Swarm Intell (2007) 1: 33–57
DOI 10.1007/s11721-007-0002-0

Particle swarm optimization

An overview

Riccardo Poli · James Kennedy · Tim Blackwell

Received: 19 December 2006 / Accepted: 10 May 2007 / Published online: 1 August 2007

Abstract Particle swarm optimization (PSO) has undergone many changes since its introduction in 1995. As researchers have learned about the technique, they have derived new versions, developed new applications, and published theoretical studies of the effects of the various parameters and aspects of the algorithm. This paper comprises a snapshot of particle swarming from the authors' perspective, including variations in the algorithm, current and ongoing research, applications and open problems.

Keywords Particle swarms · Particle swarm optimization · PSO · Social networks · Swarm theory · Swarm dynamics · Real world applications

1 Introduction

The particle swarm paradigm, that was only a few years ago a curiosity, has now attracted the interest of researchers around the globe. This article is intended to give an overview of important work that gave direction and impetus to research in particle swarms as well as some interesting new directions and applications. Things change fast in this field as investigators discover new ways to do things, and new things to do with particle swarms. It is impossible to cover all aspects of this area within the strict page limits of this journal article. Thus this paper should be seen as a snapshot of the view we, these particular authors, have at the time of writing.

R. Poli (✉)
Department of Computing and Electronic Systems, University of Essex, Essex, UK
e-mail: rpoli@essex.ac.uk

J. Kennedy
Washington, DC, USA
e-mail: kennedy.jim@gmail.com

T. Blackwell
Department of Computing, Goldsmiths College, London, UK
e-mail: t.blackwell@gold.ac.uk

The article is organized as follows. In Sect. 2, we explain what particle swarms are and we look at the rules that control their dynamics. In Sect. 3, we consider how different types of social networks influence the behavior of swarms. In Sect. 4, we review some interesting variants of particle swarm optimization. In Sect. 5, we summarize the main results of theoretical analyses of the particle swarm optimizers. Section 6 looks at areas where particle swarms have been successfully applied. Open problems in particle swarm optimization are listed and discussed in Sect. 7. We draw some conclusions in Sect. 8.

2 Population dynamics

2.1 The original version

The initial ideas on particle swarms of Kennedy (a social psychologist) and Eberhart (an electrical engineer) were essentially aimed at producing computational intelligence by exploiting simple analogues of social interaction, rather than purely individual cognitive abilities. The first simulations (Kennedy and Eberhart 1995) were influenced by Heppner and Grenander's work (Heppner and Grenander 1990) and involved analogues of bird flocks searching for corn. These soon developed (Kennedy and Eberhart 1995; Eberhart and Kennedy 1995; Eberhart et al. 1996) into a powerful optimization method—Particle Swarm Optimization (PSO).[1]

In PSO a number of simple entities—the particles—are placed in the search space of some problem or function, and each evaluates the objective function at its current location.[2] Each particle then determines its movement through the search space by combining some aspect of the history of its own current and best (best-fitness) locations with those of one or more members of the swarm, with some random perturbations. The next iteration takes place after all particles have been moved. Eventually the swarm as a whole, like a flock of birds collectively foraging for food, is likely to move close to an optimum of the fitness function.

Each individual in the particle swarm is composed of three D-dimensional vectors, where D is the dimensionality of the search space. These are the current position $\vec{x}_i$, the previous best position $\vec{p}_i$, and the velocity $\vec{v}_i$.

The current position $\vec{x}_i$ can be considered as a set of coordinates describing a point in space. On each iteration of the algorithm, the current position is evaluated as a problem solution. If that position is better than any that has been found so far, then the coordinates are stored in the second vector, $\vec{p}_i$. The value of the best function result so far is stored in a variable that can be called $pbest_i$ (for "previous best"), for comparison on later iterations. The objective, of course, is to keep finding better positions and updating $\vec{p}_i$ and $pbest_i$. New points are chosen by adding $\vec{v}_i$ coordinates to $\vec{x}_i$, and the algorithm operates by adjusting $\vec{v}_i$, which can effectively be seen as a step size.

The particle swarm is more than just a collection of particles. A particle by itself has almost no power to solve any problem; progress occurs only when the particles interact.

[1] Following standard practice, in this paper we use the acronym "PSO" also for Particle Swarm Optimizer. There is no ambiguity since in this second interpretation "PSO" is always preceded by a determiner (e.g., "a" or "the") or is used in the plural form "PSOs".

[2] In PSO the objective function is often minimized and the exploration of the search space is not through evolution. However, following a widespread practice of borrowing from the evolutionary computation field, in this article we use the terms objective function and fitness function interchangeably.

Problem solving is a population-wide phenomenon, emerging from the individual behaviors of the particles through their interactions. In any case, populations are organized according to some sort of communication structure or topology, often thought of as a social network. The topology typically consists of bidirectional edges connecting pairs of particles, so that if j is in i's neighborhood, i is also in j's. Each particle communicates with some other particles and is affected by the best point found by any member of its topological neighborhood. This is just the vector $\vec{p}_i$ for that best neighbor, which we will denote with $\vec{p}_g$. The potential kinds of population "social networks" are hugely varied, but in practice certain types have been used more frequently. Topologies will be discussed in detail in Sect. 3.

In the particle swarm optimization process, the velocity of each particle is iteratively adjusted so that the particle stochastically oscillates around $\vec{p}_i$ and $\vec{p}_g$ locations. The (original) process for implementing PSO is as in Algorithm 1.

Algorithm 1 Original PSO.

1: Initialize a population array of particles with random positions and velocities on D dimensions in the search space.
2: **loop**
3: For each particle, evaluate the desired optimization fitness function in D variables.
4: Compare particle's fitness evaluation with its $pbest_i$. If current value is better than $pbest_i$, then set $pbest_i$ equal to the current value, and $\vec{p}_i$ equal to the current location $\vec{x}_i$ in D-dimensional space.
5: Identify the particle in the neighborhood with the best success so far, and assign its index to the variable g.
6: Change the velocity and position of the particle according to the following equation (see notes below):

$$\begin{cases} \vec{v}_i \leftarrow \vec{v}_i + \vec{U}(0, \phi_1) \otimes (\vec{p}_i - \vec{x}_i) + \vec{U}(0, \phi_2) \otimes (\vec{p}_g - \vec{x}_i), \\ \vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i. \end{cases} \tag{1}$$

7: If a criterion is met (usually a sufficiently good fitness or a maximum number of iterations), exit loop.
8: **end loop**

Notes:

– $\vec{U}(0, \phi_i)$ represents a vector of random numbers uniformly distributed in $[0, \phi_i]$ which is randomly generated at each iteration and for each particle.
– $\otimes$ is component-wise multiplication.
– In the original version of PSO, each component of $\vec{v}_i$ is kept within the range $[-V_{\max}, +V_{\max}]$ (see Sect. 2.2).

2.2 Parameters

The basic PSO described above has a small number of parameters that need to be fixed. One parameter is the size of the population. This is often set empirically on the basis of the dimensionality and perceived difficulty of a problem. Values in the range 20–50 are quite common.

The parameters ϕ_1 and ϕ_2 in (1) determine the magnitude of the random forces in the direction of personal best $\vec{p}_i$ and neighborhood best $\vec{p}_g$. These are often called acceleration

coefficients. The behavior of a PSO changes radically with the value of ϕ_1 and ϕ_2. Interestingly, we can interpret the components $\vec{U}(0,\phi_1)\otimes(\vec{p}_i-\vec{x}_i)$ and $\vec{U}(0,\phi_2)\otimes(\vec{p}_g-\vec{x}_i)$ in (1) as attractive forces produced by springs of random stiffness, and we can approximately interpret the motion of a particle as the integration of Newton's second law. In this interpretation, $\phi_1/2$ and $\phi_2/2$ represent the mean stiffness of the springs pulling a particle. It is no surprise then that by changing ϕ_1 and ϕ_2 one can make the PSO more or less "responsive" and possibly even unstable, with particle speeds increasing without control. The value $\phi_1=\phi_2=2.0$, almost ubiquitously adopted in early PSO research, did just that. However, this is often harmful to the search and needs to be controlled. The technique originally proposed to do this was to bound velocities so that each component of $\vec{v}_i$ is kept within the range $[-V_{\max},+V_{\max}]$. The choice of the parameter $V_{\max}$ required some care since it appeared to influence the balance between exploration and exploitation.

The use of hard bounds on velocity, however, presents some problems. The optimal value of $V_{\max}$ is problem-specific, but no reasonable rule of thumb is known. Further, when $V_{\max}$ was implemented, the particle's trajectory failed to converge. Where one would hope to shift from the large-scale steps that typify exploratory search to the finer, focused search of exploitation, $V_{\max}$ simply chopped off the particle's oscillations, so that some hopefully satisfactory compromise was seen throughout the run.

2.3 Inertia weight

Motivated by the desire to better control the scope of the search, reduce the importance of $V_{\max}$, and perhaps eliminate it altogether, the following modification of the PSO's update equations was proposed (Shi and Eberhart 1998b):

$$\begin{cases}\vec{v}_i \leftarrow \omega\vec{v}_i+\vec{U}(0,\phi_1)\otimes(\vec{p}_i-\vec{x}_i)+\vec{U}(0,\phi_2)\otimes(\vec{p}_g-\vec{x}_i),\\ \vec{x}_i \leftarrow \vec{x}_i+\vec{v}_i,\end{cases} \tag{2}$$

where ω was termed the "inertia weight." Note that if we interpret $\vec{U}(0,\phi_1)\otimes(\vec{p}_i-\vec{x}_i)+\vec{U}(0,\phi_2)\otimes(\vec{p}_g-\vec{x}_i)$ as the external force, $\vec{f}_i$, acting on a particle, then the change in a particle's velocity (i.e., the particle's acceleration) can be written as $\Delta\vec{v}_i=\vec{f}_i-(1-\omega)\vec{v}_i$. That is, the constant $1-\omega$ acts effectively as a friction coefficient, and so ω can be interpreted as the fluidity of the medium in which a particle moves. This perhaps explains why researchers have found that the best performance could be obtained by initially setting ω to some relatively high value (e.g., 0.9), which corresponds to a system where particles move in a low viscosity medium and perform extensive exploration, and gradually reducing ω to a much lower value (e.g., 0.4), where the system would be more dissipative and exploitative and would be better at homing into local optima. It is even possible to start from values of $\omega>1$, which would make the swarm unstable, provided that the value is reduced sufficiently to bring the swarm in a stable region (the precise value of ω that guarantees stability depends on the values of the acceleration coefficients, see Sect. 5).

