles and a number N^Q of quantum particles. The quantum particles are generated in each iteration according to a UVD distribution with parameter r_{cloud} around the swarm's p_g.

Parameter Settings

MPSO as described so far introduces a number of new parameters: the number M of swarms, the number of quantum particles N^Q, the exclusion distance r_{excl}, and the quantum cloud radius r_{cloud}. Several guidelines for setting these parameters are provided in [8] and shall be briefly summarized here. Intuitively, there is a relationship between the distance a local optimum shifts, s, and the quantum cloud parameter r_{cloud}. It is expected that these factors are of the same order of magnitude so that a quantum particle might be found close to the new optimum. Previous experiments have shown that $r_{cloud} = 0.5s$ is a good default setting. r_{excl} can be estimated by assuming that all p peaks occupy the same portion of the search space X^d. The radius r_{boa} of the basin of attraction of a peak is then $p \cdot r_{boa}^d = X^d$, or $r_{boa} = X/p^{1/d}$. The exclusion radius r_{excl} is thus set to r_{boa}. The multi-swarm cardinality M can be estimated from the number of optima, p. If possible we would expect that $M > p$ is undesirable since free swarms absorb valuable function evaluations and there is no need to have many more swarms than peaks. Similarly, many fewer swarms than peaks means the multi-swarm is in danger of missing good locations.

Self-adaptation of the Number of Swarms

Because usually the number of peaks is not known beforehand, the number of swarms, M, might be difficult to set. For this reason, in [2], a self-adaptation mechanism for the number of swarms has been proposed. The mechanism for this is quite simple. The multi-swarm will need a new, patrolling swarm, if all current swarms are converging. On the other hand, too many free swarms will absorb function evaluations and one should be removed. If there is more than one free swarm, the choice for removal is arbitrary and the worst of the free swarms is removed, as judged by $f(\mathbf{p}_g)$. This birth/death mechanism removes the need for the anti-convergence operator, and for specifying the multi-swarm cardinality M.

The number of swarms $M(t)$ is then dynamic and at any iteration t given by

$$M(t) = \begin{cases} M(0) = 1 \\ M(t-1) + 1, & M_{free} = 0 \\ M(t-1) - 1, & M_{free} > n_{excess} \end{cases} \tag{5}$$

where n_{excess} is a parameter specifying the desired number of free swarms, and a free swarm is a swarm whose expansion is larger than r_{conv}. An intuitive

choice is to allow just a single free swarm, $n_{excess} = 1$, as one swarm roaming around and searching for new peaks should be sufficient. Setting $n_{excess} = \infty$ means that no swarm can ever be removed and the multi-swarm can only grow.

When the number of swarms is adapted, r_{excl} can also be adapted by assuming that M corresponds to the number of peaks:

$$r_{excl} = \frac{X}{2M^{1/d}} \tag{6}$$

where X is the extent of the search space in each dimension.

Note that this adaptation scheme, by changing the number of swarms, also changes the overall number of particles. In practice, the number of swarms should be bounded. Too many particles will slow down the PSO, as each particle is processed less frequently. But in the empirical tests reported below, no such bound was used.

Particle Conversion

Finally, let us propose another modification of the original MPSO. In the original MPSO, each population has a fixed number of neutral and quantum particles. Intuitively, quantum particles are most useful at or just after an environmental change, where they provide the tracking that a tightly converged swarm cannot perform. Their role during environmentally stable periods is less clear. Thus, we propose here to convert all neutral particles to quantum particles for one iteration immediately after a change has been detected. After this iteration, they are reverted back to neutral. We expect that this mechanism might allow us to reduce the number of permanent quantum particles in a population.

The overall algorithm is summarized in Algorithm 3.

4.3 Speciation-Based PSO

Other than the just-discussed MPSO which uses fixed swarms with a fixed number of particles each, the speciation-based PSO (SPSO) is able to dynamically distribute particles to species. It was inspired by a clearing procedure proposed in [33]. SPSO was developed based on the notion of species [27]. The definition of species depends on a 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 best-fit particle in the species. All particles that fall within distance r_s from the species seed are classified as the same species.

Algorithm 4 summarizes the steps for determining species seeds [26]. By performing this algorithm at each iteration step, each different species seed can be identified for a different species and the seed's $\mathbf{p_i}$ can be used as the $\mathbf{p_g}$ for particles belonging to that species accordingly [27, 31].

