Nat Comput (2007) 6:467–484
DOI 10.1007/s11047-007-9049-5

A review of particle swarm optimization. Part I: background and development

Alec Banks · Jonathan Vincent · Chukwudi Anyakoha

Published online: 17 July 2007

Abstract Particle Swarm Optimization (PSO), in its present form, has been in existence for roughly a decade, with formative research in related domains (such as social modelling, computer graphics, simulation and animation of natural swarms or flocks) for some years before that; a relatively short time compared with some of the other natural computing paradigms such as artificial neural networks and evolutionary computation. However, in that short period, PSO has gained widespread appeal amongst researchers and has been shown to offer good performance in a variety of application domains, with potential for hybridisation and specialisation, and demonstration of some interesting emergent behaviour. This paper aims to offer a compendious and timely review of the field and the challenges and opportunities offered by this welcome addition to the optimization toolbox. Part I discusses the location of PSO within the broader domain of natural computing, considers the development of the algorithm, and refinements introduced to prevent swarm stagnation and tackle dynamic environments. Part II considers current research in hybridisation, combinatorial problems, multicriteria and constrained optimization, and a range of indicative application areas.

Keywords Particle swarm optimization · Natural computing

1 Introduction

Conventional computing paradigms often have difficulty dealing with real world problems, such as those characterised by noisy or incomplete data or multimodality, because of their inflexible construction. Natural systems have evolved over millennia to solve such

A. Banks (✉)
Tornado In-Service Software Maintenance Team, Royal Air Force, Boscombe Down, Wiltshire, UK
e-mail: test-flt@tismt.raf.mod.uk

J. Vincent · C. Anyakoha
Software Systems Modelling Group, School of Design, Engineering and Computing,
Bournemouth University, Poole, Dorset, UK

problems, and, when closely examined, these systems often contain many simple elements that, when working together, produce complex emergent behaviour. They have inspired several natural computing paradigms that can be used where conventional computing techniques perform unsatisfactorily.

This review considers Particle Swarm Optimization (PSO), a relatively recent addition to the field of natural computing, that has elements inspired by the social behaviour of natural swarms, and connections with evolutionary computation. PSO has found widespread application in complex optimization domains, and is currently a major research topic, offering an alternative to the more established evolutionary computation techniques that may be applied in many of the same domains.

Part I of the review is structured as follows. The broader context is first established in a brief examination of natural computing paradigms. The development and refinement of the particle swarm approach is then treated, with consideration of its historical background, as well as the state-of-the-art, as evidenced by recent literature. Issues related to dynamic environments, stagnation, and parameter tuning, are also discussed, followed by a brief review of selected works on parallel implementations.

2 Natural computing paradigms

In this section, the broader context of natural computing is discussed, within which PSO is located. Natural computing, as a field of study, considers the computational utility of architectures and algorithms inspired by nature, and in particular their application to problems that are resistant to traditional methods (see, for example, Calude et al. (2001)). Whilst this section considers several selected examples from the field, it is not intended to provide a comprehensive review; there are many other forms of natural computing that are not covered here, including *inter alia* DNA computing, molecular computing, and artificial immune systems.

Many of the natural computing paradigms share high level attributes such as the ability to deal with noisy or incomplete data, the ability to solve combinatorial problems, emergent properties where the complex behaviour of the whole belies the simplicity of the agents, rules or interactions involved, use of some form of competition and cooperation, the requirement for some form of learning, and are often computationally efficient with the possibility of distributed implementation. Turing's (1952) work on how nature produces complex patterns by low level rules inspired much of the research into natural computing and emergent behaviour, which Johnson (2001) describes as *"leaderless individuals using low level rules to achieve higher levels of sophistication"*. Computationally simple processing architectures, inspired by natural artefacts or processes, have been found to provide complex and useful forms of computation. As with PSO, these often arise from the collective behaviour of decentralised, self organised systems. The drawback of such paradigms is their inherent non-determinism, and whilst they offer approximate solutions to problems for which an analytic solution or traditional numerical method is infeasible or impossible, they present challenges in terms of design and validation.

2.1 Levels of development

Each of the fields of natural computing discussed in this section can be considered to be on a particular level of organism development. Sipper et al. (1998) discuss three levels of

development (Fig. 1): phylogeny, the development of a species over time (evolution); ontogeny, the adaptation (in its own lifetime) of a particular organism to its environment (specialisation); and epigenesis, the development of a complex organism in which the overall structure requires ongoing experiential learning to fully develop.

2.2 Models of human intelligence

Artificial Neural Networks (ANNs) are a model of the epigenetic programme of the human brain. McCulloch and Pitts (1943) first modelled the physical aspects, and Hebb (1949) introduced the first learning algorithm. The paradigm offers huge potential since, with approximately 100 billion basic processing elements (neurons), the human brain is a massively powerful parallel computing system. ANNs can provide a variety of behaviours depending on their structure and training; see Haykin (1999) for a thorough treatment. As with most natural computing paradigms, the neural approach offers a mechanism for problem solving that does not require an analytic approach, but one where input-output mappings are learnt through a supervised or unsupervised training regime. They function as flexible computational structures, having numerous advantages over conventional computation systems, including *inter alia* their ability to approximate solutions with noisy or missing data, graceful degradation, and generalisation to a solution for which the input has not been seen during training. They also share the primary drawbacks of natural computing paradigms: unpredictability and the difficulty of validation due to their 'black box' nature.

