Particle Swarm Optimization in Dynamic Environments

Tim Blackwell

Department of Computing, Goldsmiths College London SE14 6NW, UK
t.blackwell@gold.ac.uk

1 Introduction

Particle Swarm Optimization (PSO) is a versatile population-based optimization technique, in many respects similar to evolutionary algorithms (EAs). PSO has been shown to perform well for many static problems [30]. However, many real-world problems are dynamic in the sense that the global optimum location and value may change with time. The task for the optimization algorithm is to track this shifting optimum. It has been argued [14] that EAs are potentially well-suited to such tasks, and a review of EA variants tested in the dynamic problem is given in [13, 15]. It might be wondered, therefore, what promise PSO holds for dynamic problems.

Optimization with particle swarms has two major ingredients, the particle dynamics and the particle information network. The particle dynamics are derived from swarm simulations in computer graphics [21], and the information sharing component is inspired by social networks [32, 25]. These ingredients combine to make PSO a robust and efficient optimizer of real-valued objective functions (although PSO has also been successfully applied to combinatorial and discrete problems too). PSO is an accepted computational intelligence technique, sharing some qualities with Evolutionary Computation [1].

The application of PSO to dynamic problems has been explored by various authors [30, 23, 17, 9, 6, 24]. The overall consequence of this work is that PSO, just like EAs, must be modified for optimal results on dynamic environments typified by the moving peaks benchmark (MPB). (Moving peaks, arguably representative of real world problems, consist of a number of peaks of changing with and height and in lateral motion [12, 7].) The origin of the difficulty lies in the dual problems of *outdated memory* due to environment dynamism, and *diversity loss*, due to convergence.

Of these two problems, diversity loss is by far the more serious; it has been demonstrated that the time taken for a partially converged swarm to re-diversify, find the shifted peak, and then re-converge is quite deleterious to performance [3]. Clearly, either a re-diversification mechanism must be employed at (or before) function change, and/or a measure of diversity can be maintained throughout the run. There are four principle mechanisms for either re-diversification or diversity maintenance: randomization [23], repulsion [5], dynamic networks [24, 36] and multi-populations [29, 6].

Multi-swarms combine repulsion with multi-populations [6, 7]. Interestingly, the repulsion occurs between particles, and between swarms. The multi-population in this case is an interacting super-swarm of charged swarms . A charged swarm is inspired by models of the atom: a conventional PSO nucleus is surrounded by a cloud of 'charged' particles. The charged particles are responsible for maintaining the diversity of the swarm. Furthermore, and in analogy to the exclusion principle in atomic physics, each swarm is subject to an exclusion pressure that operates when the swarms collide. This prohibits two or more swarms from surrounding a single peak, thereby enabling swarms to watch secondary peaks in the eventuality that these peaks might become optimal. This strategy has proven to be very effective for MPB environments.

This chapter starts with a description of the canonical PSO algorithm and then, in Section 3, explains why dynamic environments pose particular problems for unmodified PSO. The MPB framework is also introduced in this section. The following section describes some PSO variants that have been proposed to deal with diversity loss. Section 5 outlines the multi-swarm approach and the subsequent section presents new results for a self-adapting multi-swarm, a multi-population with swarm birth and death.

2 Canonical PSO

In PSO, population members (particles) possess a memory of the best (with respect to an objective function) location that they have visited in the past, *pbest*, and of its fitness. In addition, particles have access to the best location of any other particle in their own network. These two locations (which will coincide for the best particle in any network) become attractors in the search space of the swarm. Each particle will be repeatedly drawn back to spatial neighborhoods close to these two attractors, which themselves will be updated if the global best and/or particle best is bettered at each particle update. Several network topologies have been tried, with the star or fully connected network remaining a popular choice for unimodal functions. In this network, 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.

Particles possess a velocity which influences position updates according to a simple discretization of particle motion

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

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

where $\mathbf{a}$, $\mathbf{v}$, $\mathbf{x}$ and t are acceleration, velocity, position and time (iteration counter) respectively. Eqs. 1, 2 are similar to particle dynamics in swarm simulations, but PSO particles do not follow a smooth trajectory, instead moving in jumps, in a motion known as a flight [28] (notice that the time increment dt is missing from these rules). The particles experience a linear or spring-like attraction, weighted by a random number, (particle mass is set to unity) towards each attractor. Convergence towards a good solution will not follow from these dynamics alone; the particle flight must progressively contract. This contraction is implemented by Clerc and Kennedy with a constriction factor χ, $\chi < 1$, [20]. For our purposes here, the Clerc-Kennedy PSO will be taken as the canonical swarm; χ replaces other energy draining factors extant in the literature such as a decreasing 'inertial weight' and velocity clamping. Moreover the constricted swarm is replete with a convergence proof, albeit about a static attractor (although there is some experimental and theoretical support for convergence in the fully interacting swarm where particles can move attractors [10]).

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

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

where $\boldsymbol{\epsilon}$ are vectors of random numbers drawn from the uniform distribution $U[0,1]$, $c > 2$ is the spring constant and $\mathbf{p}_i$, $\mathbf{p}_g$ are particle and global attractors. This formulation of the particle dynamics has been chosen to demonstrate explicitly 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 1.

3 PSO problems with moving peaks

As has been mentioned in Sect 1, PSO must be modified for optimal results on dynamic environments typified by the moving peaks benchmark (MPB). These modifications must solve the problems of outdated memory, and of lost diversity. This explains the origins of these problems in the context of MPB, and shows how memory loss is easily addressed. The following section then considers the second, more severe, problem.