Naturally, other strategies can be adopted to adjust the inertia weight. For example, in (Eberhart and Shi 2000) the adaptation of ω using a fuzzy system was reported to significantly improve PSO performance. Another effective strategy is to use an inertia weight with a random component, rather than time-decreasing. For example, (Eberhart and Shi 2001) successfully used $\omega=U(0.5,1)$. There are also studies, e.g., (Zheng et al. 2003), in which an increasing inertia weight was used obtaining good results.

With (2) and an appropriate choice of ω and of the acceleration coefficients, ϕ_1 and ϕ_2, the PSO can be made much more stable,[3] so much so that one can either do without $V_{\max}$ or can set $V_{\max}$ to a much higher value, such as the value of the dynamic range of each variable (on each dimension). In this case, $V_{\max}$ may improve performance, though with use of inertia or constriction (see Sect. 2.4) techniques, it is no longer necessary for damping the swarm's dynamics.

2.4 Constriction coefficients

Though the earliest researchers recognized that some form of damping of the dynamics of a particles (e.g., $V_{\max}$) was necessary, the reason for this was not understood. But when the particle swarm algorithm is run without restraining velocities in some way, these rapidly increase to unacceptable levels within a few iterations. Kennedy (1998) noted that the trajectories of nonstochastic one-dimensional particles contained interesting regularities when $\phi_1+\phi_2$ was between 0.0 and 4.0. Clerc's analysis of the iterative system (see Sect. 5) led him to propose a strategy for the placement of "constriction coefficients" on the terms of the formulas; these coefficients controlled the convergence of the particle and allowed an elegant and well-explained method for preventing explosion, ensuring convergence, and eliminating the arbitrary $V_{\max}$ parameter. The analysis also takes the guesswork out of setting the values of ϕ_1 and ϕ_2.

Clerc and Kennedy (2002) noted that there can be many ways to implement the constriction coefficient. One of the simplest methods of incorporating it is the following:

$$\begin{cases} \vec{v}_i \leftarrow \chi(\vec{v}_i + \vec{U}(0,\phi_1) \otimes (\vec{p}_i - \vec{x}_i) + \vec{U}(0,\phi_2) \otimes (\vec{p}_g - \vec{x}_i)), \\ \vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i, \end{cases} \tag{3}$$

where $\phi = \phi_1 + \phi_2 > 4$ and

$$\chi = \frac{2}{\phi - 2 + \sqrt{\phi^2 - 4\phi}}. \tag{4}$$

When Clerc's constriction method is used, ϕ is commonly set to 4.1, $\phi_1 = \phi_2$, and the constant multiplier χ is approximately 0.7298. This results in the previous velocity being multiplied by 0.7298 and each of the two $(\vec{p} - \vec{x})$ terms being multiplied by a random number limited by $0.7298 \times 2.05 \approx 1.49618$.

The constricted particles will converge without using any $V_{\max}$ at all. However, subsequent experiments and applications (Eberhart and Shi 2000) concluded that a better approach to use as a prudent rule of thumb is to limit $V_{\max}$ to $X_{\max}$, the dynamic range of each variable on each dimension, in conjunction with (3) and (4). The result is a particle swarm optimization algorithm with no problem-specific parameters.[4] And this is the canonical particle swarm algorithm of today.

Note that a PSO with constriction is algebraically equivalent to a PSO with inertia. Indeed, (2) and (3) can be transformed into one another via the mapping $\omega \leftrightarrow \chi$ and

[3] As discussed in Sect. 5, the PSO parameters cannot be set in isolation.

[4] In principle, the population size and topology (see Sect. 3) are parameters that still need to be set before one can run the PSO. However, practitioners tend to keep these constant across problems, although there is some evidence to suggest that bigger populations are better for higher dimensional problems and that highly-connected topologies work better for unimodal problems, while more sparsely-connected topologies are superior on multimodal problems.

$\phi_i \leftrightarrow \chi\phi_i$. So, the optimal settings suggested by Clerc correspond to $\omega = 0.7298$ and $\phi_1 = \phi_2 = 1.49618$ for a PSO with inertia.

2.5 Fully informed particle swarm

In the standard version of PSO, the effective sources of influence are in fact only two: self and best neighbor. Information from the remaining neighbors is unused. Mendes has revised the way particles interact with their neighbors (Kennedy and Mendes 2002; Mendes et al. 2002, 2003). Whereas in the traditional algorithm each particle is affected by its own previous performance and the single best success found in its neighborhood, in Mendes' fully informed particle swarm (FIPS), the particle is affected by all its neighbors, sometimes with no influence from its own previous success. FIPS can be depicted as follows:

$$\begin{cases} \vec{v}_i \leftarrow \chi\left(\vec{v}_i + \dfrac{1}{K_i}\displaystyle\sum_{n=1}^{K_i} \vec{U}(0,\phi) \otimes (\vec{p}_{nbr_n} - \vec{x}_i)\right), \\ \vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i, \end{cases} \tag{5}$$

where K_i is the number of neighbors for particle i, and nbr_n is i's nth neighbor. It can be seen that this is the same as the traditional particle swarm if only the self and neighborhood best in a $K_i = 2$ model are considered. With good parameters, FIPS appears to find better solutions in fewer iterations than the canonical algorithm, but it is much more dependent on the population topology.

3 Population topology

The first particle swarms (Kennedy and Eberhart 1995) evolved out of bird-flocking simulations of a type described by (Reynolds 1987) and (Heppner and Grenander 1990). In these models, the trajectory of each bird's flight is modified by application of several rules, including some that take into account the birds that are nearby in physical space. So, early PSO topologies were based on proximity in the search space. However, besides being computationally intensive, this kind of communication structure had undesirable convergence properties. Therefore, this Euclidean neighborhood was soon abandoned.

3.1 Static topologies

The next topology to be introduced, the *gbest* topology (for "global best"), was one where the best neighbor in the entire population influenced the target particle. While this may be conceptualized as a fully connected graph, in practice it only meant that the program needed to keep track of the best function result that had been found, and the index of the particle that found it.

The *gbest* is an example of static topology, i.e., one where neighbors and neighborhoods do not change during a run. The *lbest* topology (for "local best") is another static topology. This was introduced in (Eberhart and Kennedy 1995). It is a simple ring lattice where each individual was connected to $K = 2$ adjacent members in the population array, with toroidal wrapping (naturally, this can be generalized to $K > 2$). This topology had the advantage of allowing parallel search, as subpopulations could converge in diverse regions of the search space. Where equally good optima were found, it was possible for the population to stabilize

with particles in the good regions, but if one region was better than another, it was likely to attract particles to itself. Thus this parallel search resulted in a more thorough search strategy; though it converged more slowly than the *gbest* topology, *lbest* was less vulnerable to the attraction of local optima.

Several classical communications structures from social psychology (Bavelas 1950), including some with small-world modifications (Watts and Strogatz 1998), were experimented with in (Kennedy 1999). Circles, wheels, stars, and randomly-assigned edges were tested on a standard suite of functions. The most important finding was that there were important differences in performance depending on the topology implemented; these differences depended on the function tested, with nothing conclusively suggesting that any one was generally better than any other.

Numerous aspects of the social-network topology were tested in (Kennedy and Mendes 2002). For instance, the effect of including the target particle in its neighborhood (as opposed to allowing only "external" influences) was evaluated, finding, surprisingly, that whether or not the particle belonged to its neighborhood had little impact on behavior. 1343 random graphs were generated and then modified to meet certain criteria, including mean degree, clustering, and the standard deviations of those two measures to introduce homogeneous and heterogeneous structures. Several regular topologies were included, as well; these were the *gbest* and *lbest* versions mentioned above, as well as a von Neumann topology which defined neighborhoods on a grid, a "pyramid" topology, and a hand-made graph with clusters of nodes linked sparsely. Results from five test functions were combined in the analysis.

One finding that emerged was the relative superiority of the von Neumann structure. This topology possesses some of the parallelism of *lbest*, yet nodes have degree $K = 4$; thus the graph is more densely connected than *lbest*, but less densely than *gbest*.

Perhaps the most thorough exploration of the effect of topology to date is Mendes' doctoral thesis (Mendes 2004), from which the previous paper was derived. The research reported there provided several important insights. As we mentioned in Sect. 2.5, Mendes developed a different rule for particle swarm interactions, FIPS, where particles use a stochastic average of all their neighbors' previous best positions, rather than selecting the best neighbor.

While apparently superior results were found with the FIPS version, it was noted that the effect of the population topology was entirely different in FIPS and in the canonical (best-neighbor) version. For instance, whereas the *gbest* topology in the standard PSO meant that each individual received the best information known to any member of the population, in FIPS it meant that every particle received information about the best solutions found by *all* members of the population; thus, *gbest* behavior could be described as nearly random, as the particles are influenced by numerous and diverse attractors.

An important lesson from (Mendes 2004) is that the meaning of the social network and the effects of its topology depend on the mode of the interactions of the particles. Another important lesson had to do with the standard used to measure good performance. Mendes looked at two measures. The best function result was a measure of local search—the swarm's ability to find the bottom of a gradient. He also examined the proportion of trials in which a criterion was met; the criteria for the functions came from the literature and typically indicated that the globally optimal region of the search space had been discovered, even if its best point was not found. Proportion of successes then is a measure of global search ability.

Surprisingly, there was no evidence of correlation between those two measures. One topology × interaction-mode version might be good at local search, another might be good at global search, and one small group was good at both (these were FIPS particles with mean

degree $\bar{K} \in (4 \ldots 4.5)$). The two measures, in general, were independent. Thus, understanding the effects of topology on performance requires taking into account the way that the particle interacts with its neighborhood—FIPS and canonical best-neighbor are just two of a number of possible modes—and what measure of performance is to be used. The topologies that produced local search were very different from those that produced good ability to search for the global optimum.