First developed by Feigenbaum et al. (1971) of Stanford University for the Dendral project (1965–1983), Knowledge Based Systems (KBSs) represent a different perspective on the (human) brain's epigenetic process (although it may be argued that this is not within the taxonomy of natural computing, its consideration is pertinent to this section of the review). Unlike ANNs, there is no attempt to model the physical structure; an inference engine fulfils that role. Learning is modelled by the construction of rules that are produced under the guidance of domain experts and held in a knowledge base. The abstract process of reasoning occurs when the inference engine fires rules as a result of data input. This

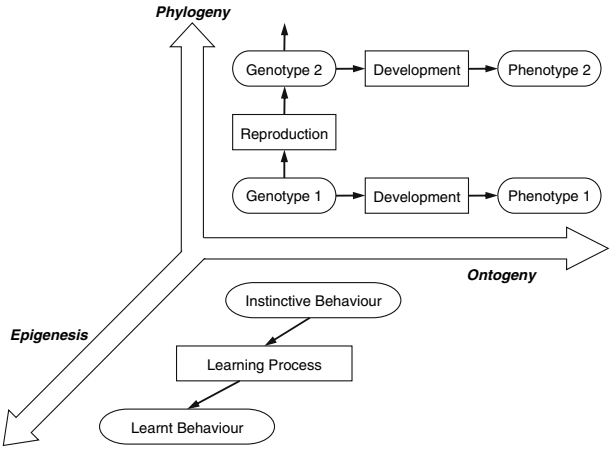

Fig. 1 Orthogonal levels of organism development (adapted from Sipper et al. (1998), with the addition of the epigenesis axis)

explicit knowledge representation and reasoning offers the advantage that knowledge can be updated dynamically due to the separation of knowledge from the inference engine, and, unlike ANNs, a KBS can provide a rationale behind its decisions. It is, however, very dependent on the experts' ability or willingness to impart their domain knowledge, and suffers from poor generalisation and adaptiveness.

2.3 Competitive populations

Not all natural computing paradigms seek to model human processes. Friedberg's automatic programming machine (1958) introduced Evolutionary Algorithms (EAs), which are a stochastic, population-based phylogenic solution that exploits useful inheritance and competition between individuals on a 'survival of the fittest' basis. The field of EAs can be broadly divided into a number of sub-fields: Genetic Algorithms (GAs) begun by Bremermann (1958) but generally given a broader appeal by Holland (1962, 1975) and Goldberg (1989), Evolution Strategies (ESs) by Rechenberg (1965), Evolutionary Programming (EP) by Fogel et al. (1966) and, more recently, Genetic Programming (GP), initially developed by Koza (1992) and Differential Evolution (DE) by Storn and Price (1995). Evolutionary approaches offer a number of advantages (largely shared by PSO) including *inter alia* general applicability across a wide range of problems, no requirement for prior knowledge of the problem space, and ease of development. As with all heuristic search/optimization approaches (including those based on ant colonies or particle swarm), the "No Free Lunch" theorems apply (Wolpert and Macready 1997), essentially recognising that general applicability comes at the cost of domain specific performance, and that there is no one best approach for all domains. Further, there are no guarantees of finding a globally optimal solution, even within a specified tolerance. The iterated evaluation of the objective function may be computationally expensive, hence the interest in computational distribution, for which evolutionary algorithms are particularly well suited, see, for example, Cantú-Paz (2000).

2.4 Cooperative populations

Whilst EAs are primarily competitive, Ant Colony Systems (ACS) and PSO adopt a more cooperative strategy. They could be said to be ontogenetic since the population may be thought of as a multicellular organism adapting to optimize its performance in its environment. ACS (Colorni et al. 1991; Dorigo and Stützle 2004) generally use the metaphor of ants searching for food to represent the search/optimization process, incorporating the division of labour and the use of pheromone trails to assist with the autonomous organisation of the group. This is advantageous because poor solutions are avoided and difficult tasks are divided between many individuals; premature convergence to sub-optimal solutions can be avoided through the individuals' ability to move away (enabled by a stochastic element in the algorithm) from current best solutions to discover new, potentially better, solutions. The primary disadvantage with this approach, also shared by PSO, is that it is difficult to develop, since the goal is the emergent behaviour of the group rather than the individual agent characteristics. This approach works well in combinatorial problems, such as network routing (Di Caro and Dorigo 1998).

PSO (Kennedy and Eberhart 1995, Kennedy et al. 2001) shares some of the characteristics of ACS, except they do not disseminate information via pheromone trails. Rather,

the individuals communicate their best solution, and the members of the group follow a combination of the group's previous best and their own previous best, with an additional stochastic element to assist exploration. PSO also shares several properties with GAs (their interacting populations, for example), and Angeline (1998a) compares the two techniques. Some of Angeline's observations have since been rendered obsolete by more recent developments of PSO, and hold only for the early formulations. For example Eberhart's inertia weight allows the effect of velocity memory to be dynamically reduced thereby allowing targeted exploitation. The research does indicate, however, that PSO finds solutions near optima faster than the GA, although the subsequent exploitation is weaker. Whilst PSO is more efficient, it is also more prone to premature convergence than EAs (Angeline 1998a) because of the possibility of an overwhelming, sub-optimal group best influence early in the search.

3 PSO: from animation to optimization

The notion of employing many autonomous agents (particles) that act together in simple ways to produce seemingly complex emergent behaviour was initially considered to solve the problem of rendering images in computer animations that attempt to resemble natural phenomena. Reeves (1983), one of the early pioneers, as part of his work at Lucasfilm, implemented a particle system that used many individuals working together to form what appeared to be a fuzzy object (such as a cloud or an explosion). A particle system stochastically generates a series of moving points, these typically being initialised to predefined locations. Each particle is assigned an initial velocity vector. In graphical simulations, these points may also have additional characteristics such as, for example, colour, texture and limited lifetime. Iteratively, velocity vectors are adjusted by some random factor. Each particle is then moved by taking its velocity vector from its current position and adjusting where necessary by a constrained angle to make the movement appear realistic. Such systems are widely deployed in the generation of special effects and realistic interactive graphical environments.

For some animations it is necessary to render group behaviour with higher level dynamics than simple particles (such as a flock of birds). Scripting of individual behaviour is possible, but tedious, and it is difficult to generate responses that look natural. Reynolds (1987) used Reeves' particle system as the basis for his higher order (in terms of the objects being modelled) flocking algorithm. He took the particle movement and added orientation and inter-object communication. These additional behaviours allowed individual Boids (*Bird-oid* objects) to follow some simple flocking rules: Boids should avoid colliding with fellow Boids; they should attempt to match each others velocity vector; and the should try to stay close to each other. Development of this underlying model, increasing the intelligence of the individuals, removed the requirement for individual trajectories to be established. The increased level of autonomy raised several further questions such as conflict resolution. Reynolds used a deterministic approach, prioritised ordering, to resolve such issues but points out that decisions could be made non-deterministically.

Inter-Boid perception is an issue that not only affects animation realism but also its efficiency. A naïve implementation might adopt a system where each Boid is aware of what all other flock members were doing; rendering the algorithm unfeasibly complex with larger flocks. To avoid this, Reynolds recommended implementing a neighbourhood system, which is how flocking appears to work in nature due to the limited vision of flock

members, although later research indicated that this changes the emergent behaviour of the flock (Kennedy and Mendes 2002).