3.1 Moving Peaks

The dynamic objective function of MPB, $f(\mathbf{x}, t)$, is optimized at 'peak' locations $\mathbf{x}^*$ and has a global optimum at $\mathbf{x}^{**} = \arg\max\{f(\mathbf{x}^*)\}$ (once more, assuming optimization means maximizing). Dynamism entails a small movement of magnitude s, and in a random direction, of each $\mathbf{x}^*$. This happens

Algorithm 1 Canonical PSO

```
FOR EACH particle $i$
    Randomly initialize $\mathbf{v}_i, \mathbf{x}_i = \mathbf{p}_i$
    Evaluate $f(\mathbf{p}_i)$
    $g = \arg\max f(\mathbf{p}_i)$
REPEAT
    FOR EACH particle $i$
        Update particle position $\mathbf{x}_i$ according to eqs.. 1, 2 and 3
        Evaluate $f(\mathbf{x}_i)$
        //Update personal best
        IF $f(\mathbf{x}_i) > f(\mathbf{p}_i)$ THEN
            $\mathbf{p}_i = \mathbf{x}_i$
        //Update global best
        IF $f(\mathbf{x}_i) > f(\mathbf{p}_g)$ THEN
            $\mathbf{p}_g = \arg\max f(\mathbf{p}_i)$
UNTIL termination criterion reached
```

every K evaluations and is accompanied by small changes of peak height and width. There are p peaks in total, although some peaks may become obscured. The peaks are constrained to move in a search space of extent X in each of the d dimensions, $[0, X]^d$.

This scenario, which is not the most general, nevertheless has been put forward as representative of real world dynamic problems [12] and a benchmark function is publicly available for download from [11]. Note that small changes in $f(\mathbf{x}^*)$ can still invoke large changes in $\mathbf{x}^{**}$ due to peak promotion, so the many peaks model is far from trivial.

3.2 The problem of outdated memory

Outdated memory happens at environment change when the optima may shift in location and/or value. Particle memory (namely the best location visited in the past, and its corresponding fitness) may no longer be true at change, with potentially disastrous effects on the search.

The problem of outdated memory is typically solved by either assuming that the algorithm knows just when the environment change occurs, or that it can detect change. In either case, the algorithm must invoke an appropriate response. One method of detecting change is a re-evaluation of f at one or more of the personal bests $\mathbf{p}_i$ [17, 23]. A simple and effective response is to re-set all particle memories to the current particle position and f value at this position, and ensuring that $\mathbf{p}_g = \arg\max f(\mathbf{p}_i)$. One possible drawback is that the function has not changed at the chosen $\mathbf{p}_i$, but *has* changed elsewhere. This can be remedied by re-evaluating f at all personal bests, at the expense of doubling the total number of function evaluations per iteration.

3.3 The problem of lost diversity

Equally troubling as outdated memory is insufficient diversity at change. The population takes time to re-diversify and re-converge, effectively unable to track a moving optimum.

It is helpful at this stage to introduce the swarm diameter $|S|$, defined as the largest distance, along any axis, between any two particles [2], as a measure of swarm diversity (Fig. 1). Loss of diversity arises when a swarm is converging on a peak. There are two possibilities: when change occurs, the new optimum location may either be within or outside the collapsing swarm. In the former case, 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 optimum shift is significantly far from the swarm, 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 [5]. This effect is illustrated in Fig. 2.

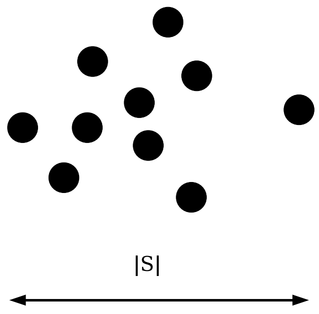


Fig. 1. The swarm diameter

These considerations can be quantified with the help of a prediction for the rate of diversity loss [2, 3, 10]. In general, the swarm shrinks at a rate determined by the constriction factor and by the local environment at the optimum. For static functions with spherical symmetric basins of attraction, the theoretical and empirical analysis of the above references suggest that the rate of shrinkage (and hence diversity loss) is scale invariant and is given by a scaling law

$$|S(t)| = C\alpha^t \tag{4}$$

for constants C and $\alpha < 1$, where $\alpha \approx 0.92$ and C is the swarm diameter at iteration $t = 0$. The number of function evaluations between change, K, can be converted into a period measured in iterations, L by considering the total number of function evaluations per iteration including, where necessary, the extra test-for-change evaluations. We might expect that, for peak shift distance s and box size $X = |S(0)|$, if $s >> S_L = X\alpha^L$, tracking will be very hard since the swarm has already converged to a very small ball at the first change. The experiments reported in [3] show that canonical PSO fails to track a single peaked dynamic environment defined by $s = 8.7, L = 100, X = 10$. Since S_L computes to 0.0024, this is hardly surprising.