Algorithm 3 Multi-Swarm

//Initialization
  Begin with a single free swarm, $M = 1$
FOR EACH particle ni
  Randomly initialize $\mathbf{v}_{ni}, \mathbf{x}_{ni} = \mathbf{p}_{ni}$
  Evaluate $f(\mathbf{p}_{ni})$
FOR EACH swarm n
  $\mathbf{p}_{ng} := argmax\{f(\mathbf{p}_{ni})\}$

REPEAT
  // Adapt number of swarms
  IF all swarms have converged THEN
    Generate a new swarm.
  ELSE IF ($M_{free} > n_{excess}$) THEN
    Remove worst swarm.
  FOR EACH swarm n
    // Test for Change
    Evaluate $f(\mathbf{p}_{ng})$.
    IF new value is different from last iteration THEN
      Convert all particles to quantum particles for one iteration.
      Re-evaluate each particle attractor.
      Update swarm attractor.
    FOR EACH particle i of swarm n
      // Update Particle
      Move particle depending on particle type.
      // Update Attractor
      Evaluate $f(\mathbf{x}_{ni})$.
      IF $f(\mathbf{x}_{ni}) > f(\mathbf{p}_{ni})$ THEN
        $\mathbf{p}_{ni} := \mathbf{x}_{ni}$.
      IF $f(\mathbf{x}_{ni}) > f(\mathbf{p}_{ng})$ THEN
        $\mathbf{p}_{ng} := \mathbf{x}_{ni}$
    // Exclusion.
    FOR EACH swarm $m \neq n$
      IF swarm attractor p_{ng} is within r_{excl} of p_{mg} THEN
        IF $f(\mathbf{p}_{ng}) \leq f(\mathbf{p}_{mg})$ THEN
          Re-initialize swarm n
        ELSE
          Re-initialize swarm m
        FOR EACH particle in re-initialized swarm
          Re-evaluate function value.
          Update swarm attractor.
UNTIL number of function evaluations performed $> max$

Algorithm 4 Algorithm for determining species seeds

```
//L_sorted - a list of particles sorted in their decreasing f(p_i) values
//S - a list of dominating particles identified as species seeds
S = ∅
REPEAT
    Get the best unprocessed p ∈ L_sorted
    found ← FALSE
    FOR all s ∈ S
        IF d(s, p) ≤ r_s THEN
            found ← TRUE
            break
        IF notfound THEN
            let S ← S ∪ {p}
UNTIL reaching the end of L_sorted
```

Basically Algorithm 4 sorts all particles in decreasing order of the fitness values of their personal bests $\mathbf{p_i}$[5]. The species seed set S is initially set to $\emptyset$. All particles' $\mathbf{p_i}$ are checked in turn (from best to least fit) against the species seeds found so far. If a particle's $\mathbf{p_i}$ 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. Figure 3 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 less fit seeds in the list L_{sorted}. For example, s_2 dominates s_3; therefore p should belong to the species led by s_2. This has the nice side-effect of helping SPSO to locate the fitter peaks before the less fit ones.

The identified species seeds represent particles that are highly fit and at least distance r_s away from each other. Since a species seed is the best-fit particle's $\mathbf{p_i}$ within a species, all particles belonging to the same species can be made to follow their species seed's $\mathbf{p_i}$ as their leader (i.e., neighborhood best). Each species acts as a separate PSO with all its particles moving according to the conventional particle velocity rules. The sorting of particles, determination of species seeds, and allocation of species seeds as leaders to particles are performed at each iteration, which have the effect of moving particles within the same species to positions that make them even fitter. Because species are formed around different peaks in parallel, species seeds will provide the right guidance for particles to converge towards different peaks that exist on the fitness landscape. Comparing with the multi-swarm concept introduced in the earlier sections, a species in SPSO is equivalent to an individual swarm of the

[5] Note that in our previous implementation, the fitness values of current particle positions $\mathbf{x_i}$ were sorted [27, 28]. We changed to use particle's personal best $\mathbf{p_i}$ because it is a more stable point compared with $\mathbf{x_i}$

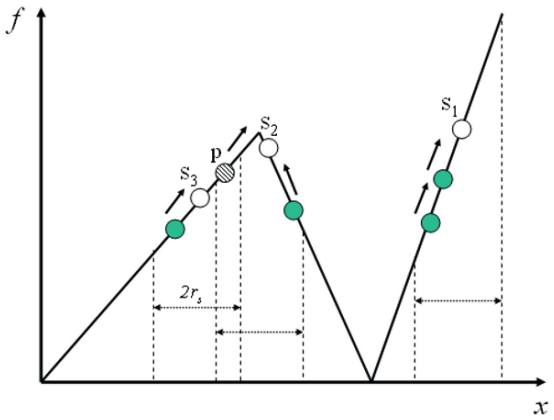


Fig. 3. An example of determining species seeds from a population of particles. s_1, s_2, and s_3 are chosen as the species seeds; for the other particles, arrows indicate which species seed they are attracted to. Note that p falls within the radius r_s of two seeds but follows s_2 as the seed which was identified earlier.

multi-swarm model, but the particles are re-distributed to species dynamically in each iteration, resulting in a variable number of species with different sizes.