In the canonical PSO, topologies are static. However, the work described above suggests that an adaptive topology might be beneficial, and several researchers have worked on this.

3.2 Dynamic topologies

In (Suganthan 1999) it was suggested that, as the *lbest* topology seemed better for exploring the search space while *gbest* converged faster, it might be a good idea to begin the search with an *lbest* ring lattice and slowly increase the size of the neighborhood, until the population was fully connected by the end of the run. That paper also reported results on another kind of topology, where neighbors were defined by proximity in the search space and the number of neighbors was dynamically increased through the course of the run. The authors reported some improvement using the neighborhood operator, though published results did not make the nature of the improvement clear.

Peram et al. (2003) used a weighted Euclidean distance in identifying the interaction partner for a particle. For each vector element, they find the particle that has the highest "fitness distance ratio" (FDR), e.g., the ratio of the difference between the target particle's fitness and the neighbor's fitness to the distance between them in the search space on that dimension. Their algorithm used this and *gbest* to influence the particle. The FDR ratio ensures that the selected neighbor is a good performer and lessens the probability that a particle interacts with a neighbor in a distant region of the search space. Note: there is no assurance that the same neighbor will be selected for each dimension. The new algorithm outperformed the standard PSO on a set of test functions.

Liang and Suganthan (2005) created random subpopulations of size n and occasionally randomized all the connections. Good results were obtained, especially on multimodal problems, with $n = 3$, re-structuring every 5 iterations.

Janson and Middendorf (2005) arranged the particles in a dynamic hierarchy, with each particle being influenced by its own previous success and that of the particle directly above it. Particles with better performance are moved up the hierarchy; the result is that they have more effect on poorer particles. The result was improved performance on most of the benchmark problems considered.

Clerc (2006b) developed a parameter-free particle swarm system called TRIBES, in which details of the topology, including the size of the population, evolve over time in response to performance feedback. The population is divided in subpopulations, each maintaining its own order and structure. "Good" tribes may benefit by removal of their weakest member, as they already possess good problem solutions and thus may afford to reduce their population; "bad" tribes, on the other hand, may benefit by addition of a new member, increasing the possibility of improvement. New particles are randomly generated. Clerc reevaluates and modifies the population structure every $L/2$ iterations, where L is the number of links in the population. In the context of the many modifications to the particle swarm that comprise the unique TRIBES paradigm, Clerc reports good results on a number of test functions.

4 PSO variants and specializations

Many tweaks and adjustments have been made to the basic algorithm over the past decade. Some have resulted in improved general performance, and some have improved performance on particular kinds of problems. Unfortunately, even focusing on the PSO variants and specializations that, from our point of view, have had the most impact and seem to hold the most promise for the future of the paradigm, these are too numerous for us to describe every one of them. So, we have been forced to leave out some streams in particle swarm research.

4.1 Binary particle swarms

Kennedy and Eberhart (1997) described a very simple alteration of the canonical algorithm that operates on bit-strings rather than real numbers. In their version, the velocity is used as a probability threshold to determine whether x_{id}—the dth component of $\vec{x}_i$—should be evaluated as a zero or a one. They squashed x_{id} in a logistic function

$$s(x_{id}) = \frac{1}{1 + \exp(-x_{id})}, \tag{6}$$

then generated a random number r for each bit-string site and compared it to $s(x_{id})$. If r was less that the threshold, then x_{id} was interpreted as 1, otherwise as 0.

Kennedy and Spears (1998) compared this binary particle swarm to several kinds of GAs, using Spears' multimodal random problem generator. This paradigm allows the creation of random binary problems with some specified characteristics, e.g., number of local optima, dimension, etc. In that study, the binary particle swarm was the only algorithm that found the global optimum on every single trial, regardless of problem features. It also progressed faster than GAs with crossover, mutation, or both, on all problems except the very simplest ones, with low dimension and a small number of local optima; the mutation-only GA was slightly faster in those cases.

Several other researchers have proposed alterations to the particle swarm algorithm to allow it to operate on binary spaces. Agrafiotis and Cedeño (2002) used the locations of the particles as probabilities to select features in a pattern-matching task. Each feature was assigned a slice of a roulette wheel based on its floating-point value, which was then discretized to {0, 1}, indicating whether the feature was selected or not. Mohan and Al-Kazemi (2001) suggested several ways that the particle swarm could be implemented on binary spaces. One version, which he calls the "regulated discrete particle swarm," performed very well on a suite of test problems. In Pamparä et al. (2005), instead of directly encoding bit strings in the particles, each particle stored the small number of coefficients of a trigonometric model (angle modulation) which was then run to generate bit strings.

Extending PSO to more complex combinatorial search spaces is also of great interest. The difficulty there is that notions of velocity and direction have no natural extensions for TSP tours, permutations, schedules, etc. Nonetheless, progress has recently been made (Clerc 2004, 2006b; Moraglio et al. 2007) but it is too early to say if PSO can be competitive in these spaces.

4.2 Dynamic problems

Dynamic problems are challenging for PSO. These are typically modeled by fitness functions which change over time, rendering particle memory obsolete (Hu and Eberhart 2001).

Parsopoulos and Vrahatis (2001) showed that the particle swarm could track slowly moving optima without any changes at all. Eberhart and Shi (2001) made a slight adjustment to the inertia by randomizing the inertia weight between 0.5 and 1.0. The idea is that when tracking a single dynamic optimum, it can not be predicted whether exploration (a larger inertia weight) or exploitation (a smaller inertia weight) will be better at any given time.

However, in many cases more specialized changes are required to handle dynamic problems. The problem can be split into two: how to detect change and how to respond. Carlisle and Dozier, Carlisle and Dozier (2000, 2001) occasionally re-evaluate the previous best of a single particle. Change is indicated by a different function value upon re-evaluation. They tried two responses to change. In the first, the current position was set to be the previous best and in the second, and more successful response, previous bests are compared with current positions, and memory is updated accordingly. Hu and Eberhart (2002) used a similar scheme for change detection but, as a response, randomized the entire swarm in the search space.

As an alternative to these strategies, a level of diversity can be maintained throughout the run. Parsopoulos and Vrahatis (2004) use repulsion to keep particles away from detected optima. The authors of (Blackwell and Bentley 2002) introduced charged particles into the swarm. These particles mutually repel and orbit a converging nucleus of 'neutral' particles, in analogy with a (picture of) an atom. Charged swarms can detect optimum shifts within their orbit and are therefore able to track quite severe changes. Diversity can also be maintained with a grid-like local neighborhood (Li and Dam 2003) and using hierarchical structures (Janson and Middendorf 2004).

Dynamic multi-modal landscapes are especially challenging for PSO (Blackwell and Branke 2006). The strategies mentioned above—restart and diversity enhancement—are less successful in situations where function peaks can vary in height as well as location because a small change in peak height might entail a large change in the position of the global optimum. In these cases, multi-population approaches have proven to be beneficial. The idea behind multi-swarm models is to position swarms on different peaks so that, should a sub-optimal peak become optimal, a swarm will be ready to immediately begin re-optimizing. Parrot and Li (2006) adjust the size and number of swarms dynamically by ordering the particles into 'species' in a technique called clearing. Blackwell and Branke (2006), borrowing from the atomic analogy referred to above, invoke an exclusion principle to prevent swarms from competing on the same peak and an anti-convergence measure to maintain diversity of the multi-swarm as a whole. Recently, a self-adapting multi-swarm has been derived (Blackwell 2007). The multi-swarm with exclusion has been favorably compared, on the moving peaks problem, to the hierarchical swarm, PSO re-initialization and a state-of-the-art dynamic-optimization evolutionary algorithm known as self-organizing scouts.

4.3 Noisy functions

Noisy fitness functions are important since they are often encountered in real-world problems. In these problems, unlike dynamic problems where the fitness function changes over time, the fitness function remains the same. However, its evaluation is noisy. Therefore, if a PSO explores the same position more than once, the fitness values associated to each evaluation may differ.

Parsopoulos and Vrahatis (2001) studied the behavior of the PSO when Gaussian distributed random noise was added to the fitness function and random rotations of the search space were performed. Experimental results indicated that the PSO remained effective in the presence of noise, and, in some cases, noise even helped the PSO avoid being trapped in local optima.

Pugh et al. (2005) compared the PSO to a noise-resistant variant where the main PSO loop was modified so that multiple evaluations of the same candidate solution are aggregated to better assess the actual fitness of this particular solution. The comparison considered several numerical problems with added noise, as well as unsupervised learning of obstacle avoidance using one or more robots. The noise-resistant PSO showed considerably better performance than the original.

4.4 Hybrids and adaptive particle swarms

Several investigators have attempted to adapt PSO parameters in response to information from the environment. Techniques from evolutionary computation and other methods have been borrowed by particle swarm researchers as well.

Angeline (1998) produced one of the first intentionally hybridized particle swarms. In his model, selection was applied to the particle population; "good" particles were reproduced with mutation, and "bad" particles were eliminated. Angeline's results showed that PSO could benefit from this modification.

Miranda and Fonseca (2002) borrowed an idea from evolution strategies. In that paradigm, points are perturbed by the addition of random values distributed around a mean of zero; the variance of the distribution is evolved along with function parameters. Those researchers used Gaussian random values to perturb χ, ϕ_1, and ϕ_2, as well as the position of the neighborhood best—but not the individual best—using selection to adapt the variance. The evolutionary self-adapting particle swarm optimization method, a hybrid of PSO and evolutionary methods, has shown excellent performance in comparison to some standard particle swarm methods. Miranda has used it for the manufacture of optical filters as well as in the optimization of power systems (Miranda and Fonseca 2002).

Løøvbjerg et al. (2001) use "breeding", borrowed from genetic algorithms, in a recent particle swarm study. Some selected particles were paired at random, and both positions and velocities were calculated from weighted arithmetic averages of the selected particles' parameters. Those researchers also divided the particle swarm into subpopulations in order to increase diversity, with some probability that individuals would breed within their own subpopulation or with a member of another. Results were encouraging, though the model as reported was not clearly superior to standard PSO or GA.

Wei et al. (2002) took a different tack, embedding velocity information in an evolutionary algorithm. They replaced Cauchy mutation with a version of PSO velocity in a fast evolutionary programming (FEP) algorithm, to give the FEP population direction. Their published results indicate that the approach is very successful on a range of functions; the new algorithm found global optima in tens of iterations, compared to thousands for the FEP versions tested.