Kennedy and Eberhart (1995) sought to extend Reynolds' model to reflect social behaviours. More importantly, they replaced the simple roost goal developed by Heppner and Grenander (1990), who had developed an alternative flocking algorithm, with a more realistic goal: that of searching for food. It was this goal that led the researchers to use non-trivial mathematical problems as a fitness function for flock members (now called agents (see Minsky 1986) since they had become more generalised than bird models). Once the goal of locating the target was achieved the variables that were found to be redundant were removed, providing a simple and more efficient model. The resultant algorithm for calculating the next particle position (x) was:

$$v_{t+1} = v_i + \varphi_1\beta_1(p_i - x_i) + \varphi_2\beta_2(p_g - x_i) \quad (1)$$

$$x_{t+1} = x_t + v_{t+1} \quad (2)$$

where constants φ_1 and φ_2 determine the balance between the influence of the individual's knowledge (φ_1) and that of the group (φ_2) (both set initially to 2), β_1 and β_2 are uniformly distributed random numbers defined by some upper limit, $\beta_{\max}$, that is a parameter of the algorithm, p_i and p_g are the individual's previous best position and the group's previous best position, and x_i is the current position in the dimension considered. Note the sign in the brackets results in an acceleration of the particles towards the previously known best points in the space. This balance between accelerations toward local or global best, *i.e.* between exploration and exploitation, could be considered as broadly equivalent to selection pressure in a GA, although control is more complex in a GA, with other critical parameters such as population size and mutation rate. For an n-dimensional space the particle velocity is calculated for each dimension, $i = 1,2...\ n$, then resolved into a final vector for updating the particle's position. The technique was applied to a number of applications and benchmark problems, which indicated robust performance. Thus, the conceptual step was made, rather serendipitously, from animation of natural phenomena and social groupings to optimization of complex functions to which it has since been extensively applied. As noted by Monson and Seppi (2005a), although subject to refinement, the basic formulation of the PSO equations has remained remarkably unchanged.

4 Algorithm refinement

At a fundamental level, one might argue that the key difference between PSO and evolving populations is the way in which new samples are generated. In GAs, new samples are produced by some recombination of selected 'parent' solutions, which may then go on to replace members of the population. In PSO, new samples are generated by perturbation of existing solutions. The latter approach can lead to issues of stability. To reduce the possibility of particles flying out of the problem space, Eberhart et al. (1996) put forward a clamping scheme that limited the speed of each particle to a range $[-v_{\max}, v_{\max}]$ with $v_{\max}$ usually being somewhere between 0.1 and 1.0 times the maximum position of the particle.

Whilst experimenting with the standard algorithm, Shi and Eberhart (1998) noted that without the velocity memory—the first part of Eq. (1)—the swarm would simply contract to the global best solution found within the initial swarm boundary (providing a local search). Conversely, with the velocity memory, the swarm will behave in the opposite

sense, expanding to provide a global search. To assist with the balance between exploration and exploitation a modified PSO, incorporating an inertia weight, ω, was introduced thus:

$$v_{t+1} = \omega v_t + \varphi_1 \beta_1 (p_i - x_i) + \varphi_2 \beta_2 (p_g - x_i) \tag{3}$$

At the time, experiments were only conducted on a single test function, but the weight has been applied many times since, and is generally considered useful (Carlisle and Dozier 2000; Trelea 2003). The initial experiments suggested that a value between 0.8 and 1.2 provided good results, although in later work (Eberhart and Shi 2000) they indicate that the value is typically set to 0.9 (reducing the stepwise movement of each particle, allowing greater initial exploration) reducing linearly to 0.4 (speeding convergence to the global optimum) during an optimization run.

Constriction is an alternative method for controlling the behaviour of particles in the swarm. Rather than applying inertia to the velocity memory, Clerc and Kennedy (developed 1999, published 2002) applied a constriction factor, χ, to the new velocity (see also Carlisle and Dozier (2001)):

$$v_{t+1} = \chi \{ v_t + \varphi_1 \beta_1 (p_i - x_i) + \varphi_2 \beta_2 (p_g - x_i) \} \tag{4}$$

$$\chi = \frac{2}{\left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|} \quad \text{where} \quad \varphi = \varphi_1 + \varphi_2, \varphi > 4 \tag{5}$$

With this formulation, the velocity limit, $v_{\max}$, is no longer necessary. A choice need not be made between constriction and inertia; Eberhart and Shi (2000) showed that with judicious parameter settings (ω set to χ and $\varphi_1 + \varphi_2 > 4$) the two approaches were algebraically equivalent and improved performance could be achieved across a wide range of problems.

Kennedy (2000) extended the social metaphor by experimenting with a model of social stereotyping, using cluster analysis of the particles best performances. Several versions of the algorithm were produced, each replacing the previous best aspects of the constricted algorithm. The results were predictable; replacing the individual best with the local cluster centre showed the most promising results, since the cluster centres are likely to be more fit. Cluster analysis does come at a high computational cost and Kennedy only performed simplistic analyses in an attempt to minimize the effect. Even so, the technique still generally took longer in real time.

Since the PSO algorithm was developed from the original flocking system, various components have been added (such as inertia, constriction) and removed (such as velocity matching, collision avoidance). Kennedy (2003) revisited the constricted PSO and examined whether the added components were necessary, and whether any further components could be removed. Various experiments were performed with a view to paring the process for further efficiency gains. To achieve this, a Gaussian PSO was developed. In this implementation the entire velocity vector is replaced by a random number generated around the mean $(p_{id} + p_{gd})/2$ with a Gaussian distribution of $|p_{id} - p_{gd}|$ in each dimension (d). This effectively means that the particles no longer 'fly' but are 'teleported'. Kennedy justified this departure on the grounds that it is the social aspect of the swarm that is more important to its effectiveness. This was empirically backed up with the Gaussian influenced swarms performing competitively.