To detect whether a change has occurred, SPSO simply re-evaluates the recorded personal best positions of the top t best species seeds at each iteration. A change is considered to have occurred if any of these t re-evaluations is different from its corresponding personal best's recorded fitness. All particles' personal bests are then reset to their associated current positions since these recorded personal bests are outdated.

Species Cap, Quantum Swarm, and Anti-convergence

This section describes several useful techniques incorporated into SPSO to enhance its performance in dynamic environments [28].

Since the algorithm for identifying species seeds favors those seeds with higher fitness values resulting in more particles being allocated to fitter species than to less fit ones, on a multi-modal fitness landscape this may result in too many particles assigned to just a few very best peaks while leaving other lower peaks unoccupied. In order to distribute more evenly the number of particles across different species, a parameter p_{max} can be used to set a maximum number of particles that a species is allowed [30]. This means that only the best-fit p_{max} particles will be allocated as members of a species. Least-fit members that cause the species population to exceed p_{max} are reinitialized as randomly generated new particles into the search space, as a side-effect helping SPSO better explore the search space.

The quantum atom model described in Section 4.1 is also adopted in SPSO, but as described in [28], only triggered by convergence: When the neutral par-

ticles belonging to a species converge below a pre-specified threshold (largest distance between any pair of particles), half of the neutral particles are converted to quantum particles around the species seed to form a 'quantum cloud'.

Anti-convergence was first proposed and used as an effective mechanism for global information sharing in the multi-swarm model [8] (see also Section 4.2). This idea can be adopted in SPSO to reduce the number of less fit species [28] and to improve information sharing among species. Anti-convergence is carried out simply by replacing the least-fit species and reinitializing them into the search space at each iteration step. While this seems drastic, note that the speciation procedure usually results in some isolated particles forming species of size 1. Anti-convergence as implemented here randomizes these until they either discover a promising region or are close enough to join a larger species.

4.4 Improving Local Convergence

SPSO is shown to be an effective optimizer for solving static multi-modal problems [31], and with enhancements described in the previous section, it can be used for handling multi-modal problems in a dynamic environment [28]. In order to track moving peaks, SPSO must be able to locate peaks and follow them as closely as possible if they have moved. It is observed that SPSO can consistently locate the majority of the peaks most of the time. However, relocating moved peaks with a satisfactory convergence speed (so as to reduce the offline error) still remains as a challenge. Techniques promoting faster local convergence would be desirable to tackle this problem.

The convergence-triggered strategy as described above and in [28] keeps the particles in a species spread out so that the species will have a better chance to recapture the peak if it has moved again. The downside is that these quantum particles contribute little to the local convergence of the species most of the time. Although changes occur only occasionally, at each iteration quantum particles are generated to form a 'cloud' spreading out around the species seed, rather than being used to converge towards the species attractor, like the neutral particles.

In this study, instead of using the usual quantum 'cloud' approach, for one iteration after a change has been detected, all particles are moved as quantum particles, i.e., to a randomly selected position according to a given distribution. Also, we found that centering the distribution not around the species' seed, but around the center of species seed and particle position, has slight advantages. After this, all particles will move again as neutral particles according to their allocated seed and personal best positions, following the standard PSO velocity update rules. This new variant of SPSO is summarized in Algorithm 5.

Algorithm 5 SPSO with local sampling

//Initialization
FOR EACH particle i
    Randomly initialize $\mathbf{v}_i, \mathbf{x}_i$, $\mathbf{p}_i = \mathbf{x}_i$
REPEAT
    FOR EACH particle i
        Evaluate $f(\mathbf{x_i})$
        //Update personal best
        IF $f(\mathbf{x_i}) > f(\mathbf{p_i})$ THEN
            $\mathbf{p_i} \leftarrow \mathbf{x_i}$
    IF change is detected THEN
        $\mathbf{p_i} \leftarrow \mathbf{x_i}$
    //Following steps are carried out at each iteration
    Sort particles in descending order of their $\mathbf{p_i}$ fitness values
    Identify species seeds from the above sorted list based on r_s
    $\mathbf{p_i}$ of each species seed is assigned as the leader (neighborhood best) to all particles belonging to the same species
    IF $numParticles > p_{max}$ THEN
        Anti-convergence to replace the excess particles with new particles
    Replace particles in the least-fit species by initializing them
    Adjust all particle positions according to Eqs. (1) and (2)
    //Invoke local sampling only if change is detected
    IF change is detected THEN
        Generate a new particle for each particle by local sampling
UNTIL number of function evaluations performed $> max$

5 Empirical Results

5.1 Moving Peaks Benchmark and Experimental Setup

For empirical tests, we used the publicly available moving peaks benchmark (MPB) [10]. It consists of p peaks changing in height and width, and moving by a fixed shift length s in random directions every K evaluations. The peaks are constrained to move in a search space of extent X in each of the d dimensions, $[0, X]^d$.