Robinson et al. (2002), trying to optimize a profiled corrugated horn antenna, noted that a GA improved faster early in the run, and PSO improved later. As a consequence of this observation, they hybridized the two algorithms by switching from one to the other after several hundred iterations. They found the best horn by going from PSO to GA (PSO-GA) and noted that the particle swarm by itself outperformed both the GA by itself and the GA-PSO hybrid, though the PSO-GA hybrid performed best of all. It appears from their result that PSO more effectively explores the search space for the best region, while GA is effective at finding the best point once the population has converged on a single region; this is consistent with other findings.

Krink and Løøvbjerg (2002) similarly alternated among several methods, but they allowed individuals in a population to choose whether to belong to a population of a genetic

algorithm, a particle swarm, or to become solitary hill-climbers. In their self-adaptive search method, an individual changed its stage after 50 iterations with no improvement. The population was initialized as PSO particles; the "LifeCycle" algorithm outperformed all three of the methods that comprised it. Krink and Løøvbjerg's graphs show interesting changes in the proportion of individuals in each state for various problems.

A hybrid between a PSO and a hill-climber was proposed by Poli and Stephens (2004) who considered swarms of particles sliding on a fitness landscape rather than flying over it. The method uses particles without memory and requires no book-keeping of personal best. Instead it uses the physics of masses and forces to guide the exploration of fitness landscapes. Forces include: gravity, springs, and friction. Gravity provides the ability to seek minima. Springs provide exploration. Friction slows down the search and focuses it.

Clerc's recent experiments (Clerc 2006b) have shown that adaptation of the constriction factor, population size, and number of neighbors can produce improved results. His studies found that best performance were obtained when all three of these factors are adapted during the course of the run. Clerc used three rules: (a) Suicide and generation: a particle kills itself when it is the worst in its neighborhood and generates a new copy of itself when it is the best; (b) Modifying the coefficient: good local improvement caused an increase in the constriction coefficient, while poor improvement caused its decrease; (c) Change in neighborhood: the locally best particle could reduce the number of its neighbors, while poorly performing particles could increase theirs. Adaptive changes were not made on every iteration, but only occasionally.

Vesterstrøm et al. (2002) borrowed the idea of division of labor from research on insect swarm algorithms. In their hybrid particle swarm model, individuals in the swarm were assigned, after some number of iterations without improvement, to conduct local search. Local search was implemented by placing a particle at the population's global best position with a new random velocity vector. The division of labor modification was intended to improve performance on unimodal problems; this improvement was seen, though performance on multimodal functions was not significantly improved.

Hendtlass (2001) proposed a hybridization of PSO with ant colony optimization (ACO) (Dorigo and Stützle 2004) but did not present any results. More recently, Holden and Freitas (2005) introduced a hybrid PSO/ACO algorithm for hierarchical classification. This was applied to the functional classification of enzymes, with very promising results.

Hendtlass (2001) combined PSO with differential evolution (DE) but with mixed results. While the hybridized algorithm did better than either PSO or DE on one multimodal problem, the particle swarm by itself tended to be faster and more robust than either DE or two version of hybrids that were tested. However, more recently, others, for example, Zhang and Xie (2003), have obtained more positive results.

A hybridization of PSO based on genetic programming (GP) was proposed (Poli et al. 2005b, 2005a) showing some promise. GP is used to evolve new laws for the control of particles' movement for specific classes of problems. The method has consistently provided PSOs that performed better than some standard reference PSOs in the problem class used for training, and, in some cases, also generalized outside that class.

A PSO that borrows from estimation of distribution algorithms has recently been proposed in (Iqbal and Montes de Oca 2006). In this approach, the swarm's collective memory is used to bias the particles' movement towards regions in the search space which are estimated to be promising and away from previously sampled low-quality regions. Experiments suggest that this PSO hybrid finds better solutions than the canonical PSO using fewer function evaluations.

4.5 PSOs with diversity control

Some researchers have noted a tendency for the swarm to converge prematurely on local optima. Several approaches have been implemented in order to correct for the decline of diversity as the swarm concentrates on a single optimum.

Loøvbjerg (2002) used self-organized criticality to help the PSO attain more diversity, making it less vulnerable to local optima. When two particles are too close to one another, a variable called the "critical value" is incremented. When it reaches the criticality threshold, the particle disperses its criticality to other particles that are near it and relocates itself.

Other researchers have attempted to diversify the particle swarm by preventing the particles' clustering too tightly in one region of the search space. Blackwell and Bentley (2002) collision-avoiding swarms achieve this by reducing the attraction of the swarm center. Krink et al. (2002) developed "spatially extended" particles, where each particle is conceptualized as being surrounded by a sphere of some radius. When the spatially extended particle collides with another, it bounces off.

Xie et al. (2002) added negative entropy to the particle swarm in order to discourage premature convergence (excessively rapid convergence towards a poor quality local optimum). In some conditions, they weighted the velocity and, in some conditions, the particle's location, by some random value, thereby obtaining a sort of "dissipative particle swarm."

4.6 Bare-bones PSO

In bare-bones formulations of PSO, Kennedy (2003) proposed to move particles according to a probability distribution rather than through the addition of velocity—a "velocity-free" PSO. Bare-bones seeks to throw light on the relative importance of particle motion and the neighborhood topology of the $\vec{p}_i$'s which inform this motion.

In a first study of bare-bones PSO (Kennedy 2003), the particle update rule was replaced with a Gaussian distribution of mean $(\vec{p}_i + \vec{p}_g)/2$ and standard deviation $|\vec{p}_i - \vec{p}_g|$. This idea was inspired by a plot of the distribution of positions attained by a single canonical PSO particle moving in one dimension under the influence of fixed $\vec{p}_i$ and $\vec{p}_g$. This empirical distribution resembled a bell curve centered at $(\vec{p}_i + \vec{p}_g)/2$. Theoretical studies (Sect. 5) also support this finding.

Gaussian bare-bones works quite well, imitating the performance of PSO on some problems, but proving less effective on others (see also the comparisons in Richer. and Blackwell 2006). On closer examination, what appears to be a bell curve actually has a kurtosis which increases with iteration (Kennedy 2004), and the distribution has fatter tails than Gaussian. It has been suggested that the origin of this lies in the production of "bursts of outliers" (Kennedy 2004).[5] The trigger for these bursts is unknown; however Kennedy discovered that if burst events are added by hand to Gaussian bare-bones, performance is improved. The conjecture, therefore, is that the fat tails in the position distribution of canonical PSO enhance the ability of the swarm to move from sub-optimal locations.

Following on from this result, Richer. and Blackwell (2006) replaced the Gaussian distribution on bare-bones with a Lévy distribution. The Lévy distribution is bell-shaped like the Gaussian but with fatter tails. The Lévy has a tunable parameter, α, which interpolates between the Cauchy distribution ($\alpha = 1$) and Gaussian ($\alpha = 2$). This parameter can be used

[5] It has recently come to light (Poli 2007b; Poli and Broomhead 2007) (see also Sect. 5.2) that the growth in the kurtosis of the sampling distribution of the canonical PSO is a form of instability. Whether the bursts of outliers are the cause of this or more simply a manifestation of this is unknown.

to control the fatness of the tails. In a series of trials, Richer and Blackwell found that Lévy bare-bones at $\alpha = 1.4$ reproduces canonical PSO behavior, a result which supports the above conjecture.

A statistical distribution also appears in canonical PSO; the $p - x$ terms are multiplied by a random number from a uniform distribution. This injection of noise is believed to be critical to the search properties of PSO. The uniform distributed spring constant was replaced by a Gaussian random variable in (Secrest and Lamont 2003) and by the Lévy distribution in (Richer. and Blackwell 2006). The Gaussian spring constants PSO performed worse than standard PSO in a nine-function test suite, but Lévy spring constants PSO produced excellent results (Richer. and Blackwell 2006). The explanation might lie at the tails again, where large spring constants induce big accelerations and move particles away from local optima.

5 Theoretical analyses

Despite its apparent simplicity, the PSO presents formidable challenges to those interested in understanding swarm intelligence through theoretical analyses. So, to date a fully comprehensive mathematical model of particle swarm optimization is still not available. There are several reasons for this.

Firstly, the PSO is made up of a large number of interacting elements (the particles). Although the nature of the elements and of the interactions is simple, understanding the dynamics of the whole is nontrivial. Secondly, the particles are provided with memory and the ability to decide when to update the memory. This means that from one iteration to the next a particle may be attracted towards a new $\vec{p}_i$ or a new $\vec{p}_g$ or both. Thirdly, forces are stochastic. This prevents the use of standard mathematical tools used in the analysis of dynamical systems. Fourthly, the behavior of the PSO depends crucially on the structure of the fitness function. However, there are infinitely many fitness functions, and so it is extremely difficult to derive useful results that apply to all, even when complete information about the fitness function is available.

Nonetheless, in the last few years, a number of theoretical advances have been made. We briefly review them in this section.

5.1 Deterministic models

Because of the aforementioned difficulties, researchers have often been forced to make simplifying assumptions in order to obtain models that could be studied mathematically. These include: isolated single individuals, search stagnation (i.e., no improved solutions are found), and absence of randomness (see Engelbrecht 2005 for a good summary). Although these simplifications allow a researcher to observe a particular aspect of the particle's dynamics, their effect on the system's behavior may be severe. So, the conclusions drawn from the resulting mathematical models tend to be approximate and need to be verified empirically with the real system.

Following Kennedy's graphical examinations of the trajectories of individual particles and their responses to variations in key parameters (Kennedy 1998), the first real attempt at providing a theoretical understanding of PSO was the "surfing the waves" model presented by Ozcan and Mohan (1998). The model was a simplified model of the type mentioned above, where the behavior of one particle, in isolation, in one dimension, in the absence of stochasticity, and during stagnation was considered. Also $\vec{p}_i$ and $\vec{p}_g$ were assumed to coincide, as is the case for the best particle in a neighborhood. No inertia, velocity clamping, nor

constriction were used. The work was extended in (Ozcan and Mohan 1999) where multiple multi-dimensional particles were covered by the model. All other conditions remained as in (Ozcan and Mohan 1998), except that $\vec{p}_i$ and $\vec{p}_g$ were not required to coincide. Under these assumptions, solutions for the trajectories of the particle were derived. They then studied how particle trajectories changed with ϕ_1 and ϕ_2 and found that the behavior of the PSO depends on the value of $\phi = \phi_1 + \phi_2$. For example, the model suggests that $\phi > 4$ leads to growth in the oscillations of the particles, which may eventually leave the region of interest in the search space. For $\phi < 4$, the model predicts that particles exhibit bounded periodic oscillations.