Kennedy was not alone in believing that the social metaphor is the prime mover and that the mechanism for particle motion may be radically modified. Monson and Seppi (2004)

showed that particle motion could be further improved through changing the mechanism to a system influenced by Kalman filtering in which the motion is entirely described by predictions produced by the filter. Once again, the justification for such a radical change to particle movement is the maintenance of the social aspect of PSO, which is achieved through the Kalman filter's sensor model. The approach, whilst providing very accurate optimization, is computationally expensive compared with the canonical PSO. Further research has been conducted (Monson and Seppi 2005b) and the filter modified to match its application. This had the effect of returning optimization speed to canonical PSO levels whilst maintaining the accuracy benefits achieved by filtering.

Using different social networks has been a popular area of research in PSO since its inception (see Kennedy (1999) for an analysis). Parsopoulos and Vrahatis (2004) modified the constricted algorithm to harness the explorative behaviour of global search and exploitative nature of a local neighbourhood scheme. To combine the two, two velocity updates are initially calculated:

$$G_{t+1} = \chi\{v_t + \varphi_1\beta_1(p_i - x_i) + \varphi_2\beta_2(p_g - x_i)\} \tag{6}$$

$$L_{t+1} = \chi\{v_t + \varphi_1\beta_1'(p_i - x_i) + \varphi_2\beta_2'(p_{gl} - x_i)\} \tag{7}$$

where G and L are the global and local velocity updates respectively, p_g is the global best particle position and p_{gl} is the particle's local neighbourhood best particle position. These two updates are then combined to form a unified velocity update (U), which is then applied to the current position:

$$U_{t+1} = (1-u)L_{t+1} + uG_{t+1} \quad u \in [0,1] \tag{8}$$

$$x_{t+1} = x_t + U_{t+1} \tag{9}$$

where u is a unification factor that balances the global and local aspects of the search and suggestions are given to add mutation style influences to each in turn. Experimentation was promising and further work has been carried out since for dynamic environments (Parsopoulos and Vrahatis 2005a, 2005b).

Recently, a more realistic model of particle perception was introduced to PSO (Kaewkamnerdpong and Bentley 2005), with the aim of making it a viable option for applications that might otherwise suffer from imperfect communication, such as robotics. In the natural world, a social animal will often suffer imperfect perception of the other animals in its sociometric neighbourhood and must rely on information it receives via limited perceptive acuity. To model this, the particles were not allowed to communicate directly but only to observe particles within their pre-set perceptive range (replicating the stigmergic behaviour of some species, such as dancing honeybees); this is also realistic in that the information is regarded as more reliable where its source is closer. Each particle is also allowed to observe the local area (*i.e.* sample the local solution space). If there is a better solution nearby it uses that as its individual best position. The position update algorithm for this method is dependent on the current state of the particle and can be summarised as follows:

- If there is an observed position $p_{observed} \geq p_i$ then set p_i to $p_{observed}$.
- If no neighbouring particles are *perceived* then do not use a social component; *i.e.* use Kennedy's (1997) cognition only model.
- If multiple particles are *perceived* use their average current position as the group best (p_g).
- If a single particle is *perceived* then use its position as the group best (p_g).

Experimentation showed the perceptive swarm to be competitive in terms of accuracy but had higher computational overheads. This was justified on the grounds of the potential fields of application.

5 Discrete PSO

Although the main body of PSO development work has concentrated on optimization in continuous search spaces, research has also been conducted into the application of the algorithm to discrete problems. Kennedy and Eberhart (1997) developed the first discrete version of PSO for binary problems. They solved the problem of moving the particle through the problem space by changing the velocity in each vector to the probability of each bit being in one state or the other. For example, in any given dimension, d, if v_{id} represents the probability of the value $x_{id,}$ being a 1 and $v_{id} = 0.4$ there is a 40% chance that x_{id} will be 1. Note that the calculation of the velocity still used equation (1), so to constrain the probability a sigmoid transformation was applied. Later, Al-kazemi and Mohan (2002) compared this with an alternative velocity update technique called Multi-phase Discrete PSO (M-DiPSO). This employed 3 coefficients (c_v, c_x and c_g) whose values were set to either 1 or -1 depending on the phase of the optimization. The M-DiPSO employed a hill climbing strategy where only fitter solutions were accepted, and, therefore, the individual best was replaced by the previous position in the velocity update equation, which for each component of the particle was now given by:

$$v_{(t+1)} = c_v \cdot v_{(t)} + c_x \cdot x_{(t)} + c_g \cdot g_{(t)} \tag{10}$$

where x and g are the particle's previous position and the swarm best position respectively. This new approach was found to be competitive across several problems, requiring fewer objective function evaluations to converge, and was able to operate using small swarm sizes.

Using these techniques a wide range of bounded problems may be solved through the encoding of potential solutions into binary patterns. Where the solution space is not specifically bounded, other techniques can be applied. Laskari et al. (2002) presented an implementation of PSO that initially operated in continuous space but truncated the real number values to integers. In experimentation that compared both a simple truncation and a progressive truncation with a conventional branch and bound technique, the approach was found to be effective across a range of integer programming problems.

More recently, Afshinmanesh et al. (2005) have developed a hybrid approach, combining PSO and Artificial Immune Systems (AIS), for the optimization of binary problems. The process begins with the production of a potential velocity update vector for each particle through the combination, using various Boolean operators, of the various PSO influences described earlier. Then the AIS aspect is applied; its purpose is to ensure the hamming distance of the update vector is not too distant from the particle's present position. If it is too distant, the vector is limited by randomly clearing excessive '1's. The update vector is then applied to the original particle position. Limited experimentation presented in the work indicates that the technique could be effective and very efficient.

6 Swarm analyses

Kennedy (1997) conducted one of the first and, possibly, most useful analyses of the PSO algorithm. Its purpose was to establish the benefit of social interaction in attaining the

swarm's goal. This was achieved by comparing the standard algorithm with three new models, all in a basic neighbourhood configuration. The new models were: cognition-only, where there is no influence from other individuals; social-only, where the best of all neighbourhood members is used; and the selfless model, where the individual is excluded from the neighbourhood best competition. The two new social models performed best, but the author did not suggest replacing the standard algorithm, probably because the objective function was overly simple and in more complex situations, without the individual influence, the swarm could easily converge to an early sub-optimal solution.