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 1. They correspond to Scenario 2 in [10] and have been used to facilitate comparison with other published results. The termination condition for each experiment is 100 peak changes, corresponding to 500,000 function evaluations.

Table 1. 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

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 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 [13] 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 than or equal to 0 and would be 0 for perfect tracking.

Unless specified otherwise, PSO acceleration parameters χ and c are set to standard values 0.72984 and 2.05, respectively. For MPSO, parameters were set according to the guidelines from Section 4.2, and n_{excess} was set to 1. For SPSO, the overall population size was set to 100, p_{max} was set to 10, and $t = 5$ best species seeds were re-evaluated to detect a change.

5.2 Optimal Swarm Size

We begin by determining the optimum neutral swarm size for a single stationary peak in five dimensions. Since canonical PSO does not use the local shape of the function, all spherically symmetric peaks are equivalent. Table 5.2 reports on tests of the neutral subswarm, which is a canonical PSO. The results demonstrate that a small, five-particle swarm is the best hill climber in 5 dimensions. We will therefore set the number of neutral particles in MPSO swarms to five.

5.3 Quantum Particles in MPSO

In previous experiments [8], it was shown that a swarm's particles should be divided equally into neutral and quantum particles performed, and swarms with five neutral and five quantum particles yielded best results. However, in Section 4.2, we have proposed converting the neutral particles to quantum particles for one iteration after a change has been detected, hoping this would allow us to reduce the number of quantum particles in a swarm.

Table 2. Performance of canonical PSO for a single cone, $f(\mathbf{x}) = |\mathbf{x}^* - \mathbf{x}|$. The table shows the best f attained after 2,500 evaluations, averaged over 100 runs with differing initial configurations and cone position

Number of particles	*f (std error)*
1	65.62 (2.37)
2	6.86(1.00)
3	0.0072 (0.0066)
4	5.43E-8 (4.17)
5	3.64E-10 (1.52E-10)
6	1.18E-9 (3.59E-10)
7	9.40E-9 (1.90E-9)
8	6.64E-8 (1.52E-8)
9	2.87E-7 (5.34E-8)
10	9.13E-7 (1.12E-7)

Table 5.3 shows results for MPSO on the MPB ($p = 10$, $s = 1.0$) for various swarm configurations. A configuration with a neutral particles and b quantum particles is denoted as $(a + b)$. As can be seen, even the $(5 + 0)$ configuration, which has no permanent quantum particles (only the converted particles at the iteration immediately following function change), performs remarkably well. The optimum configuration appears to have just one permanent quantum particle for both values of n_{excess}. Without particle conversion after a change, a $(5 + 1)$ MPSO obtains an offline error of only 2.05 (0.08). This confirms our hope that we might be able to reduce the overall number of particles due to the conversion method. It also demonstrates the importance of the exclusion operator which continuously repositions swarms until they find a peak to settle on. After this happens, swarm diversity is not required and the neutral particles will rapidly converge to the center of the peak. The presence of a small amount of diversity, i.e., just one quantum particle, presumably helps tracking in the few iterations just after the function change.

Comparing the results for $n_{excess} = 1$ and $n_{excess} = 3$, we see that differences are small, but slightly better results are obtained with three rather than one patrolling swarms. So, n_{excess} can be used for fine-tuning if necessary, but $n_{excess} = 1$ seems a good and intuitively justifiable default setting. This was also confirmed in some additional tests with only one peak or 200 peaks, where $n_{excess} = 1$ performed slightly better than $n_{excess} = 3$.

5.4 Quantum Distribution in SPSO

The following experiments look at the influence of the quantum distribution in scenarios with $p = 10$ peaks and with shift severity $s \in \{1, 5\}$.