Similar assumptions were used by Clerc and Kennedy's model (Clerc and Kennedy 2002): one particle, one dimension, deterministic behavior, stagnation. Under these conditions, the particle is a discrete-time linear dynamical system. The dynamics of the state (position and velocity) of the particle can be determined by finding the eigenvalues and eigenvectors of the state transition matrix. The model, therefore, predicts that the particle will converge to equilibrium if the magnitude of the eigenvalues is smaller than 1. Since the eigenvalues of the system are functions of the parameters of the PSO, specifically ϕ, the model then can suggest which parameter settings would guarantee convergence. Clerc and Kennedy (2002) also defined a very general class of PSO-like systems, which are described by five parameters. The original PSO corresponds to a particular choice for these parameters, while a PSO with inertia corresponds to another. They then derived relationships between the parameters that would guarantee convergence by imposing that the eigenvalues of the generalized system always had magnitudes less than 1. This led to various classes of PSOs, including one class, the PSO in (3), which has now become almost a standard. In this PSO, a constriction coefficient is used to reduce the amplitude of velocities in such a way to guarantee convergence.

A similar approach was used by van den Bergh van den Bergh (2002), who, again, modeled one particle, with no stochasticity and during stagnation, although in this case the analysis considered explicitly a PSO with an inertia weight. As in previous work, van den Bergh provided an explicit solution for the trajectory of the particle and showed that the particles are attracted towards a fixed point corresponding to the weighted sum of neighborhood best and personal best (with weights ϕ_1 and ϕ_2). Arguments were also put forward as to why the analysis would be valid also in the presence of stochasticity. The work also suggested the possibility that particles may converge on a point that is not an optimum (whether global or local). This would imply that a PSO is not guaranteed to always behave as an optimizer. So, modified PSOs were proposed where additional sources of randomness and restarts would allow the PSO to home into local optima or even to visit multiple local optima until eventually a global optimum is found, given enough time.

A simplified model of particle was also studied by Yasuda et al. (2003) (the theory is also summarized in Iwasaki and Yasuda 2005). The assumptions were: one one-dimensional particle, stagnation, and absence of stochasticity. Inertia was included in the model. Again an eigenvalue analysis of the resulting dynamical system was performed with the aim of determining for what parameter settings the system is stable and what classes of behaviors are possible for a particle. Conditions for cyclic behavior were analyzed in detail.

Blackwell (2003a, 2005) investigated how the spatial extent (effectively a measure of diversity) of a particle swarm varies over time. A simplified swarm model was adopted which is an extension of the one by Clerc and Kennedy where more than one particle and more than one dimensions are allowed, so particles could interact in the sense that they could change their personal best. Constriction was included. Stochasticity was still not modeled. The analysis suggested that, in a swarm of noninteracting particles, spatial extent decreases

exponentially with time as $(\sqrt{\chi})^t$, while if interactions are possible, there is still exponential decrease but with a slightly different rate. The derivation of these results is based on numerous assumptions. However, empirical evidence (Blackwell 2003b, 2005) strongly confirms the findings. The results effectively extend the convergence proof in (van den Bergh 2002).

Brandstatter and Baumgartner (2002) drew an analogy between Clerc and Kennedy's model (Clerc and Kennedy 2002) and a damped mass-spring oscillator, making it possible to rewrite the model using the notions of damping factor and natural vibrational frequency. Like the original model, this model assumes one particle, one dimension, no randomness and stagnation.

Under the same assumptions as Clerc and Kennedy's model and following a similar approach, Trelea (2003) performed a lucid analysis of a 4-parameter family of particle models. The analysis revealed that, surprisingly, two of the parameters can be discarded without loss of generality. Trelea then identified regions in the space of the two remaining parameters where the model exhibits qualitatively different behaviors (stability, harmonic oscillations, or zigzagging behavior) and emphasized how different behaviors can be combined by properly setting the PSO parameters.

The dynamical system approach proposed by Clerc and Kennedy has recently been extended by Campana et al. (2006a, 2006b) who studied an extended PSO which effectively corresponds to FIPS (Kennedy and Mendes 2002; Mendes 2004) (Sect. 2.5). Under the assumption that no randomness is present, the resulting model is a discrete, linear, and stationary dynamical system. So, its trajectories in state space can be seen as the sum of two trajectories: the free response and the forced response. Campana et al. expressed the free and forced responses for the PSO. However, they were able to study in detail only the free response, since the forced response depends inextricably on the specific details of the fitness function. An eigenvalue analysis revealed what types of behaviors a particle can exhibit and for which parameters settings each behavior is obtained. An interesting proposal in this work is to use the predictions of the theory to choose initial positions and velocities for particles in such a way to guarantee as much orthogonality as possible in the particles' trajectories in the search space.

5.2 Modeling PSO's randomness

To better understand the behavior of the PSO during phases of stagnation, Clerc (2006a) analyzed the distribution of velocities of a particle controlled by the standard PSO update rule with inertia and *stochastic forces*. In particular, he was able to show that a particle's new velocity, $\vec{v}(t+1)$, is the sum of three components: a forward force $F\vec{v}(t)$, a backward force $-B\vec{v}(t-1)$, and noise $N(\vec{p}_i - \vec{p}_g)$, where F, B, and N are stochastic variables. Clerc studied the distributions of F, B, and N. These depend on the parameters of the PSO. For example, $E[F] = w - c^2 \ln(2)$ while $E[B] = 2w \ln(2)$, where w is the inertia factor, and c is an acceleration coefficient. By manipulating these distributions, it is possible to derive useful recipes to set such parameters. For example, to obtain an unbiased search Clerc imposed $E[F] = E[B]$.

Kadirkamanathan et al. (2006) were able to study the stability of particles *in the presence of stochasticity* by using Lyapunov stability analysis. They considered the behavior of a single particle—the swarm best—with inertia and during stagnation. By representing the particle as a nonlinear feedback system, they were able to apply a large body of knowledge from control theory. For example, they found the transfer function of the particle, proved observability and controllability, found a Lyapunov function for the system, and found sufficient conditions on the PSO parameters to guarantee convergence. Unfortunately, Lyapunov

theory is very conservative, and so the conditions found are extremely restrictive, effectively forcing the PSO to have little oscillatory behavior.

Rudolph (1996) proposed a generic theoretical model of EA. In this model, the EA is seen as a homogeneous Markov chain. The probabilistic modifications on the population caused by the genetic operators are represented by a stochastic kernel, which effectively represents the probability of state transitions for the system. Since an EA consists of a population of N individuals represented by the N-tuple $(x_1, \ldots, x_N)$, where the x_i belong to some domain $\mathcal{M}$ (e.g., $\mathcal{M} = \mathbb{R}$) for $i = 1, \ldots, N$, typically the state space is $E = \mathcal{M}^N$. However, nothing prevents E from being the state space for a PSO, $E = \mathbb{R}^{3N}$, nor the kernel to represent the state changes produced by attraction forces and inertia, as in the PSO update rules. So, in principle, Rudolph's model could be applied to exactly model PSOs in their full richness. Unfortunately, the complexity of the calculations involved in this theory make it hard to apply.

In very recent work (Poli et al. 2007), Poli et al. proposed a method to build discrete Markov chain models of continuous stochastic optimizers that can approximate them on arbitrary continuous problems to any precision. The idea is to discretize the objective function using a finite element method grid which produces corresponding distinct states in the search algorithm. The spirit of the approach is the same as in Rudolph's model, but the discretization makes it easy to compute the transition matrix for the system. Iterating the transition matrix gives precise information about the behavior of the optimizer at each generation, including the probability of finding global optima, the expected run time, etc. The approach was tested on a bare-bones PSO and other optimizers. In the case of the bare-bones PSO, it was able to model the real system, that is, a stochastic swarm with multiple interacting individuals, exploring *any fitness function*, thereby overcoming many of the limitations of previous theoretical models. Empirical results revealed that the predictions of the Markov chain are remarkably accurate.

Even more recently, a novel method was introduced, which allows one to exactly study the sampling distribution and its changes over any number of generations for the canonical PSO, FIPS, and other PSOs during stagnation (Poli and Broomhead 2007; Poli 2007b). This was achieved by first deriving and then studying the exact dynamic equations for the moments of the sampling distribution. Mean, variance, skewness, and kurtosis were studied in detail, and the regions in parameter spaces where these are stable were analytically determined.

5.3 Executable models

Except when exact models are possible, modeling requires the selection of a subset of features of the real system which the modeler considers important in determining the system's behavior. Depending on which features are picked, certain aspects of the system are put in the foreground, while others are hidden. The target of the models described in Sects. 5.1 and 5.2 was their mathematical treatability. However, models can also be studied through computer simulations. These *executable models* are an important new development because they allow the study of a system from radically different view points.

A model of this type was recently suggested to study the emergent coordinated behavior of population-based search algorithms, including PSOs (Poli et al. 2006b; Langdon and Poli 2006). The model is a modified spring mass model where masses can perceive the environment and generate external forces. As a result of the interactions, the population behaves like a single organism. The force controlling the single emergent organism is proportional to the gradient of a modified fitness function that is the result of applying a filtering kernel

to the original. A combination of theory and empirical evidence was used to show that this model fits quite well both PSOs and genetic algorithms and that, in both cases, low-pass filtering kernels are good explanations for search behavior.

Another executable model of PSO was proposed in (Poli et al. 2006a) which focused on how the structure of social networks and the nature of social interactions affect the behavior of PSOs. A particular aim was to investigate whether and how consensus is reached and groupings of individuals are formed in different settings and with different forms of interaction and leadership. To this end, the details of the dynamics and the optimum seeking behavior of PSOs were omitted from the model.