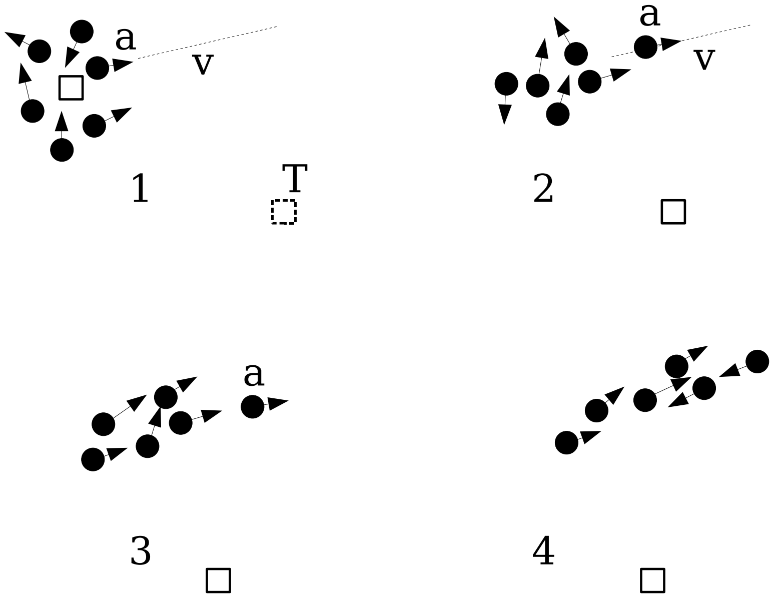


Fig. 2. Sequence of frames showing possible behavior when optimum shift is greater that swarm diversity. When the attractor T (square box, frame 1) shifts, particle a is at the global best, p_g. a continues along trajectory v since it is not accelerated in this update (frame 2). Particle a continues to move along v, repositioning p_g at each update, becoming in effect the swarm leader (frame 3). After a while, the swarm oscillates along v, about a point perpendicular to T (frame 4). Eventually random fluctuations will cause another particle to deviate from v and move closer towards the attractor. The swarm soon follows and converges on T.

4 Diversity lost and diversity regained

There are two solutions in the literature to the problem of insufficient diversity. Either a diversity increasing mechanism can be invoked at change (or at predetermined intervals), or some permanent means can be put in place to ensure there is sufficient diversity at all times [7]. These modifications are the subject of this section.

4.1 Re-diversification

Hu and Eberhart [23] study a number of re-diversification mechanisms. These all involve randomization of the entire, or part of, the swarm. This happens when re-evaluation of the objective function at one or several of the attractors detects change, or at a pre-set interval. Clearly the problem with this approach is the arbitrariness of the extra parameters. 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. And, of course, if tests for change happen at pre-determined intervals, there is a danger of missing a shift. The arbitrariness of the extra parameters can only be solved if much prior knowledge about f's dynamism is available, or some other higher-level modification mechanism scheme is implemented. Such a scheme could infer details about f's dynamism during the run, making appropriate adjustments to the re-diversification parameters. So far, though, higher level modifications such as these have not been studied.

4.2 Maintaining diversity by repulsion

A constant, and hopefully good enough, degree of swarm diversity can be maintained at all times either through some type of repulsive mechanism, or by adjustments to the information sharing neighborhood. Repulsion can either be between particles, or from an already detected optimum. For example, Krink et al [34] study finite-size particles as a means of preventing premature convergence. The hard sphere collisions produce a constant diversification pressure. Alternatively, Parsopoulos and Vrahatis [30] place a repeller at an already detected optima, 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 [5, 4, 9, 6]. In this model, a swarm is comprised of a 'charged' and a 'neutral' sub-swarm. The model can be depicted as a cloud of charged particles orbiting a contracting, neutral, PSO nucleus, Fig. 3. The charged particles can be either classical or quantum particles; either type are discussed in some depth in references [6, 7] and in the following section. Charge enhances diversity in the vicinity of the converging PSO sub-swarm, so that optimum shifts within this cloud should be trackable. Good tracking (outperforming canonical PSO) has been demonstrated for unimodal dynamic environments of varying severities [3].

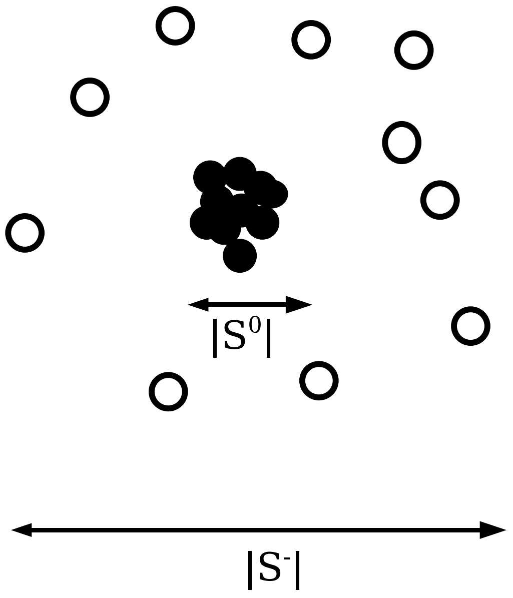


Fig. 3. The Atom Analogy. The situation depicted here shows a PSO sub-swarm of neutral particles (filled circles), converging at an optimum. The neutral swarm diameter, $|S^0|$, is shrinking by a factor of 0.92 at each iteration. This sub-swarm is surrounded by a number of charged particles with constant diversity $|S^-|$. Both sub-swarms share the same global attractor p_g. Optimum moves to locations within the charged sub-swarm will be rapidly re-optimized by the swarm as a whole.

4.3 Maintaining diversity with 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 use a grid-like neighborhood structure, and Jansen and Middendorf test a hierarchical structure, reporting improvements over unmodified PSO for unimodal dynamic environments [24, 36].

4.4 Maintaining diversity with multi-populations

A number of research groups have considered multi-populations as a means of enhancing diversity. The multi-population idea is particularly helpful in multi-modal environments such as many peaks. The aim here is to allow each population to converge on a promising peak. Then, if any secondary peak becomes the global optimum as a result of change, a population is close at hand. Multi-population techniques include niching, speciation and multi-swarms.

In the static context, the niching PSO of Brits et al [16] can successfully optimize some static benchmark problems. In *nichePSO*, if a particle's fitness changes very little (the variance in fitness is less than a threshold) over a small number of iterations, a two particle sub-swarm is created from this particle and its nearest spatial neighbor. This technique, as the authors point out, would fail in a dynamic environment because niching depends on a homogeneous distribution of particles in the search space, and on a training phase.