Table 5.4 provides the results on offline errors using Gaussian, UVD and NUVD distribution respectively. These results are also visualized in Figure 4. The best offline error for Gaussian distribution is 1.73, at $\sigma = 0.3$. The best

Table 3. Variation of average offline error with swarm configuration and excess parameter for MPSO. The data is for 50 runs of MPB, scenario 2, p = 10, and s = 1.0 and with 500,000 evaluations per run

Configuration	oe (std error), $n_{excess} = 1$	oe (std error), $n_{excess} = 3$
5 + 0	1.80 (0.08)	1.85 (0.07)
5 + 1	1.73 (0.08)	1.69 (0.07)
5 + 2	1.85 (0.09)	1.69 (0.06)
5 + 3	1.82 (0.09)	1.83 (0.07)
5 + 4	1.77 (0.07)	1.88 (0.07)
5 + 5	1.85 (0.09)	1.92 (0.08)

Table 4. Offline errors for sampling using UVD and NUVD distribution with radius r_{cloud}, and Gaussian with standard deviation σ (set to r_{cloud}). The shift severity value $s = 1.0$ and $p = 10$

r_{cloud}	Gaussian	UVD	NUVD
0.05	1.86 (0.06)	2.24 (0.07)	2.31 (0.07)
0.1	1.82 (0.06)	1.97 (0.07)	2.02 (0.07)
0.2	1.84 (0.07)	1.72 (0.06)	1.78 (0.04)
0.3	1.73 (0.05)	1.74 (0.07)	1.74 (0.08)
0.4	1.85 (0.06)	1.64 (0.06)	1.73 (0.06)
0.5	2.00 (0.08)	1.77 (0.06)	1.62 (0.05)
0.6	1.94 (0.06)	1.84 (0.08)	1.69 (0.06)
0.7	2.23 (0.06)	1.76 (0.06)	1.66 (0.05)
0.8	2.29 (0.07)	1.79 (0.06)	1.87 (0.07)
0.9	2.42 (0.06)	1.78 (0.05)	1.98 (0.06)
1.0	2.55 (0.06)	1.79 (0.06)	1.89 (0.05)
1.2	2.75 (0.07)	2.01 (0.07)	1.89 (0.05)
1.5	3.07 (0.09)	2.09 (0.06)	1.95 (0.06)
2.0	3.48 (0.08)	2.40 (0.09)	2.12 (0.06)

result for UVD is 1.64 at $r_{cloud} = 0.4$, and for NUVD is 1.62 at $r_{cloud} = 0.5$. The differences between the results of UVD and NUVD are insignificant, but both the best results for UVD (1.64) and NUVD (1.62) are better than the best for Gaussian (1.73). These results show that sampling more frequently closer to the mean is not always beneficial for the purpose of the quantum particles. The reason may be that their task is not local improvement (as, e.g., for mutations), but exploration. In any case, the results are are better than the 1.98 reported previously with the convergence-triggered particle diversification scheme [28].

The scenario with $s = 5.0$ should be more difficult to track than the $s = 1.0$ counterpart. Consequently, the performance values summarized in Table 5.4 report higher offline errors. Note that r_{cloud} values in the range [1.0, 5.0] were

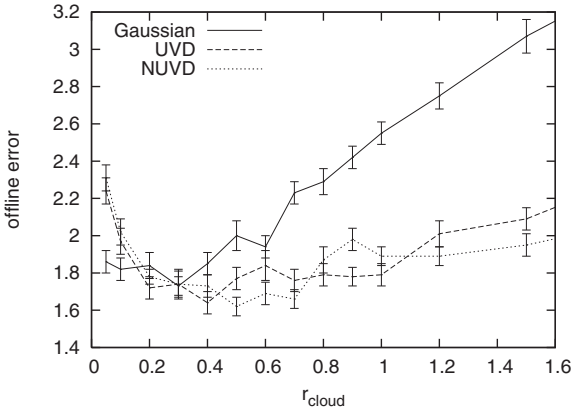


Fig. 4. Offline errors depending on radius r_{cloud} and distribution used for sampling.

Table 5. Offline errors for sampling using Gaussian, UVD, and NUVD distribution, on scenario 2. The shift severity $s = 5.0$ and $p = 10$.

r_{cloud}	Gaussian	UVD	NUVD
1.0	4.53(0.10)	4.43(0.10)	4.53(0.12)
1.5	4.62(0.09)	4.20(0.11)	4.36(0.08)
2.0	5.06(0.09)	4.15(0.08)	4.47(0.09)
2.5	5.54(0.11)	4.28(0.09)	4.56(0.10)
3.0	5.68(0.13)	4.43(0.09)	4.62(0.08)
3.5	6.29(0.11)	4.58(0.13)	4.79(0.11)
4.0	6.40(0.11)	4.61(0.12)	4.97(0.10)
4.5	7.15(0.13)	4.85(0.11)	4.86(0.11)
5.0	7.85(0.17)	4.93(0.12)	5.20(0.11)

used to reflect our assumption that $r_{cloud} \approx s/2$ should be a good default value, which is also confirmed with the empirical data.