A more theoretical approach was taken by Ozcan and Mohan (1998 and 1999), in which they observe that rather than 'flying' through the problem space, particles actually 'surf' on sine waves. More importantly they give an insight into the role of φ, which affects the amplitude and frequency of the sine wave being followed, and show that an adaptive inertia value could help a particle stuck in a poor region move away by hopping onto a more productive wave. This work made several simplifications, the most significant being the removal of the stochastic element of the algorithm. Clerc and Kennedy (2002) in a thorough treatment of the underlying dynamics, reintroduced this, and showed that far from enjoying perfect surf, unrestricted stochastic particles engaged in a 'drunken walk' toward infinity (explosion). The work does not refute the surfing behaviour though; it supported it for a restricted particle when time was treated discretely, however, when time was treated continuously the particle would spiral toward its goal. From this analysis the constriction factor, described earlier, was developed, preventing swarm explosion through the control of the stochastic elements β_1 and β_2.

It is not just particle trajectory that is important to swarm behaviour; the method by which p_g is calculated affects how particles are allowed to explore areas away from the current swarm best. Through investigations into the impact of a variety of social networks, Kennedy (1999) focussed on four basic topologies: circles, wheels, stars and random edges; sociometric network shortcuts for each were also tested. The research concluded that the topology does affect the swarm's performance but that the optimal configuration is objective function dependent; the shortcuts were found to have very unpredictable effects. An inferred finding from the work was that networks that slow down communication could help prevent premature convergence in multimodal landscapes. Kennedy and Mendes (2002) supported this after extensive experimentation and concluded that the best, on average, configuration is a von Neumann topology. This makes sense since it has connectivity somewhere between that of a circle (using the local neighbourhood best, which is slow and better in multimodal landscapes) and totally connected (using whole group best), which is fast and suited to simple landscapes). See also, the further work in Mendes et al. (2003).

Particle swarms, in common with many population based algorithms, have an unintentional bias to perform better in problems that have optimal solutions toward the certain areas of the swarm's initialised position, for example the centre of the solution space or the absolute origin. Monson and Seppi (2005b) analysed this effect and showed that Angeline's (1998b) use of Gelhaar and Fogel's (1996) "Regional Scaling" method was insufficient to fully detect the problem. They proposed an alternative technique called "Centre Offset". To illustrate this they used Clerc's TRIBES (2003), a PSO tool that does not require sociometric or swarm size parameter tuning (thus allowing for more consistent analyses of particle motion), to apply PSO to several common problems. Whilst the regional scaling was successful in detecting bias in functions with optima at the centre of the initialised swarm, the centre offset was successful in highlighting bias toward the absolute origin. The authors conclude that neither technique is sufficient on its own and, where such analysis is required, both should be used.

7 Swarm stagnation

Rapid swarm convergence is one of the main advantages of PSO, but this can also be problematic since, if an early solution is sub-optimal, the swarm can easily stagnate around it without any pressure to continue exploration. To avoid this, van den Bergh and Engelbrecht (2002) introduced the Guaranteed Convergence PSO (GCPSO), which uses a different velocity update equation for the p_g particle. This equation causes the particle to perform a random search around p_g within a radius defined by a scaling factor that adapts to the success of the particle in not stalling at p_g. In unimodal problems the GCPSO easily outperforms the modified PSO but produces similar results when used in multimodal environments. The main drawback is that a priori knowledge may be required to optimize the adaptation of the scaling factor. Its real strength is that its ability to operate with small swarm sizes makes it an enabling technique for parallel niching solutions.

A collision avoidance system was introduced by Krink et al. (2002), in which particles attempting to cluster about a potential solution bounce away from the particle already in the potentially sub-optimal position. Three bouncing strategies were tested: a random direction changer, a realistic bounce and a random velocity changer. The first was found to provide little benefit, probably because it more or less became a random search. Conversely, the latter two provided significant improvement, especially in highly multimodal landscapes, because the particles were never allowed to cluster and stagnate.

Low levels of particle diversity can be a good indicator of swarm stagnation. Riget and Vesterstrøm (2002) implemented a measure of diversity to control the alternating attraction and repulsion of the particles to and from the swarm best position. Initially the particles attracted each other until a minimum diversity level was met, the swarm then entered repulsion mode where the particles repelled each other until a maximum diversity level was reached. The attraction phase was achieved through the normal application of the canonical PSO velocity update formula; repulsion inverted the velocity update. The approach was compared with the canonical PSO and GA, and it performed better in all multimodal problems; indeed, when the dimensionality of the problems was increased, the magnitude of increased performance grew.

In the natural world, animals are often dispersed from a good location, such as a water hole, by predatory activity. Silva et al. (2002) introduced a predatory particle that was attracted to the swarm best particle but repulsed the rest of the swarm, forcing them to disperse and explore other areas of the search space. This approach was tested using four common high-dimensional problems and improved on the performance of PSO with a variety of parameter settings.

Xie et al. (2002) turned to thermodynamics to resolve the issue. The enhanced system, termed Dissipative PSO (DPSO), introduced negative entropy into the swarm to maintain a far-from-equilibrium state. These fluctuations are justified, with reference to Kennedy's social model, as an implementation of human creativity, introducing new ideas that, when seen to provide system improvement, are adopted by the other swarm members. Experimentation indicated that the DPSO performed similarly as standard PSO in the early stages and improved performance at higher iterations (when the standard system had stagnated).

Slowing down particle movement toward the early group best positions can also reduce the risk of stagnation. Veeramachaneni et al. (2003) achieved this with their Fitness-Distance-Ratio-PSO (FDR-PSO), which slows convergence through adding the influence of the fittest near neighbour to the existing individual and swarm best influences. Several algorithm variants were tested against the modified PSO using six benchmark functions

with the best variant easily outperforming the modified PSO in terms of optimization quality. This inevitably comes at a price, and convergence is shown to take approximately twice as many iterations, and hence an expensive approach for less complex objective functions where the modified PSO could achieve acceptable results.