A speciation PSO variation, known as clearing [31] has been adopted by Li in the static context and generalized by Parrot and Li to dynamic functions [27, 29]. Under clearing, the number and size of swarms is adjusted dynamically by constructed an ordered list of particles, ranked according to their fitness, with spatially close particles joining a particular species. This method relies on a speciation radius and has no further diversity mechanism. Other related work includes using different swarms in cooperation to optimize different parts of a solution [35], a two swarm min-max optimization algorithm [33] and iteration by iteration clustering of particles into sub-swarms [26]. Apart form Parrot and Li's speciation, none of these multi-population techniques have been generalized as dynamic optimizers. The multi-swarm approach of Blackwell and Branke is described in detail in the next section.

5 Multi-swarms

A combined approach might be to incorporate the virtues of the multi-population approach and of swarm diversity enhancing mechanisms such as repulsion. Such an optimizer would be well suited to the many peaks environment. Multi-swarms, first proposed in a non-optimization context [8] would seem to do just this. The extension of multi-swarms to a dynamic optimizer was made by Blackwell and Branke [6], and is inspired by Branke's own self-organizing scouts (SOS) [13]. The scouts have been shown to give excellent results on the many peaks benchmark.

A multi-swarm is a colony of charged swarms interacting locally via *exclusion* and globally by *anti-convergence*. The motivation for these operators is that a mechanism must be found to prevent two or more swarms from trying to optimize the same peak (exclusion) and also to maintain multi-swarm diversity, that is to say the diversity amongst the population of swarms as a whole (anti-convergence). Multi-swarms have been compared very favorably to both hierarchical swarms and to self-organizing scouts.

We consider below the main ingredients of the multi-swarm algorithm in more depth. In particular we will assess the values of parameters with relation to the many peaks benchmark. A complete discussion of parameter choices is given in [7]. The multi-swarm algorithm is presented in Algorithm 2.

5.1 Atom Analogy

In the atom analogy, each swarm is pictured as an atom with a contracting nucleus of neutral PSO particles, and an enveloping cloud of charged particles. All particles are in fact members of the same information network

Algorithm 2 Multi-Swarm

```
//Initialization
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
  // Anti-Convergence
  IF all swarms have converged THEN
    Re-initialize worst swarm.
  FOR EACH swarm $n$
    // Test for Change
    Evaluate $f(\mathbf{p}_{ng})$.
    IF new value is different from last iteration THEN
      Re-evaluate each particle attractor.
      Update swarm attractor.
    FOR EACH particle $i$ of swarm $n$
      // Update Particle
      Apply equations (3) - (9) 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 > $max$
```

so that they all (in the star topology) have access to $\mathbf{p}_g$. The mutual repulsions between the charged particles may follow a deterministic, classical, rule (Coulomb repulsion, parameterized by particle charge Q). Alternatively, in a quantum atom, the particles are positioned within a hypersphere of radius r_{cloud} centered on $\mathbf{p}_g$ according to a probability distribution. So far, two uniform distributions have been tested. References [6, 7] consider a uniform shell distribution, $p(r, dr) = \rho(r)dr = const$, where ρ is a probability density, r is a shell radius, dv is a volume element and p is a probability. Recent work has investigated a uniform volume distribution, $p(\mathbf{x}, dv) = \rho(\mathbf{x})dv = const$. In both cases, the distributions are normalized so that $p(r > r_{cloud}) = 0$. Other, non-uniform and possibly dynamic distributions might be favorable for the quantum swarm in some cases, but such distributions remain unexplored.

The quantum atom has the advantages of lower complexity and an easily controllable distribution: the Coulomb repulsion has quadratic complexity and highly fluctuating electron orbits [2, 3]. An order of magnitude estimation for the parameters r_{cloud} (for quantum swarms), or Q, for classically charged clouds, can be made by supposing that good tracking will occur if the mean charged particle separation $< |\mathbf{x}^- - \mathbf{p}_g| >$ is comparable to s. This separation is easy to compute for the quantum swarm: only empirical data is available for classical charged particles.

5.2 Exclusion

In order to demonstrate the necessity for exclusion, first consider an assembly of non-interacting swarms, also known as a many-swarm. A many-swarm has M swarms, and each swarm, for symmetrical configurations has N^0 neutral and N^- charged particles. Such a many-swarm is written $M * (N^0 + N^-)$. Since the swarms do not interact - either dynamically through the particle velocity and positions updates, or by sharing information - the M swarms are completely independent, and any number of them may try to optimize the same peak. This is undesirable because it is clearly inefficient to have two or more swarms on the same peak, and in any case, we wish to distribute the swarms throughout the search space for peak watching. Hence the swarms must interact in some way. One possibility is to allow the swarms to interact topologically and share a single information network. The multi-swarm approach is to seek a *spatial* interaction between swarms. Such an interaction might repel entire swarms from already occupied peaks. However, Coulomb repulsion, or some such similar physics-inspired repulsion would not be satisfactory, because the attractive pull towards the peak might be balanced by the repulsive force away from other nearby swarms. In such an equilibrium, no swarm would be able to optimize the peak.

Exclusion is inspired by the exclusion principle in atomic and molecular physics. This principle states that no two electrons may occupy the same state. The exclusion principle provides an effective repulsive force between two gas molecules with overlapping electron clouds [22]. However the effective force does not arise from any deterministic equation that governs the electron motion, but is a rule imposed on the probability distributions of the electron positions. A version of this principle for interacting swarms is a rule that 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. The exclusion operator simple randomizes, in the entire search space, the worse swarm in any collision, as judged by the current best value determined by the swarm, $f(\mathbf{p}_g)$. The configuration of the interacting multi-swarm is written $M(N^0 + N^-)$.

An order of magnitude estimation for r_{excl} can be made by assuming that all p peaks are evenly distributed in X^d. The linear diameter of the basin of attraction of a peak is then, on average, $d_{boa} = X/p^{1/d}$. It is reasonable to assume that swarms that are closer than this distance should experience exclusion, since the overall strategy is to place one swarm on each peak.