Again, UVD performs better than NUVD and Gaussian. The best offline error was obtained by UVD with an offline error 4.15, at $r_{cloud} = 2.0$.

5.5 Adapting the Number of Swarms

Both MPSO and SPSO have mechanisms to adapt the number of swarms over time. While MPSO starts with a single swarm and adds additional swarms as needed, SPSO usually starts with many swarms, slowly converging to the required number of swarms to cover all peaks.

Fig. 5 shows a typical SPSO simulation run for an MPB problem with $p = 10$ and $s = 1$. As expected, SPSO starts with a large number of species seeds, but over iterations this number decreases to a value close to the number of existing peaks (if they are found). Anti-convergence is particularly effective

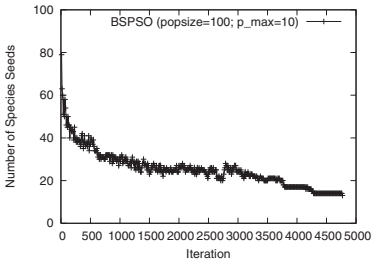


Fig. 5. The number of species seeds is decreasing over the run of a (basic) SPSO model on an MPB problem with ten peaks.

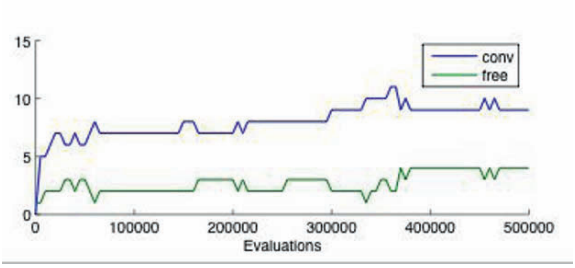


Fig. 6. The number of swarms in MPSO for a ten peak MPB problem.

during the early stage of the run in replacing the least-fit species with new particles that can be better used to explore other parts of the search space. Although it is possible that anti-convergence may remove a species that has already occupied a peak, the chance of this being the best-fit peak is small since the removed species is always the least-fit species. Furthermore, the best-fit peak is most likely watched and tracked by the fittest species.

The number of swarms in a typical MPSO run for a ten-peak MPB problem and with $n_{excess} = 3$ is shown in Figure 6. The figure displays both the number of converged swarms at any stage of the optimization, and the number of free swarms, where a swarm is deemed to be converging if its spatial extent is less than a convergence radius. This distance is dynamically determined by Eq. 6.

In the environment with 200 peaks and $s = 5$, SPSO with a population size of 100 is no longer able to cover all the peaks, and fluctuates between 20 and 30 species. MPSO, although potentially able to add an arbitrary number of subpopulations, converges only slightly higher to around 32 subpopulations. One reason for this convergence is certainly that of the 200 peaks, several smaller peaks are "covered" by higher peaks and thus not visible to the swarm, or they are too close to be regarded as separate peaks. Another reason may be that increasing the number of subpopulations also increases the total number of particles and thus slows down convergence, which leads to a slower convergence of swarms and thus fewer new swarms being spawned.

5.6 Comparing MPSO and SPSO

In this section, we compare MPSO and SPSO on four scenarios: shift severity $s \in \{1, 5\}$ and number of peaks $p \in \{10, 200\}$. Default parameters are used i.e., $r_{cloud} = 0.5s$ (SPSO, MPSO), $n_{excess} = 1$ (MPSO), $p_{max} = p$ (SPSO) and the UVD distribution (MPSO, SPSO). The exclusion radius and number of swarms (MPSO) are set dynamically according to Eqs. 5 and 6.

Table 6. Comparison of MPSO and SPSO on four test scenarios. Offline error and standard error

s	p	MPSO	SPSO
1	10	1.73 (0.08)	1.77 (0.06)
1	200	2.18 (0.02)	2.88 (0.04)
5	10	3.52 (0.11)	4.28 (0.09)
5	200	3.93 (0.03)	4.36 (0.05)

For $s = 1$ and $p = 200$, the average offline error of MPSO was found to be 2.12(0.02), a figure which compares favorably with the best previous adaptive MPSO result without particle conversion and $n_{excess} = 4$ of 2.37 (0.03) [2]. This is also slightly better than the 10 (5 + 5) MPSO result of 2.26 (0.03) of the original version in [8], although it does not require us to specify the number of swarms. For SPSO, the offline error obtained is 2.88 (0.04), i.e., slightly worse. One possible explanation is that SPSO can not adjust the overall number of particles, and that the individual species are becoming too small to successfully track the moving peaks.

Both approaches suffer significantly as the shift severity is increased to $s = 5$, although it seems that MPSO is slightly better in this scenario.