A PSO alone is not a fully specified system for which one could study the dynamics. Only when the PSO is coupled with a problem the system is fully specified and produces a behavior. Traditionally very little theoretical effort has been devoted to understanding how behavior is influenced by the fitness function. However, a recent executable model (Langdon and Poli 2005, 2007; Langdon et al. 2005) has been a rich source of information as to for which problems a PSO is expected to do better or worse than other optimizers. Instead of studying the performance of different optimizers on standard problems in the hope of finding an informative degree of difference, new problems that maximize the difference in performance between optimizers were evolved. This led to the identification of problems where PSOs can easily outperform other techniques (including differential evolution, covariance matrix adaptation evolution strategy, and Newton–Raphson) and vice versa. The evolved problems revealed new swarm phenomena, including deception.

6 Applications

Particle swarm optimization can be and has been used across a wide range of applications. Areas where PSOs have shown particular promise include multimodal problems and problems for which there is no specialized method available or all specialized methods give unsatisfactory results.

PSO applications are so numerous and diverse that a whole paper would be necessary just to review a subset of the most paradigmatic ones. Here we only have a limited space to devote to this topic. So, we will limit ourselves to listing the main application areas where PSOs have been successfully deployed.

We divide applications into 26 different categories, although many applications span more than one category. We have based our categorization on an analysis of over 1100 publications on particle swarm optimization stored, at the time of writing, in the IEEE Xplore database. Of these publications, around 350 are proposals for improvements and specializations of PSO. Around 700 papers (55 of which are journal papers) can be classified as applications of PSO, although many of these also involve the customization or extension of the method to best suit the specific application of interest. (The remaining entries in the database are conference proceedings books and other miscellaneous entries.) Because of the large number of applications reviewed, here we will provide our categorization without citations. The interested reader can, however, refer to (Poli 2007a) for a complete list of applications within each application area, an extensive list of references and a detailed description of how the categorization was obtained.

The main PSO application categories are as follows: image and video analysis applications (51 papers, approximately 7.6% of all application papers in the IEEE Xplore database); design and restructuring of electricity networks and load dispatching (48 papers, 7.1%); control applications (47 papers, 7.0%); applications in electronics and electromagnetics (39 papers, 5.8%); antenna design (39 papers, 5.8%); power generation and power systems (39

papers, 5.8%); scheduling (38 papers, 5.6%); design applications (30 papers, 4.4%); design and optimization of communication networks (30 papers, 4.4%); biological, medical, and pharmaceutical applications (29 papers, 4.3%); clustering, classification and data mining (29 papers, 4.3%); fuzzy and neuro-fuzzy systems and control (26 papers, 3.8%); signal processing (26 papers, 3.8%); neural networks (26 papers, 3.8%); combinatorial optimization problems (24 papers, 3.5%); robotics (23 papers, 3.4%); prediction and forecasting (20 papers, 2.9%); modeling (19, papers 2.8%); detection or diagnosis of faults and the recovery from them (16 papers, 2.3%); sensors and sensor networks (13 papers, papers 1.9%); applications in computer graphics and visualization (12, papers 1.7%); design or optimization of engines and electrical motors (10 papers, 1.4%); applications in metallurgy (9 papers, 1.3%); music generation and games (9 papers, 1.3%); security and military applications (9 papers, 1.3%); finance and economics (7 papers, 1.0%).

7 Open questions

The action of a PSO can be broken down into many steps for each of which there are several open questions regarding its practical extensions and realizations. We discuss these in Sects. 7.1–7.4. There are also numerous questions facing PSO theorists, which we discuss in Sect. 7.5.

7.1 Initialization and termination

Initialization consists of choosing swarm size, choosing particle positions randomly in the search space, where appropriate, and choosing random velocities. Even at this stage of the algorithm there are unresolved issues. There is controversy concerning an alleged origin seeking bias in PSO—whereby problems with a global optimum near the origin would be easier—and concerning the tendency of PSO to converge to optima lying within the initialization region if this is smaller than the search space.

Although the termination criteria (usually acceptable error and total number of function evaluations) are part of the problem definition, PSO can suffer, in some circumstances, from premature convergence and a slow-down of the swarm as it approaches optima. This situation arises when the swarm is to one side of the optimum and is moving as a coordinated body down the function gradient. The schemes in Sect. 4.5 partially alleviate these problems.

7.2 Particle selection, movement, and evaluation

In standard PSO, each particle is selected for updating in turn. A complete iteration corresponds to a single position update of each particle. However, in a more general swarm model, particles could be chosen at random or according to a selection scheme perhaps similar to parent selection in evolutionary computation. For example, better performing particles could be updated more frequently than others. Also, a particle might be given a number of iterations in which to adapt to a new attractor configuration. Or, perhaps, resources could be concentrated on badly performing particles, in order to improve the population as a whole. More research is needed to clarify these issues.

Both in the classical velocity-dependent PSOs and bare-bone type of PSOs (Sect. 4.6), a distinction is made between the memory of the best particle in the neighborhood and the particle's own memory. FIPS (Sect. 2.5) blurs this distinction by equally weighting the memories of all particles in the neighborhood. However, clearly there are many opportunities

for generalization. For example, the weighting given to each informer could be proportional to its fitness, or a ranking scheme could be employed.

There has been much exploration of particle dynamics, with many PSO variants borrowing from the rich range of particle interactions in physical systems. For example, researchers have explored the use of spherical particles, inter-particle repulsive forces, sliding (rather than flying) particles, and quantum effects. However, many other types of physical interactions remain unexplored. Also, whilst it does seem that the inclusion of randomness in the form of random force parameters, or sampling from a distribution, is useful, it is not known if stochasticity is strictly necessary for good performance. However, so far a completely deterministic algorithm has eluded researchers.

Finally, occasionally particles may move to regions at very large distances from the main swarm or even outside the search space. There are alternative actions to deal with outliers, one of which is not evaluating the particle. There are other alternatives (e.g., moving the particle to the edge of the space or re-initialization), but there is controversy as to which method is best, and much more research is needed on this topic.

7.3 Memory selection and update

In a PSO, each particle has an associated private memory—its own personal best. However, there is no need to require that each particle owns a specific portion of the PSO's memory. It is possible to imagine schemes where the size of PSO's memory is different from the population size. Also, each particle, in standard PSO, is informed by the same set of particles (its neighborhood). Since this neighborhood is fixed, the PSO memories themselves are part of a permanent social network. Once more this idea can be generalized. The memory network might adapt to take into account the performance of a particular memory, a subset of particles, and/or the swarm as a whole. Some of these dynamic adaptations of the "memory" population and of the sub-population as seen by a particle have been explored in Clerc's Tribes (Clerc 2006b). Other adaptations have been explored in a multi-swarm formulation (Blackwell 2007). However, far too little is known about the behavior of these adaptations.

Another deep issue in PSO concerns the very importance of particle movement. Perhaps the important aspect of PSO is the information flow into and out from the memories, rather than the details of how particles actually move to new positions. This issue will need to be clarified in future research.

7.4 Adaptation

Current forms of swarm adaptations include addition and removal of particles and tuning of any of the PSO's parameters. A worthy goal of PSO research is to find the completely adapting, parameter-free optimizer. For example, a multitude of swarms could run on a given problem, each one with different parameters, numbers of particles, informer topologies etc. The swarms could breed and produce progeny, as in an evolutionary algorithm, with survival of the better optimizers for the current problem. Alternatively one could use a series of sequential trials with parameter tuning between each trial. However, the number of function evaluations of such schemes would be very high. More research is needed to establish if PSO self-adaptation can be obtained with efficiency, ease of use, and simplicity of implementation.

Table 1 PSO publications and citations by year

Year	Google Scholar Hits	IEEE Xplore Publications
1995	(28)	(2)
1996	(14)	(0)
1997	(24)	(3)
1998	(49)	(4)
1999	(62)	(7)
2000	(78)	(8)
2001	(118)	(11)
2002	(256)	(46)
2003	(373)	(82)
2004	(850)	(181)
2005	(1210)	(279)
2006	(1230)	(462)

7.5 Theory

The questions in front of PSO theorists are not very different from those faced for other stochastic search algorithms. We highlight some challenges. How do the details of the equations of motion or the communications topology affect the dynamics of the PSO? How does the problem (fitness function) affect the dynamics of a PSO? How do we classify PSO/problem pairs and understand when the dynamics of two different systems should be qualitatively similar? On which classes of problems should a particular PSO be expected to provide good or bad performance? Theoretical studies have given recipes on how to set χ, ω, ϕ_1, and ϕ_2 so as to guarantee certain properties of the behavior. However, how should one set other important parameters such as the population size, the duration of runs, the initialization strategy, the distributions from which to draw random numbers, the social network topology, etc.?

8 Conclusions

In 2001, Kennedy and Eberhart wrote (Kennedy et al. 2001):

> "...we are looking at a paradigm in its youth, full of potential and fertile with new ideas and new perspectives... Researchers in many countries are experimenting with particle swarms... Many of the questions that have been asked have not yet been satisfactorily answered."

How has the situation evolved since then?

An analysis of IEEE Xplore and Google Scholar's citations and publications before 2007 is quite illuminating in that sense. As show in Table 1, particle swarm optimization is exponentially growing.[6] So, clearly, *we are still looking at a paradigm in its youth, full of potential and fertile with new ideas and new perspectives*. Researchers in many countries are experimenting with particle swarms and applying them in real-world applications. Many more questions have been asked, although still too few have been satisfactorily answered, so the quest goes on. Come and join the swarm!

[6]The search string "Particle Swarm Optimization" was used in Google Scholar and IEEE Xplore. Note that there are certain artifacts in Google Scholar's hits. For example, there were only two papers published in 1995. Also, the 2006 value is still not stable due to publication indexing delays.

Acknowledgements The authors would like to thank EPSRC (Extended Particle Swarms project, GR/T11234/01) for financial support. The anonymous reviewers, the associate editor, and the editor-in-chief are warmly thanked for their many useful suggestions for improving the manuscript.

References

Agrafiotis, D. K., & Cedeño, W. (2002). Feature selection for structure-activity correlation using binary particle swarms. *Journal of Medicinal Chemistry*, *45*(5), 1098–1107.