5.3 Anti-Convergence

Anti-convergence is a simple operator that is designed to ensure there is at least one free swarm in the multi-swarm at all times. A free swarm is one that is patrolling the search space rather than converging on a peak. A swarm is assumed to be converging when the neutral swarm diameter is less than a convergence diameter, $2r_{conv}$. The idea is that if the number of swarms is less than the number of peaks, all swarms may converge, leaving some peaks unwatched. One of these unwatched peaks may later become optimal. The presence of free swarms maintains multi-swarm diversity and encourages response to peak promotion.

Estimations of r_{conv} are difficult, but some progress can be made by considering the rate of convergence of the neutral swarm, as given by Equation 4. A lower bound on r_{conv} can be estimated from the ideal case that a swarm immediately tracks a shifted peak. This means that the swarm size at the shift is about s and the swarm has K function evaluations worth of time to contract around the peak. On the other hand, r_{conv} should certainly be less than r_{excl} because exclusion occurs before convergence.

Note that there are two levels of diversity. Diversity at the swarm level , as enforced by exclusion, enables a single swarm to track a single moving peak and diversity at the multi-swarm level enables the multi-swarm as a whole to find new peaks.

5.4 Multi-swarm cardinality

The multi-swarm cardinality M can be estimated from 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. Anti-convergence is expected to be beneficial for $M < p$. Optimally, we suppose that $M = p$, and in this case anti-convergence can be switched off, since the multi-swarm has just the right number of swarms.

5.5 Results

A exhaustive series of multi-swarm experiments has been conducted for the many peaks benchmark. The standard settings for MPB are number of peaks, $p = 10$, change period in function evaluations, $K = 5000$, peak shift severity, $s = 1.0$, dimensionality, $d = 5$ and search space range $X = 100$. The peak heights and widths vary randomly but are constrained to $[30, 70]$ and $[1, 12]$ respectively. Each experiment actually consists of 50 runs for a given set of multi-swarm parameters. Each run uses a different random number generator seed for initialization of the swarms, the swarm update algorithm and the MPB generator. Other non-standard MPB's were also tested for comparisons. Multi-swarm performance is quantified by the offline error which is the average, at any point in time, of the error of the best solution found since the last environment change. This measure is commonly used for scenarios such as the MPB and is zero for perfect tracking.

Various values of multi-swarm cardinality M were tested for fixed total number of particles as given by the expression $N_{total} = M(N^0 + N^-)$. As expected, $M = p$ was found to be optimal. The multi-swarm coped well with shift severities between 1.0 and 6.0. The multi-swarm offline errors for MPB's with different numbers of peaks were lass than 3.0 for $5 \leq p \leq 200$. Anti-convergence was found to bring a significant improvement when $M < p$. The robustness of the algorithm, and its generalizability into higher dimensions, was also tested by taking $d = 10$ and varying the predicted optimal parameter settings (i.e. r_{excl}, r_{conv}, r_{cloud} and Q) by 20

In all cases, the multi-swarms with charge outperform many-swarms, PSO and multi-swarms without charge. Furthermore, quantum swarms perform better than classical charged swarms. The offline error, as compared to hierarchical swarms and self-organizing scouts, is cut by a half, and the improvement over a randomization scheme is about an order of magnitude. It seems, therefore, that the multi-swarm is a very promising approach for problems of the MPB class.

6 Self-adapting multi-swarms

The multi-swarm model of the previous section introduced a number of new parameters. Although recommendations can be made for these settings, this analysis depends on prior knowledge of the environment and on many test runs. A laudable goal for any optimization technique is the reduction of hand-tunable parameters. Although parameter adjustments might improve performance for any particular problem, a general purpose method that might perform reasonably well across a spectrum of problems is certainly attractive. We therefore wonder to what extent PSO and PSO-variants can find their own best parameter settings during a single run. Such self-adapting algorithms might not return the best performance on any particular problem instance. However to make fair comparisons, the total number of function evaluations (or iterations) involved in all the trials of the hand-tuned method must be taken into account. This point is emphasized by Clerc in his explorations of 'Tribes', a self-adapting PSO [18, 19].

Here we will describe self-adaptations at the level of the multi-swarm. Future work will seek to incorporate adaptations of individual swarms into this scheme. The following will assume $(5^0 + 5^-)$ swarms with canonical PSO neutral particles and quantum charged particles and with the swarm diversity parameter, r_{cloud}, determined by the peak shift severity. This recipe gave the best results for the environments studied in the previous section. The parameters at the multi-swarm level are the number of swarms M and the exclusion and convergence radii r_{excl} and r_{conv}. It is assumed in the following that the multi-swarm has access to the dimensionality d and the extent of the search space X, but not the MPB parameters p or K. (Previously, r_{excl}, r_{conv} and M were determined with knowledge of p and K.)

6.1 Swarm birth and death

The basic idea is to allow the multi-swarm to regulate its size by bringing new swarms into existence, or by removing redundant swarms. The aim, as before, is to place a swarm on each peak, and to maintain multi-swarm diversity with (at least one) patrolling swarm. The multi-swarm therefore needs a new swarm if all swarms are currently converging. Alternatively, if there are too many free swarms (i.e. those that fail the convergence criterion), a free swarm should be removed. If there is more than one free swarm, the choice for removal is arbitrary and a simple rule might be to remove the worst of the free swarms, as judged by $f(\mathbf{p}_g)$.

This simple birth/death mechanism removes the need for the anti-convergence operator, and for specifying the multi-swarm cardinality. The self-adapting version of Algorithm 2 is given below in Algorithm 3, where M_{free} is the number of free swarms at iteration t, and identical steps in Algorithm 2 have been abbreviated.