6 Conclusions

This chapter has reviewed the application of particle swarms to dynamic optimization. The canonical PSO algorithm must be modified for good performance in dynamic environments. In particular the problems of outdated memory information and diversity loss must be addressed. Two promising PSO variants are the multi-swarm PSO (MPSO) and the speciation-based PSO (SPSO). They both maintain diversity by dividing the swarm into several subswarms. While MPSO starts with a single swarm and adds additional swarms as needed (all consisting of a predefined number of particles), SPSO has a fixed overall number of particles and dynamically distributes particles to swarms, usually starting with many swarms, slowly converging to the required number of swarms to cover all peaks. Also, they both use a mechanism to maintain diversity within a swarm. We have described MPSO and SPSO in detail and suggested new variations of both. For both approaches, we suggest

a reduction of the number of permanent quantum particles and a conversion of all neutral particles to quantum particles for one iteration only after a change has been detected. This approach seems to be more efficient as it speeds up convergence towards local peaks by maximizing its use of neutral particles. On the other hand, it still provides the diversity necessary to recapture a peak after it has moved. Also, we looked at the influence of the probability distribution for the quantum particles on performance.

As the empirical results showed, both variants outperform their previously published originals. Among the examined distributions, the uniform volume distribution outperformed the distributions with higher sampling probability around the species' best. On the other hand, the best possible radius is significantly less than the actual distance a peak shifts. The reason for the benefit of the uniform distribution is probably that the task for the quantum particles is exploration and tracking, rather than fine-tuning, which is better done by the neutral particles. The reason for the smaller radius is probably that as particles generated still possess velocities, placing them too close to the new peak might not be necessarily better, as the particles tend to 'overshoot' the optimum frequently.

Overall, the performance of MPSO and SPSO is comparable on slowly changing problems with fewer peaks, but MPSO seemed to be able to better cope with many peaks, while SPSO seems to be better when the changes are more severe.

Future work will aim at making both approaches more flexible, allowing MPSO to adapt the number of particles within a species, and SPSO to adapt the overall number of particles.

References

1. A. Engelbrecht. *Computational Intelligence*. John Wiley and sons, 2002.
2. T. Blackwell. Particle swarm optimization in dynamic environments. In: S. Yang et al., editors, *Evolutionary Computation in Dynamic and Uncertain Environments*. Springer, Berlin, Germany, 2007.
3. T. M. Blackwell. Particle swarms and population diversity I: Analysis. In: J. Branke, editor, *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pages 9–13, 2003.
4. T. M. Blackwell. Particle swarms and population diversity II: Experiments. In: J. Branke, editor, *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pages 14–18, 2003.
5. T. M. Blackwell and P. Bentley. Don't push me! Collision avoiding swarms. In: *Proc. of the 2002 Congress on Evolutionary Computation*, pages 1691–1696. IEEE Press, 2002.
6. T. M. Blackwell and P. J. Bentley. Dynamic search with charged swarms. In: W. B. Langdon et al., editor, *Proc. of the 2002 Genetic and Evolutionary Computation Conference*, pages 19–26. Morgan Kaufmann, 2002.
7. T. M. Blackwell and J. Branke. Multi-swarm optimization in dynamic environments. In: G. R. Raidl, editor, *Applications of Evolutionary Computing*, volume