Two further extensions were proposed by Ratnaweera et al. (2004) that used time-varying acceleration coefficients (TVAC). The first of these being an EA mutation based hybrid (MPSO-TVAC). In this approach, if the group best solution does not improve over a given number of iterations, a random particle is moved (mutated) to another random location in the search space. This was found to be competitive in comparisons with other PSO variants for multimodal problems but suffered in unimodal landscapes due to the movements away from the global optimum in situations where the swarm should have located the global optimum quickly. The second, a self-organising hierarchy (HPSO-TVAC), is a parameter automation strategy that varies the values of φ_1 and φ_2 over the duration of the optimization run, with the aim of reducing group influence in the early stages, thus preventing premature convergence. Experimentation indicated that, for optimal results the cognitive component, φ_1, should reduce from 2.5 to 0.5 and the social component, φ_2, should increase from 0.5 to 2.5. The TVAC modification alone was found to be insufficient in preventing premature convergence in multimodal landscapes. HPSO-TVAC, inspired by the investigation into the removal of particle momentum during the original work of Kennedy and Eberhart (1995), modified the PSO still further by removing the velocity memory of each particle. In the canonical PSO, if the velocity memory is removed the swarm careers toward early local optima, at which point they stagnate. This problem is resolved in this extension by reinitialising the velocity vector of a swarm particle each time they stagnate. The swarm is then allowed to operate until a convergence criterion is met (such as solution quality). During experimentation it was found that, provided the re-initialisation of the velocity was time-variant, the method provided good performance across a wide variety of unimodal and multimodal problems, including the real-life application of tuning an internal combustion engine.

8 Configuration

Whilst it can be seen that the basic formulation of the particle system is straightforward, configuration—parameter tuning—for optimal performance can be non-trivial. Carlisle and Dozier (2001b) provide some guidance on achieving a working, if not optimal, solution supported by evidence from six experiments. The advice is to start with a constricted algorithm with a global neighbourhood of 30 particles that are updated asynchronously, setting $\varphi_1 = 2.8$ and $\varphi_2 = 1.3$ and, where the objective function enforces $x_{\max}$, setting the velocity of particles at $x_{\max}$ to zero. There is, however, no guidance on how to improve an optimizer from this initial configuration.

Parsopoulos and Vrahatis (2002) showed that further assistance can be provided to improve the probability of a PSO algorithm achieving a valid solution, especially where rapid convergence is sought. They seeded the initial swarm with particles whose locations had been calculated using the Nonlinear Simplex Method (NSM) as developed by Nelder and Mead (1965). Results showed that, whilst there was the overhead of the initial NSM processing, the benefits of initialising some particles to good positions could be significant. Trelea (2003) provides less prescriptive, but more qualitative, advice for the running of parameter tuning experiments and what to try when performance is poor; this is also supported by empirical evidence.

More recently, Zhang et al. (2005) built on Carlisle and Dozier's work by conducting extensive experimentation across a wide range of test functions to provide further guidance on the optimal parameter setting of the constricted PSO (Clerc and Kennedy, 2002). They found that different test problems required different settings; in particular, unimodal and multimodal problems required different settings of φ and the reduction rate of $v_{\max}$, and higher dimensional problems require larger swarm sizes.

9 PSO in dynamic environments

Dynamic environments are defined by variations in objective function evaluation over time, and are problematic when the optimum position (the goal) moves throughout the problem space in steps larger than the group best particle can track naturally. Canonical PSO is prevented from tracking the goal, since, unless the goal happens to move to a space also occupied by another swarm member, the swarm will continue to move toward the swarm's previous best, set at the goal's previous position. There are several options to address this, such as re-initialising the swarm or using the old swarm but resetting the previous best values.

Carlisle and Dozier (2000) proposed to solve this in a unimodal environment by occasionally resetting each particle's p_i and p_g to their current position causing them to 'forget' their previous experience and thus reduce the influence of previous good positions. Two triggering mechanisms were considered: goal velocity (PSO can track small changes without amending the algorithm) and time. Testing was performed using Kennedy's (1997) social models, where fitness was judged to be the geometric distance between particle and goal; goal velocity was altered for each trial. The experiments performed were limited in that the goal maintained a constant velocity and heading but did show that, by forgetting its past, a particle stands more chance of tracking a moving goal, although a priori information regarding the update rate of the environment is advantageous for setting an optimal memory reset rate. Further work was conducted on the method (Carlisle and Dozier, 2001a and 2002) in which 'Sentries' were added to the system to monitor the environment and initiate memory resets at appropriate times. Whilst this improved the performance, it arguably moves away from the agent-based nature of the paradigm. Perhaps more significantly, the authors note that the inertia mechanism can be counter productive in dynamic systems because it slows the particles down, making them less able to track moving optima.

Eberhart and Shi (2001) used a standard PSO that had a randomly varying inertia value with a mean of 0.75 and range of 0.5–1.0. They applied it successfully to a 10 dimensional parabolic function, but the swarm was allowed 100 iterations between dynamic updates and the swarm performance degraded dramatically as the movement severity increased. The authors propose the re-introduction of a momentum parameter, removed during initial development of the paradigm, but they were unsure of its value.

In a move away from the biological metaphor, Blackwell and Bentley (2002) utilised an analogy of electrostatic energy to track a dynamic global optimum. In this work they endowed the particles with identical charges that produce a repulsive force between them (canonical PSO being considered as having neutral particles), replicating Coulomb's electrostatic inverse square law. It was found that a combination of neutral and positively charged particles produced the best results because the neutral particles allowed exploitation whilst the enforced separation of the charged particles maintained exploration. This approach worked satisfactorily on the test functions because the goal movement was stepwise and the spread of particles ensured fast convergence at each new optimum.

Parrott and Li (2004) used a GA inspired speciation approach to track dynamic multimodal peaks. Sub-species (groups of particles that share the same goal) are created by artificially seeding the swarm with sub-species leaders that act as p_g for particles within a given radius. The sub-species are then reformed post iteration of the whole swarm with any new p_g particles replacing their previous sub-species leaders. To prevent crowding and preserve diversity, sub-species size is limited; those particles insufficiently fit to be in a sub-species are reset to random positions. The technique is less efficient than canonical PSO algorithms due to its requirement to re-evaluate the fitness of every particle's p_i position as well as that of their current position. However, this may be justified, since it prevents the particle being drawn toward what could have become a worse location in the search space. A further limitation is that the goal must be travelling slower than the particle's maximum velocity, or it can escape from the swarm.

It can be seen from these earlier approaches that particle memory is generally regarded as the problem area in dynamic environments. To make particle memory more transient, Cui et al. (2005) modified the technique for assessing best particle fitness from (10) to (11):

$$p_{t+1} = \begin{cases} p_t & f(x_{t+1}) \leq f(x_t) \\ x_{t+1} & f(x_{t+1}) > f(x_t) \end{cases} \tag{11}$$

$$p_{t+1} = \begin{cases} p_t \times T & f(x_{t+1}) \leq p_t \times T \\ x_{t+1} & f(x_{t+1}) > p_t \times T \end{cases} \tag{12}$$

where T is a known as evaporation, having a value between 0 and 1. As its name suggests, this decreases over time. The approach showed some success in tracking moving optima but only where the target was moving slower than particle maximum velocity. Even so, this was shown to be an improvement over canonical PSO and the authors suggest further work that indicates that it may be possible to improve the scheme further.