Algorithm 3 Self-adapting multi-swarm

```
Begin with a single free swarm, randomized in $X^d$
At each iteration $t$:
  IF $M_{free} = 0$, generate a new free swarm
  ELSE If $M_{free} > n_{excess}$, remove worst free swarm
  FOR EACH swarm $n$
    test for change
    IF swarm $n$ is excluded, randomize
    ELSE update particle velocities and positions
    update attractors
    test for convergence
  apply exclusion
REPEAT
```

For generality, Algorithm 3 also specifies a redundancy parameter n_{excess} which is set to the desired number of free swarms. A simple choice is to suppose that $n_{excess} = 1$, but this may not give sufficient diversity if there are many peaks. Alternatively, $n_{excess} = \infty$ means that no swarm can ever be removed. Intermediate values control the amount of multi-swarm diversity. Part of the purpose of the experiments reported below is to assess the algorithm for robustness in n_{excess}.

The multi-swarm size $M(t)$ is dynamic and at any iteration t is given by

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

Swarm convergence and exclusion are now determined by a dynamic convergence radius $r(t)$ defined by

$$r(t) = \frac{X}{2M^{1/d}} \tag{6}$$

which has been chosen to ensure a mean volume per swarm of $(2r)^d = \frac{X^d}{M}$, a condition which might be expected to be if the peaks are, on average, uniformly distributed in X^d. The number of free swarms at any time is the difference between the multi-swarm size and the number of converging swarms. A swarm is defined as ‘converging’ if its diameter is less than $2r$. The exclusion radius is replaced by $r(t)$. Hence two parameters, M and r_{excl} and one operator, anti-convergence, have been removed from the multi-swarm algorithm, at the expense of introducing a new parameter, n_{excess}.

6.2 Results

A number of experiments using the MPB of Section 5.5 with 10 and 200 peaks were conducted to test the efficacy of the self-adapting multi-swarm for various values of n_{excess}. The uniform volume distribution described in Section 5.1 was used. An empirical investigation revealed that $r_{cloud} = 0.5s$ for shift length s yields optimum tracking for these MPB’s, and this was the value used here.

Table 6.2 shows the raw and rounded offline errors for $1 \leq n_{excess} \leq 7$ and for $n_{excess} = \infty$. Only the rounded errors are significant, but the pre-rounded values have been reported in order to examine algorithm functionality. For comparison, the best performance of an unadapted $10(10^0+10^-)$ multi-swarm for the $p = 10$ and $p = 200$ environments is, respectively, 1.75(0.06) and 2.26(0.03) [7]. The best self-adapting multi-swarm errors are 1.77(0.05) and 2.37(0.03), only slightly higher than the hand-tuned values. The constancy of the raw offline error for $n_{excess} \geq 5$ shows that the algorithm never tries to generate 6 or more swarms: $n_{excess} = 5$ is equivalent to setting n_{excess} to ∞.

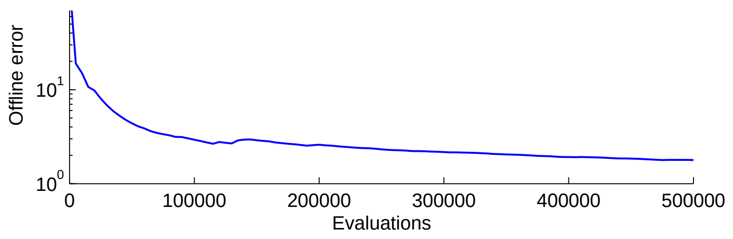
Table 1. Variation of offline error with n_{excess} for 10 and 200 dynamic peaks. The raw data demonstrates identical algorithm behavior for $n_{excess} \geq 5$

	Raw		Rounded (standard error)	
n_{excess}	$p = 10$	$p = 200$	$p = 10$	$p = 200$
1	1.9729028458116666	2.5383187958098343	1.97(0.09)	2.54(0.04)
2	1.879641879674056	2.398361911062971	1.88(0.07)	2.40(0.02)
3	1.7699076299648027	2.386396596554201	1.77(0.05)	2.39(0.03)
4	1.8033988974332964	2.372590853208213	1.80(0.06)	2.37(0.03)
5	1.8013758537004643	2.365026825401844	1.80(0.06)	2.37(0.03)
6	1.8010120393728533	2.3651325663361167	1.80(0.06)	2.37(0.03)
7	1.8010120393728533	2.3651325663361167	1.80(0.06)	2.37(0.03)
infinity	1.8010120393728533	2.3651325663361167	1.80(0.06)	2.37(0.03)

6.3 Discussion

Theoretically, $n_{excess} = 1$ would appear to be an ideal setting, allowing the multi-swarm to adapt to the number of peaks, whilst always maintaining a single free swarm. However, inspection of the numbers of free and converged swarms for a single function instance revealed that the self-adaptation mechanism at $n_{excess} = 1$ frequently adds a swarm, only to remove one at the subsequent iteration. The explanation is believed to be the following: suppose a swarm (swarm A) has started to converging on a peak. A new free swarm, swarm B, will be created. Since A is just at the edge of convergence, fluctuations in the swarm size may cause the swarm to appear to be free at the next iteration. Hence there will be two free swarms, and one must be removed - almost certainly swarm B, since this has had little chance to improve its $\mathbf{p}_g$. Swarm A will again start to converge (according to the criterion), causing the creation of a new free swarm at the next iteration. Such repeated creations and annihilations of the free swarm waste valuable function evaluations.