3005 of *Lecture Notes in Computer Science*, pages 489–500. Springer, Berlin, Germany, 2004.
8. T. M. Blackwell and J. Branke. Multi-swarms, exclusion and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 10(4):459–472, 2006.
9. T. M. Blackwell. Swarms in dynamic environments. In: E. Cantu-Paz, editor, *Genetic and Evolutionary Computation Conference*, volume 2723 of *Lecture Notes in Computer Science*, pages 1–12. Springer, Berlin, Germany, 2003.
10. J. Branke. The Moving Peaks Benchmark Website. *http://www.aifb.uni-karlsruhe.de/jbr/movpeaks*.
11. J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer, 2001.
12. J. Branke, E. Salihoglu, and S. Uyar. Towards an analysis of dynamic environments. In: Beyer H.-G. et al., editor, *Proc. of the Genetic and Evolutionary Computation Conference, GECCO-2005*, pages 1433–1440. ACM Press, 2005.
13. J. Branke and H. Schmeck. Designing evolutionary algorithms for dynamic optimization problems. *Theory and Application of Evolutionary Computation: Recent Trends*, pages 239–262. Springer, Berlin, 2002. S. Tsutsui and A. Ghosh, editors.
14. A. Carlisle and G. Dozier. Adapting particle swarm optimisation to dynamic environments. In: *Int. Conference on Artificial Intelligence*, pages 429–434. CSREA Press, 2000.
15. M. Clerc. *Particle Swarm Optimization*. ISTE Publishing Company, 2006.
16. M. Clerc and J. Kennedy. The particle swarm: explosion, stability and convergence in a multi-dimensional space. *IEEE Transactions on Evolutionary Computation*, 6:158–73, 2000.
17. J. P. Coelho, P. B. De Moura Oliveira, and J. Boaventura Cunha. Non-linear concentration control system design using a new adaptive particle swarm optimiser. In: *Proc. of the 5th Portuguese Conference on Automatic Control*, pages 132–137, 2002.
18. R. Eberhart and Y. Shi. *Swarm Intelligence*. Morgan Kaufmann, 2001.
19. R. C. Eberhart and Y. Shi. Tracking and optimizing dynamic systems with particle swarms. In: *Proc. of 2001 Congress on Evolutionary Computation*, pages 94–100. IEEE Press, 2001.
20. X. Hu and R.C. Eberhart. Adaptive particle swarm optimisation: detection and response to dynamic systems. In: *Proc. of the 2002 Congress on Evolutionary Computation*, pages 1666–1670. IEEE Press, 2002.
21. S. Janson and M. Middendorf. A hierachical particle swarm optimizer for dynamc optimization problems. In: G. R. Raidl, editor, *Applications of Evolutionary Computing*, volume 3005 of *Lecture Notes in Computer Science*, pages 513–524. Springer, Berlin, Germany, 2004.
22. S. Janson and M. Middendorf. A hierarchical particle swarm optimizer for noisy and dynamic environments. *Genetic Programming and Evolvable Machines*, 7(4):329–354, 2006.
23. Y. Jin and J. Branke. Evolutionary optimization in uncertain environments – a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.
24. J. Kennedy. Bare bones particle swarm. In: *Proc. of the 2003 Conference on Evolutionary Computation*, pages 80–87. IEEE Press, 2003.
25. J. Kennedy and R.C. Eberhart. Particle swarm optimization. In: *International Conference on Neural Networks*, pages 1942–1948. IEEE Press, 1995.

26. J.-P. Li, M.E. Balazs, G.T. Parks, and P.J. Clarkson. A species conserving genetic algorithm for multimodal function optimization. *Evolutionary Computation*, 10(3):207–234, 2002.
27. X. Li. Adaptively choosing neighborhood bests in a particle swarm optimizer for multimodal function optimization. In: K. Deb et al., editor, *Proc. of the 6th Genetic and Evolutionary Copmutation Conference*, volume 3102 of *Lecture Notes in Computer Science*, pages 105–116. Springer, Berlin, Germany, 2004.
28. X. Li, J. Branke, and T. Blackwell. Particle swarm with speciation and adaptation in a dynamic environment. In: M. Keijzer et al., editor, *Proc. of the 8th Genetic and Evolutionary Computation Conference*, volume 1, pages 51–58. ACM Press, 2006.
29. G. Pan, Q. Dou, and X. Liu. Performance of two improved particle swarm optimization in dynamic optimization environments. In: *Proc. of the 6th International Conference on Intelligent Systems Design and Applications*, pages 1024–1028. IEEE Press, 2006.
30. D. Parrott and X. Li. A particle swarm model for tracking multiple peaks in a dynamic environment using speciation. In: *Proc. of the 2004 Congress on Evolutionary Computation*, pages 98–103, 2004.
31. D. Parrott and X. Li. Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Transactions on Evolutionary Computation*, 10(4):440 – 458, 2006.
32. K. E. Parsopoulos and M.N. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, pages 235–306, 2002.
33. A. Petrowski. A clearing procedure as a niching method for genetic algorithms. In: *Proc. of the 2003 Conference on Evolutionary Computation*, pages 798–803. IEEE Press, 2003.
34. C. Reynolds. Flocks, herds and schools: a distributed behavioral model. *Computer Graphics*, 21:25–34, 1987.
35. T. Richer and T. Blackwell. The Lévy particle swarm. In: *Proc. of the 2006 Congress on Evolutionary Computation*, pages 808– 815. IEEE Press, 2006.
36. J. Vesterstrom T. Krink and J. Riget. Particle swarm optimisation with spatial particle extension. In: *Proc. of the 2002 Congress on Evolutionary Computation*, pages 1474–1479. IEEE Press, 2002.
37. X. Li and K. H. Dam. Comparing particle swarms for tracking extrema in dynamic environments. In: *Proc. of the 2003 Congress on Evolutionary Computation*, pages 1772–1779. IEEE Press, 2003.
38. X. Zhang et al. Two-stage adaptive PMD compensation in a 10 Gbit/s optical communication system using particle swarm optimization algorithm. *Optics Communications*, 231:233–242, 2004.