10 Parallel implementation

Despite the agent-based nature of PSO naturally lending itself to parallel implementation, the literature indicates that a relatively small amount of research has been conducted in this area. The expected gains in parallel implementation are speed and robustness. In speed terms, theoretically, a parallel swarm should be able to operate at n times the speed of a serial equivalent, where n is the number of processing nodes. However, communication overheads will reduce this speedup. Gies and Rahmat-Samii (2003) indicate that in their 10 node system the performance gain was 8 times that of the serial implementation. It should be noted that this figure would be expected to vary with fitness function and communication requirements. More processor intensive fitness functions and smaller communication requirements, for example, would make parallel implementations more efficient. Schutte et al. (2004) support these findings in their parallel solution to a large-scale engineering optimization problem. Their approach differed from Gies and Rahmat-Samii's in that, rather than applying the same fitness function to all particles in parallel, it was decomposed and distributed across several computational nodes with a single coordinating node performing the PSO algorithm. This forced the implementation to become synchronous and created the overhead that the swarm could only operate as fast as the slowest element of the fitness function. Further, it rendered the system less robust since any node failure effectively halted the optimization process.

An asynchronous implementation by Venter and Sobieszczanski-Sobieski (2005) overcame the problems created by the synchronous nature of Schutte et al.'s work through the implementation where possible of data pools. The only conventional operator that required updating at each complete swarm iteration was the inertia (the craziness operator (Kennedy and Eberhart 1995) would also require identical treatment where used). When tested using a non-trivial design problem the asynchronous system was found to produce results that matched the numerical accuracy of the synchronous system but was far more efficient, the improvement increasing with the number of processors.

Chang et al. (2005) presented a parallel PSO in which each particle evaluated the fitness function independently, and demonstrated that the efficiency of the algorithm is dependent on the strategy for communicating p_g, and that each strategy they investigated had its own merits that could be exploited where the correlation of the solution parameters was known a priori. Selecting an incorrect strategy was found to be very sub-optimal, so where the correlation was unknown, a hybrid strategy was found to be effective.

11 Conclusions

Part I of this review has considered the background and history of Particle Swarm Optimization and its place within the broader Natural Computing paradigm. The review then moved on to consider various refinements to the original formulation of PSO in both continuous and discrete problem spaces, analyses of swarm behaviour, and measures included to address stagnation. Finally, the review considered research relating to algorithm configuration, adaptations for dynamic environments and parallel implementation.

PSO offers rapid optimization of complex multidimensional search spaces, and is a popular contemporary algorithm for a wide range of search and optimization problems. However, this review has identified two significant areas of challenge for future development: swarm stagnation and dynamic environments. From the earliest presentation of the algorithm it has been acknowledged that the technique's major weakness is its propensity to converge prematurely on early, possibly suboptimal, group best solutions. The challenge to potential users is to avoid this by ensuring that particles keep moving. Several techniques for this have been proposed, but, akin to other optimization paradigms operating in multimodal landscapes, the problem lies in knowing when an optimal solution has been found. To ensure maximal exploration, several mechanisms were proposed to restrict the velocity of the particles; slower particles complete more fitness function evaluations while moving toward the final solution, increasing the likelihood that the global optimum will be found. Hybrids with other natural computing paradigms have also been utilised in further attempts to resolve PSO's inherent challenges. The main aims of these are to relocate stagnant particles (akin to mutation in genetic algorithms) or to slow them down (for example, through the use of subpopulations).

Considering the challenge of deploying PSO in dynamic environments, the major hindrance to successful optimization of such problems is the possibility of the optimum position having higher than particle velocity, and thus escaping the swarm. In dynamic problems, a balance has to be sought between velocity restriction and the need to keep track of optima. Slow moving optima can be tracked naturally because the particles continue to move about in an area dictated by the velocity of the particles. Problems arise when the movements of optima are larger than can be achieved by the particles. The underlying principle in all the reviewed dynamic systems has been to maintain a diverse swarm, thus increasing the likelihood that the optimum will move into an area that has coverage.

Swarm stagnation and dynamic environments remain current research topics. Part II of this review considers other research trends, covering recent work in hybridisation, multi-criteria and constrained optimization, as well as reviewing briefly a range of indicative application areas for PSO based systems.

References

Afshinmanesh F, Marandi A, Rahimi-Kian A (2005) A novel binary particle swarm optimization method using artificial immune system. EuroCon 2005 – The international conference on computer as a tool, Serbia & Montenegro, Belgrade

Al-kazemi B, Mohan CK (2002) Multi-phase Discrete Particle Swarm Optimization. Proceedings of fourth international workshop on frontiers in evolutionary algorithms (FEA 2002)

Angeline PJ (1998a) Evolutionary optimization versus particle swarm optimization: philosophy and performance differences. 7th Annual conf evolutionary programming

Angeline PJ (1998b) Using Selection to Improve Particle Swarm Optimization. Proceedings of IEEE congress on evolutionary computation, Anchorage, Alaska

Blackwell TM, Bentley PJ (2002) Dynamic search with charged swarms. Proceedings of the genetic and evolutionary computation conference 2002 (GECCO 2002), New York, NY, USA, pp 19–26

Bremermann HJ (1958) The evolution of intelligence. The Nervous System as a Model of its Environment. Technical report, no 1, contract no 477(17), Dept Mathematics, Univ Washington, Seattle, July, 1958

Calude CS, Paun G, Tataram M (2001) A Glimpse into natural computing. CDMTCS Tech Rep 117, Univ of Auckland, 2000 and J Multi-Valuate Logic 7:1–28

Cantú-Paz E (2000) Efficient and accurate parallel genetic algorithms. Kluwer Academic Publishers

Carlisle A, Dozier G (2000) Adapting particle swarm optimization to dynamic environments. Proceedings of international conference on artificial intelligence, vol 1, pp 429–434, Las Vegas, NV