Angeline, P. (1998). Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In V. W. Porto, N. Saravanan, D. Waagen, & A. E. Eiben (Eds.), *Proceedings of evolutionary programming VII* (pp. 601–610). Berlin: Springer.

Bavelas, A. (1950). Communication patterns in task-oriented groups. *Journal of the Acoustical Society of America*, *22*, 271–282.

Blackwell, T. M. (2003a). Particle swarms and population diversity I: Analysis. In A. M. Barry (Ed.), *Proceedings of the bird of a feather workshops of the genetic and evolutionary computation conference (GECCO)* (pp. 103–107), Chicago. San Francisco: Kaufmann.

Blackwell, T. M. (2003b). Particle swarms and population diversity II: Experiments. In A. M. Barry (Ed.), *Proceedings of the bird of a feather workshops of the genetic and evolutionary computation conference (GECCO)* (pp. 108–112), Chicago. San Francisco: Kaufmann.

Blackwell, T. M. (2005). Particle swarms and population diversity. *Soft Computing*, *9*, 793–802.

Blackwell, T. M. (2007). Particle swarm optimization in dynamic environments. In S. Yand, Y. Ong, & Y. Jin (Eds.), *Evolutionary computation in dynamic environments* (pp. 29–49). Springer, Berlin. DOI 10.1007/978-3-540-49774-5-2.

Blackwell, T., & Bentley, P. J. (2002). Don't push me! Collision-avoiding swarms. In *Proceedings of the IEEE congress on evolutionary computation (CEC)* (pp. 1691–1696), Honolulu, HI. Piscataway: IEEE.

Blackwell, T. M., & Branke, J. (2006). Multi-swarms, exclusion and anti-convergence on dynamic environments. *IEEE Transactions on Evolutionary Computation*, *10*, 459–472.

Brandstatter, B., & Baumgartner, U. (2002). Particle swarm optimization-mass-spring system analogon. *IEEE Transactions on Magnetics*, *38*(2), 997–1000.

Campana, E. F., Fasano, G., & Pinto, A. (2006a). Dynamic system analysis and initial particles position in particle swarm optimization. In *Proceedings of the IEEE swarm intelligence symposium (SIS)*, Indianapolis. Piscataway: IEEE.

Campana, E. F., Fasano, G., Peri, D., & Pinto, A. (2006b). Particle swarm optimization: Efficient globally convergent modifications. In C. A. Mota Soares, et al. (Eds.), *Proceedings of the III European conference on computational mechanics, solids, structures and coupled problems in engineering*, Lisbon, Portugal.

Carlisle, A., & Dozier, G. (2000). Adapting particle swarm optimization to dynamic environments. In *Proceedings of international conference on artificial intelligence* (pp. 429–434), Las Vegas, NE.

Carlisle, A., & Dozier, G. (2001). *Tracking changing extrema with particle swarm optimizer*. Auburn University Technical Report CSSE01-08.

Clerc, M. (2004). Discrete particle swarm optimization, illustrated by the traveling salesman problem. In B. V. Babu & G. C. Onwubolu (Eds.), *New optimization techniques in engineering* (pp. 219–239). Berlin: Springer.

Clerc, M. (2006a). *Stagnation analysis in particle swarm optimization or what happens when nothing happens*. Technical Report CSM-460, Department of Computer Science, University of Essex, August 2006.

Clerc, M. (2006b). *Particle swarm optimization*. London: ISTE.

Clerc, M., & Kennedy, J. (2002). The particle swarm—explosion, stability, and convergence in a multidimensional complex space. *IEEE Transaction on Evolutionary Computation*, *6*(1), 58–73.

Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. Cambridge: MIT Press.

Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science* (pp. 39–43), Nagoya, Japan. Piscataway: IEEE.

Eberhart, R. C., & Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the IEEE congress on evolutionary computation (CEC)* (pp. 84–88), San Diego, CA. Piscataway: IEEE.

Eberhart, R. C., & Shi, Y. (2001). Tracking and optimizing dynamic systems with particle swarms. In *Proceedings of the IEEE congress on evolutionary computation (CEC)* (pp. 94–100), Seoul, Korea. Piscataway: IEEE.

Eberhart, R. C., Simpson, P. K., & Dobbins, R. W. (1996). *Computational intelligence PC tools*. Boston: Academic Press.

Engelbrecht, A. P. (2005). *Fundamentals of computational swarm intelligence*. Chichester: Wiley.

Hendtlass, T. (2001). A combined swarm differential evolution algorithm for optimization problems. In L. Monostori, J. Váncza & M. Ali (Eds.), *Lecture notes in computer science: Vol. 2070. Proceedings of the 14th international conference on industrial and engineering applications of artificial intelligence and expert systems (IEA/AIE)* (pp. 11–18), Budapest, Hungary. Berlin: Springer.

Heppner, H., & Grenander, U. (1990). A stochastic non-linear model for coordinated bird flocks. In S. Krasner (Ed.), *The ubiquity of chaos* (pp. 233–238). Washington: AAAS.

Holden, N., & Freitas, A. A. (2005). A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data. In *Proceedings of the IEEE swarm intelligence symposium (SIS)* (pp. 100–107). Piscataway: IEEE.

Hu, X., & Eberhart, R. C. (2001). Tracking dynamic systems with PSO: where's the cheese? In *Proceedings of the workshop on particle swarm optimization*. Purdue school of engineering and technology, Indianapolis, IN.

Hu, X., & Eberhart, R. C. (2002). Adaptive particle swarm optimization: detection and response to dynamic systems. In *Proceedings of the IEEE congress on evolutionary computation (CEC)* (pp. 1666–1670), Honolulu, HI. Piscataway: IEEE.

Iqbal, M., & Montes de Oca, M. A. (2006). An estimation of distribution particle swarm optimization algorithm. In M. Dorigo, L. M. Gambardella, M. Birattari, A. Martinoli, R. Poli & T. Stützle (Eds.), *Lecture notes in computer science: Vol. 4150. Proceedings of the fifth international workshop on ant colony optimization and swarm intelligence ANTS 2006* (pp. 72–83). Berlin: Springer.

Iwasaki, N., & Yasuda, K. (2005). Adaptive particle swarm optimization using velocity feedback. *International Journal of Innovative Computing, Information and Control*, *1*(3), 369–380.

Janson, S., & Middendorf, M. (2004). A hierarchical particle swarm optimizer for dynamic optimization problems. In G. R. Raidl (Ed.), *Lecture notes in computer science: Vol. 3005. Proceedings of evoworkshops 2004: 1st European workshop on evolutionary algorithms in stochastic and dynamic environments* (pp. 513–524), Coimbra, Portugal. Berlin: Springer.

Janson, S., & Middendorf, M. (2005). A hierarchical particle swarm optimizer and its adaptive variant. *IEEE Transactions on System Man and Cybernetics B*, *35*(6), 1272–1282.

Kadirkamanathan, V., Selvarajah, K., & Fleming, P. J. (2006). Stability analysis of the particle dynamics in particle swarm optimizer. *IEEE Transactions on Evolutionary Computation*, *10*(3), 245–255.

Kennedy, J. (1998). The behavior of particles. In V. W. Porto, N. Saravanan, D. Waagen & A. E. Eiben (Eds.), *Lecture notes in computer science. Evolutionary programming VII: proceedings of the 7-th annual conference on evolutionary programming* (pp. 581–589). San Diego, CA. Berlin: Springer.

Kennedy, J. (1999). Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In *Proceedings of the IEEE congress on evolutionary computation* (pp. 1931–1938). Piscataway: IEEE.

Kennedy, J. (2003). Bare bones particle swarms. In *Proceedings of the IEEE swarm intelligence symposium (SIS)* (pp. 80–87), Indianapolis, IN. Piscataway: IEEE.

Kennedy, J. (2004). Probability and dynamics in the particle swarm. In *IEEE congress on evolutionary computation (CEC04)* (pp. 340–347). Piscataway: IEEE.

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks IV* (pp. 1942–1948). Piscataway: IEEE.

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. In *Proceedings of the conference on systems, man, and cybernetics* (pp. 4104–4109). Piscataway: IEEE.

Kennedy, J., & Mendes, R. (2002). Population structure and particle swarm performance. In *Proceedings of the IEEE congress on evolutionary computation (CEC)* (pp. 1671–1676), Honolulu, HI. Piscataway: IEEE.

Kennedy, J., & Spears, W. M. (1998). Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. In *Proceedings international conference on evolutionary computation* (pp. 78–83). Piscataway: IEEE.

Kennedy, J., Eberhart, R. C., & Shi, Y. (2001). *Swarm intelligence*. San Francisco: Kaufmann.

Krink, T., & Løovbjerg, M. (2002). The LifeCycle model: combining particle swarm optimization, genetic algorithms and hillclimbers. In *Lecture notes in computer science. Proceedings of parallel problem solving from nature (PPSN)* (pp. 621–630). Granada, Spain. Berlin: Springer.

Krink, T., Vesterstrøm, J. S., & Riget, J. (2002). Particle swarm optimization with spatial particle extension. In *Proceedings of the IEEE congress on evolutionary computation (CEC-2002)* (pp. 1474–1479). Piscataway: IEEE.

Langdon, W. B., & Poli, R. (2005). Evolving problems to learn about particle swarm and other optimisers. In D. Corne et al. (Eds.), *Proceedings of IEEE congress on evolutionary computation (CEC)* (pp. 81–88), Edinburgh, UK. Piscataway: IEEE.

Langdon, W. B., & Poli, R. (2006). Finding social landscapes for PSOs via kernels. In G. G. Yen, L. Wang, P. Bonissone, & S. M. Lucas (Eds.), *Proceedings of the 2006 IEEE congress on evolutionary computation (CEC)* (pp. 6118–6125), Vancouver, Canada. Piscataway: IEEE.

Langdon, W. B., & Poli, R. (2007, accepted for publication). Evolving problems to learn about particle swarm optimisers and other search algorithms. *IEEE Transaction on Evolutionary Computation*. DOI 10.1109/TEVC.2006.886448.

Langdon, W. B., Poli, R., Holland, O., & Krink, T. (2005). Understanding particle swarm optimization by evolving problem landscapes. In L. M. Gambardella, P. Arabshahi & A. Martinoli (Eds.), *Proceedings SIS 2005 IEEE swarm intelligence* (pp. 30–37), Pasadena, CA. Piscataway: IEEE.