Another possible setting is $n_{excess} = \infty$. This effectively turns swarm removal off. Although there is no check to the number of swarms, it is not unreasonable to suppose that given enough time, the multi-swarm would stabilize at $M_{conv} = p$ and $M_{free} = 1$. The convergence criterion is rather naive and may mark a free swarm as converging even though it is not associated with a peak. This will cause the generation of another free swarm, with no means of removal. This will only be a problem when $M_{conv} > p$, a situation that might not even happen within the time-scale of the run. For example, Figures 4 and 5 show convergence data for a single run at $p = 10$ and $p = 200$. For $p = 10$, the eleventh swarm is generated at function evaluation, $n_{eval}, =$ 254054. There are 500000 evaluations in a run, and in the remaining evaluations the multi-swarm size is, at most 13, and remains fairly steady after the 400000th evaluation. The $p = 200$ trial shows a steadily growing multi-swarm, which never attains complete coverage of all peaks, ending with 47 converging swarms and 1 free swarm.



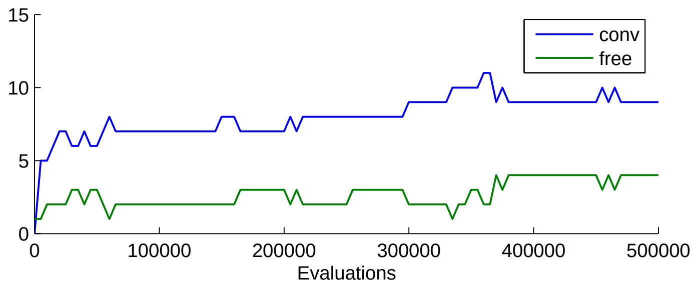


Fig. 4. Convergence of the self-adapting $n_{excess} = \infty$ multi-swarm for a single instance of the 10 peak MPB environment. Upper plot shows offline error, lower plot shows number of converged and free swarms

The flexibility of a swarm removal mechanism is desirable for a number of reasons. For example, two peaks might move within r_{excl} of each other, causing a previously converged swarm to vaporize through exclusion (the better swarm remains). Or maybe a peak i becomes invisible if its height is smaller than other peak heights at its optimizer i.e. if $f(\mathbf{x_i^*}) < f(\mathbf{x_j})$ for some $j \neq i$. In either case, superfluous free swarms will consume function evaluations.

The redundancy n_{excess} can be set at any value between the two extremes of $n_{excess} = 1$ and $n_{excess} = \infty$. ($n_{excess} = 0$ gives very bad performance, no swarms at all may be added and the single swarm converges, and remains on, the first peak it finds.) The results for $p = 10$ and $p = 200$ indicate that tuning of n_{excess} can improve performance for runs where the multi-swarm has found all the peaks. Tuning can prevent the multi-swarm from generating too many free swarms - for example 3 free swarms are optimal for $p = 10$. However, setting n_{excess} at either extreme still produces good performance, and better than the comparison algorithms cited in Section 5.5. Perhaps the multi-swarm itself could tune n_{excess} during a run. A more sophisticated convergence criterion would also have to be devised. For example, the convergence criterion could take into account both the swarm diameter and the rate of improvement of $f(\mathbf{p}_g)$.

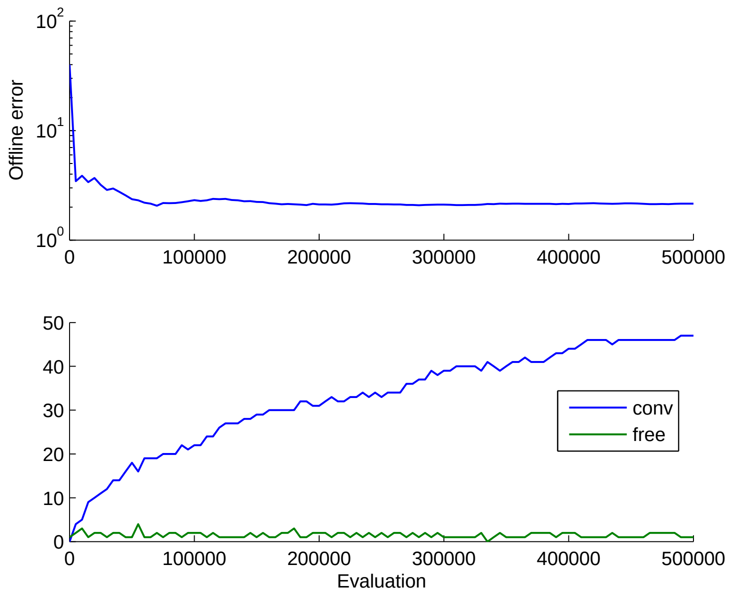


Fig. 5. Convergence of the self-adapting $M_{excess} = \infty$ multi-swarm for a single instance of the 200 peak MPB environment. Upper plot shows offline error, lower plot shows number of converged and free swarms

7 Summary

This chapter has reviewed the application of particle swarms to dynamic optimization. The canonical PSO algorithm must be modified for good performance in environments such as many peaks. In particular the problem of diversity loss must be addressed. The most promising PSO variant to date is the multi-swarm; a multi-swarm is a colony of swarms, where each swarm, drawing from an atomic analogy, consists of a canonical PSO surrounded by a cloud of charged particles. The underlying philosophy behind multi-swarms is to place a separate PSO on the best peaks, and to maintain a population of patrolling particles for the purposes of identifying new peaks. Movement of any peak that is being watched by a swarm is tracked by the charged particles. An exclusion operator ensures that only one swarm can watch any one peak, and anti-convergence seeks to maintain a free, patrolling swarm.

New work on self-adaptation has also been presented here. Self-adaptation aims at reducing the number of tunable parameters and operators. Some progress has been made at the multi-swarm level, where a mechanism for swarm birth and death has been suggested; this scheme eliminates one operator and allows the number of swarms and an exclusion parameter to adjust dynamically. One free parameter, the number of patrolling swarms, still exists, but results suggest that the algorithm is not overly sensitive to this number.

Self-adaptations at the level of each swarm, in particular allowing particles to be born and to die, and self-regulation of the charged cloud radius remain unexplored.

References

1. A.Engelbrecht. *Computational Intelligence.* John Wiley and sons, 2002.
2. 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. http://www.ubka.uni-karlsruhe.de/cgi-bin/psview?document=2003%2Fwiwi%2F1.
3. 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. http://www.ubka.uni-karlsruhe.de/cgi-bin/psview?document=2003%2Fwiwi%2F1.
4. T. M. Blackwell and P. Bentley. Don't push me! collision avoiding swarms. In *Congress on Evolutionary Computation*, pages 1691–1696, 2002.
5. T. M. Blackwell and P. J. Bentley. Dynamic search with charged swarms. In W. B. Langdon et al., editor, *Genetic and Evolutionary Computation Conference*, pages 19–26. Morgan Kaufmann, 2002.
6. T. M. Blackwell and J. Branke. Multi-swarm optimization in dynamic environments. In G. R. Raidl, editor, *Applications of Evolutionary Computing*, volume 3005 of *LNCS*, pages 489–500. Springer, 2004.
7. T. M. Blackwell and J. Branke. Multi-swarms, exclusion and anti-convergence in dynamic environments. *IEEE transactions on Evolutionary Computation*, to appear.
8. T.M. Blackwell. Swarm music: Improvised music with multi-swarms. In *Proc AISB '03 Symposium on artificial intelligence and creativity in arts and science*, pages 41–49, 2003.
9. T.M. Blackwell. Swarms in dynamic environments. In E. Cantu-Paz, editor, *Genetic and Evolutionary Computation Conference*, volume 2723 of *LNCS*, pages 1–12. Springer, 2003.
10. T.M. Blackwell. Particle swarms and population diversity. *Soft Computing*, 9:793–802, 2005.
11. J. Branke. The moving peaks benchmark website. *http://www.aifb.uni-karlsruhe.de/jbr/movpeaks.*
12. J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. In *Congress on Evolutionary Computation CEC99*, volume 3, pages 1875–1882. IEEE, 199. ftp://ftp.aifb.uni-karlsruhe.de/pub/jbr/branke_cec1999.ps.gz.
13. J. Branke. Evolutionary approaches to dynamic environments - updated survey. In *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pages 27–30, 2001. http://www.aifb.uni-karlsruhe.de/ jbr/EvoDOP/Papers/gecco-dyn2001.pdf.
14. J. Branke. *Evolutionary Optimization in Dynamic Environments.* Kluwer, 2001. http://www.aifb.uni-karlsruhe.de/ jbr/book.html.

15. J. Branke and H. Schmeck. Designing evolutionary algorithms for dynamic optimization problems. *Theory and application of evolutionary computation: recent trends*, pages 239–262, 2002. S. Tsutsui and A. Ghosh, editors.
16. R. Brits, A.P. Engelbrecht, and F. van den Bergh. A niching particle swarm optimizer. In *Fourth Asia-Pacific conference on simulated evolution and learning*, pages 692–696, 2002.
17. A. Carlisle and G. Dozier. Adapting particle swarm optimisationto dynamic environments. In *Proc of int conference on artificial intelligence*, pages 429–434, 2000.
18. M. Clerc. Think locally act locally - a framework for adaptive particle swarm optimizers. Technical report, 2002. http://clerc.maurice.free.fr/pso/ (accessed June 29, 2006).
19. M. Clerc. *Particle Swarm Optimization.* ISTE publishing company, 2006.
20. 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.
21. C.Reynolds. Flocks, herds and schools: a distributed behavioral model. *Computer Graphics*, 21:25–34, 1987.
22. A.P. French and E.F. Taylor. *An introduction to quantum physics.* W.W. Norton and Company, 1978.
23. X. Hu and R.C. Eberhart. Adaptive particle swarm optimisation: detection and response to dynamic systems. In *Proc Congress on Evolutionary Computation*, pages 1666–1670, 2002.
24. 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 *LNCS*, pages 513–524. Springer, 2004.
25. J. Kennedy and R.C. Eberhart. Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on neural networks*, pages 1942–1948, 1995.
26. J. Kennnedy. Stereotyping: improving particle swarm performance with cluster analysis. In *Congress on Evolutionary Computation*, pages 1507–12, 2000.
27. X. Li. Adaptively choosing neighborhood bests in a particle swarm optimizer for multimodal function optimization. In K.Deb et al, editor, *Proceedings of the Genetic and Evolutionary Copmutation Conference, GECCO-2004*, volume 3102 of *LNCS*, pages 105–116. Springer, 2004.
28. B. Mandelbrot. *The Fractal Geometry of Nature.* W. H. Freeman and Company, 1983.
29. D. Parrott and X. Li. A particle swarm model for tracking multiple peaks in a dynamic environment using speciation. In *Congress on Evolutionary Computation*, pages 98–103, 2004.
30. K.E. Parsopoulos and M.N. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, pages 235–306, 2002.
31. A. Petrowski. A clearing procedure as a niching method for genetic algorithms. In J. Grefenstette, editor, *Int'l Conference on Evolutionary Computation*, pages 798–803. IEEE, 2003.
32. R.Eberhart and Y.Shi. *Swarm intelligence.* Morgan Kaufmann, 2001.
33. Y. Shi and A. Khrohling. Co-evolutionary particle swarm optimization to solve min-max problems. In *Congress on Evolutionary Computation*, pages 1682–1687, 2002.

34. J. Vesterstrom T. Krink and J. Riget. Particle swarm optimisation with spatial particle extension. In *Congress on Evolutionary Computation*, page 14741479, 2002.
35. F. van den bergh and A. P. Englebrecht. A cooperative approach to particle swarm optimization. *IEEE transactions on Evolutionary Computation*, pages 225–239, 2004.
36. X.Li and K. H. Dam. Comparing particle swarms for tracking extrema in dynamic environments. In *Congress on Evolutionary Computation*, pages 1772–1779, 2003.

Index