Li, X., & Dam, K. H. (2003). Comparing particle swarms for tracking extrema in dynamic environments. In *Proceedings of the IEEE congress on evolutionary computation (CEC)* (pp. 1772–1779). Piscataway: IEEE.

Liang, J. J., & Suganthan, P. N. (2005). Dynamic multiswarm particle swarm optimizer (DMS-PSO). In *Proceedings of the IEEE swarm intelligence symposium (SIS)* (pp. 124–129). Piscataway: IEEE.

Løøvbjerg, M., & Krink, T. (2002). Extending particle swarms with self-organized criticality. In *Proceedings of the IEEE congress on evolutionary computation (CEC-2002)* (pp. 1588–1593). Piscataway: IEEE.

Løøvbjerg, M., Rasmussen, T. K., & Krink, T. (2001). Hybrid particle swarm optimiser with breeding and subpopulations. In *Proceedings of the third genetic and evolutionary computation conference (GECCO)* (pp. 469–476). San Francisco: Kaufmann.

Mendes, R. (2004). *Population topologies and their influence in particle swarm performance*. PhD thesis, Departamento de Informatica, Escola de Engenharia, Universidade do Minho, 2004.

Mendes, R., Cortes, P., Rocha, M., & Neves, J. (2002). Particle swarms for feedforward neural net training. In *Proceedings of the international joint conference on neural networks* (pp. 1895–1899), Honolulu, HI. Piscataway: IEEE.

Mendes, R., Kennedy, J., & Neves, J. (2003). Watch thy neighbor or how the swarm can learn from its environment. In *Proceedings of the IEEE swarm intelligence symposium (SIS)* (pp. 88–94). Piscataway: IEEE.

Miranda, V., & Fonseca, N. (2002). New evolutionary particle swarm algorithm (EPSO) applied to voltage/VAR control. In *Proceedings of the 14th power systems computation conference (PSCC)* (Session 21, Paper 5, pp. 1–6), Seville, Spain.

Mohan, C. K., & Al-Kazemi, B. (2001). Discrete particle swarm optimization. In *Proceedings of the workshop on particle swarm optimization*, Indianapolis, IN, Purdue School of Engineering and Technology, IUPUI.

Moraglio, A., Di Chio, C., & Poli, R. (2007). Geometric particle swarm optimization. In M. Ebner et al. (Eds.), *Lecture notes in computer science: Vol. 4445. Proceedings of the European conference on genetic programming (EuroGP)*. (pp. 125–136). Berlin: Springer.

Ozcan, E., & Mohan, C. K. (1998). Analysis of a simple particle swarm optimization system. *Intelligent Engineering Systems Through Artificial Neural Networks*, *8*, 253–258.

Ozcan, E., & Mohan, C. (1999). Particle swarm optimization: surfing the waves. In *Proceedings of the IEEE congress on evolutionary computation (CEC)* (pp. 1939–1944). Piscataway: IEEE.

Pamparä, G., Franken, N., & Engelbrecht, A. P. (2005). Combining particle swarm optimization with angle modulation to solve binary problems. In *Proceedings of the IEEE congress on evolutionary computation (CEC)* (pp. 225–239). Piscataway: IEEE.

Parrot, D., & Li, X. (2006). Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Transactions on Evolutionary Computation*, *10*, 440–458.

Parsopoulos, K. E., & Vrahatis, M. N. (2001). Particle swarm optimizer in noisy and continuously changing environments. In M. H. Hamza (Ed.), *Artificial intelligence and soft computing* (pp. 289–294). Anaheim: IASTED/ACTA.

Parsopoulos, K. E., & Vrahatis, M. N. (2004). On the computation of all global minimizers through particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, *8*, 211–224.

Peram, T., Veeramachaneni, K., & Mohan, C. (2003). Fitness-distance ratio based particle swarm optimization. In *Proceedings of the IEEE swarm intelligence symposium (SIS)* (pp. 174–181), Indianapolis, IN. Piscataway: IEEE.

Poli, R. (2007a). *An analysis of publications on particle swarm optimization applications*. Technical Report CSM-469, Department of Computer Science, University of Essex.

Poli, R. (2007b, forthcoming). On the moments of the sampling distribution of particle swarm optimisers. In *Proceedings of the workshop on particle swarm optimization: the second decade of the genetic and evolutionary computation conference (GECCO)*, London, July 2007. New York: ACM.

Poli, R., & Broomhead, D. (2007, forthcoming). Exact analysis of the sampling distribution for the canonical particle swarm optimiser and its convergence during stagnation. In *Genetic and evolutionary computation conference (GECCO)*, London. ACM, New York.

Poli, R., & Stephens, C. R. (2004). Constrained molecular dynamics as a search and optimization tool. In M. Keijzer et al. (Eds.), *Lecture notes in computer science: Vol. 3003. Proceedings of the 7th European conference on genetic programming (EuroGP)* (pp. 150–161), Coimbra, Portugal. Berlin: Springer.

Poli, R., Di Chio, C., & Langdon, W. B. (2005a). Exploring extended particle swarms: a genetic programming approach. In H.-G. Beyer, et al. (Eds.), *GECCO 2005: Proceedings of the 2005 conference on genetic and evolutionary computation* (pp. 169–176), Washington, DC. New York: ACM.

Poli, R., Langdon, W. B., & Holland, O. (2005b). Extending particle swarm optimization via genetic programming. In M. Keijzer et al. (Eds.), *Lecture notes in computer science: Vol. 3447. Proceedings of the 8th European conference on genetic programming* (pp. 291–300), Lausanne, Switzerland. Berlin: Springer.

Poli, R., Langdon, W. B., Marrow, P., Kennedy, J., Clerc, M., Bratton, D., & Holden, N. (2006a). Communication, leadership, publicity and group formation in particle swarms. In *Lecture notes in computer science: Vol. 4150. International workshop on Ant colony optimization and swarm intelligence (ANTS)* (pp. 132–143). Berlin: Springer.

Poli, R., Wright, A. H., McPhee, N. F., & Langdon, W. B. (2006b). Emergent behaviour, population-based search and low-pass filtering. In *Proceedings of the IEEE world congress on computational intelligence, IEEE congress on evolutionary computation (CEC)* (pp. 395–402), Vancouver. Piscataway: IEEE.

Poli, R., Langdon, W. B., Clerc, M., & Stephens, C. R. (2007, forthcoming). Continuous optimization theory made easy? Finite-element models of evolutionary strategies, genetic algorithms and particle swarm optimizers. In *Lecture notes in computer science. Proceedings of the foundations of genetic algorithms (FOGA) workshop*. Springer, Berlin, Germany. (Also available as Technical Report CSM-463, Department of Computer Science, University of Essex.)

Pugh, J., Martinoli, A., & Zhang, Y. (2005). Particle swarm optimization for unsupervised robotic learning. In *Proceedings of IEEE swarm intelligence symposium (SIS)* (pp. 92–99). Piscataway: IEEE.

Reynolds, C. W. (1987). Flocks, herds, and schools: a distributed behavioral model. *Computer Graphics*, *21*(4), 25–34.

Richer., T., & Blackwell, T. M. (2006). The Lévy particle swarm. In *Proceedings of IEEE congress on evolutionary computation* (pp. 3150–3157), Vancouver. Piscataway: IEEE.

Robinson, J., Sinton, S., & Rahmat-Samii, Y. (2002). Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna. In *Proceedings IEEE international symposium on antennas and propagation* (pp. 314–317), San Antonio, TX. Piscataway: IEEE.

Rudolph, G. (1996). Convergence of evolutionary algorithms in general search spaces. In *Proceedings of the IEEE international conference on evolutionary computation* (pp. 50–54), Nayoya University, Japan. Piscataway: IEEE.

Secrest, B., & Lamont, G. (2003). Visualizing particle swarm optimization—Gaussian particle swarm optimization. In *Proceedings of the IEEE swarm intelligence symposium (SIS)* (pp. 198–204). Piscataway: IEEE.

Shi, Y., & Eberhart, R. C. (1998b). A modified particle swarm optimizer. In *Proceedings of the IEEE international conference on evolutionary computation* (pp. 69–73). Piscataway: IEEE.

Suganthan, P. N. (1999). Particle swarm optimiser with neighbourhood operator. In *Proceedings of the IEEE congress on evolutionary computation (CEC)* (pp. 1958–1962). Piscataway: IEEE.

Trelea, I. C. (2003). The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, *85*(6), 317–325.

van den Bergh, F. (2002). *An analysis of particle swarm optimizers*. PhD thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa.

Vesterstrøm, J. S., Riget, J., & Krink, T. (2002). Division of labor in particle swarm optimization. In *Proceedings of the IEEE congress on evolutionary computation (CEC)* (pp. 1570–1575), Honolulu, HI. Piscataway: IEEE.

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, *393*, 440–442.

Wei, C., He, Z., Zhang, Y., & Pei, W. (2002). Swarm directions embedded in fast evolutionary programming. In *Proceedings of the IEEE congress on evolutionary computation (CEC)* (pp. 1278–1283), Honolulu, HI. Piscataway: IEEE.

Xie, X., Zhang, W., & Yang, Z. (2002). Dissipative particle swarm optimization. In *Proceedings of the IEEE congress on evolutionary computation (CEC)* (pp. 1456–1461), Honolulu, HI. Piscataway: IEEE.

Yasuda, K., Ide, A., & Iwasaki, N. (2003). Adaptive particle swarm optimization. In *Proceedings of the IEEE international conference on systems, man and cybernetics* (pp. 1554–1559). Piscataway: IEEE.

Zhang, W.-J., & Xie, X.-F. (2003). DEPSO: hybrid particle swarm with differential evolution operator. In *Proceedings of the IEEE International conference on systems, man and cybernetics (SMCC)* (pp. 3816–3821), Washington, DC. Piscataway: IEEE.

Zheng, Y.-L., Ma, L.-H., Zhang, L.-Y., & Qian, J.-X. (2003). On the convergence analysis and parameter selection in particle swarm optimization. In *Proceedings of the IEEE international conference on machine learning and cybernetics* (pp. 1802–1807). Piscataway: IEEE.