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

Carlisle A, Dozier G (2001b) An Off-The-Shelf PSO. Proceedings of workshop on particle swarm optimization. Indianapolis, IN

Carlisle A, Dozier G (2002) Tracking changing extrema with adaptive particle swarm optimizer. Proceeding of WAC 2002, Orlando, Florida

Chang J-F, Chu S-C, Roddick JF, Pan JS (2005) A parallel particle swarm optimization algorithm with communication strategies. J Information Sci Eng 21:809–818

Clerc M, Kennedy J (2002) The particle swarm: explosion, stability and convergence in a multi-dimensional complex space. IEEE Trans Evol Comput 6:58–73

Clerc M (2003) TRIBES – Un Exemple D'Optimisation par Essaim Particulaire Sans Parametres de Contrôle. In: Optimisation par Essaim Particulaire (OEP 2003), Paris, France

Colorni A, Dorigo M, Maniezzo V (1991) Distributed Optimization by Ant Colonies. Proceedings of European conference on artificial life, Paris, France, pp 134–142

Cui X, Hardin CT, Ragade RK, Potok TE, Elmagraghby AS (2005) Tracking non-stationary optimal solution by particle swarm optimizer. Proceedings of the sixth international conference on software engineering, artificial intelligence, networking and parallel/distributed computing and first acis international workshop on self-assembling wireless networks (SNPD/SAWN'05)

Di Caro G, Dorigo M (1998) AntNet: Distributed Stigmergetic Control for Communications Networks. J Artificial Intelligence Res (JAIR) 9:317–365

Dorigo M, Stützle T (2004) Ant colony optimization. MIT Press

Eberhart RC, Simpson P, Dobbins R (1996) Computational intelligence PC tools. AP Professional, San Diego, CA, Chapter 6, pp 212–226

Eberhart RC, Shi Y (2000) Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. Proceedings of IEEE congress evolutionary computation, San Diego, CA, pp 84–88

Eberhart RC, Shi Y (2001) Tracking and optimizing dynamic systems with particle swarms. Proceedings of the 2001 congress on evolutionary computation, vol 1, pp 94–100

Feigenbaum EA, Buchanan BG, Lederberg J (1971) On Generality and Problem Solving: A Case Study using the DENDRAL Program. In: Meltzer B, Michie D (eds) Machine Intelligence, 6th edn. American Elsevier, New York, pp 165–190

Fogel LJ, Owens AJ, Walsh MJ (1966) Artificial intelligence thorough simulated evolution. John Wiley & Sons, Ltd, Chichester, UK

Friedberg RM (1958) A learning machine: Part I. IBM J 2–13

Gehlhaar DK, Fogel DB (1996) Tuning evolutionary programming for conformationally flexible molecular docking. In: Evolutionary programming: proceedings of the fifth annual conference on evolutionary programming February 29–March 3, 1996, San Diego, California, pp 419–429
Gies D, Rahmat-Samii Y (2003) Particle swarm optimization for reconfigurable phase-differentiated array design. Microwave Opt Technol Lett 38(3):168–175
Goldberg D (1989) Genetic algorithms in search, optimization and machine learning. Addison Wesley
Haykin S (1999) Neural networks – a comprehensive foundation, 2nd edn. Pearson, Delhi, India
Hebb DO (1949) The organization of behavior. Wiley, New York
Heppner F, Grenander U (1990) A stochastic nonlinear model for coordinated bird flocks. In: Krasner S (eds) The ubiquity of chaos. AAAS Publications, Washington, DC
Holland JH (1962) Outline for a logical theory of adaptive systems. J Assoc Comput Machinery 3: 297–314
Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbour
Johnson S (2001) Emergence: the connected lives of ants, brains, cities, and software. Scribner, New York
Kaewkamnerdpong B, Bentley P (2005) Perceptive particle swarm optimization. Proceedings of the seventh international conference on adaptive and natural computing algorithms (ICCANGA 2005)
Kennedy J, Eberhart RC (1995) Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, pp 1942–1948
Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. Proceedings of the conference on systems, man and cybernetics, Piscataway, New Jersey, pp 4104–4109
Kennedy J (1997) The particle swarm: social adaptation of knowledge. Proceedings of international conference evolutionary computation, IEEE, pp 303–308, Piscataway, NJ
Kennedy J (1999) Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance. Proceedings of IEEE Congress on Evolutionary Computation, Piscataway, NJ
Kennedy J (2000) Stereotyping: improving particle swarm performance with cluster analysis. Proceedings of IEEE congress on evolutionary computation, pp 1507–1512, San Diego, CA
Kennedy J, Eberhart RC, Shi Y (2001) Swarm intelligence. Morgan Kaufman, San Francisco, USA
Kennedy J, Mendes R (2002) Population structure and particle swarm performance. Proceedings of Congress on Evolutionary Computation 2:1671–1676
Kennedy J (2003) Bare bones particle swarms. Proceedings of the IEEE swarm intelligence symposium 2003 (SIS 2003), Indianapolis, Indiana, USA, pp 80–87
Koza J (1992) Genetic Programming: on the programming of computers by means of natural selection. MIT Press, Cambridge, MA
Krink T, Vestertroem JS, Riget J (2002) Particle swarm optimization with spatial particle extension. Proceedings of the IEEE congress on evolutionary computation (CEC 2002), Honolulu, Hawaii (2002)
Laskari EC, Parsopoulos KE, Vrahatis MN (2002) Particle swarm optimization for integer programming. Proceedings of the IEEE
McCulloch WS, Pitts W (1943) A logical calculus of the ideas imminent in nervous activity. Bull Mathematical Biophys 5:115–133
Mendes R, Kennedy J, Neves J (2003) Watch thy neighbor or how the swarm can learn from its environment. Proceedings of IEEE swarm intelligence symposium, Indianapolis, Indiana, pp 88–94
Minsky M (1986) The society of mind. Simon and Schuster, New York
Monson CK, Seppi KD (2004) The Kalman swarm. Proceedings of the genetic and evolutionary computation conference (GECCO), Seattle, Washington
Monson CK, Seppi KD (2005a) Bayesian optimization models for particle swarm. Proceedings of genetic and evolutionary computation conference (GECCO), ACM, Washington, DC
Monson CK, Seppi KD (2005b) Improving on the Kalman swarm extracting its essential characteristics. Proceedings of genetic and evolutionary computation conference (GECCO), ACM, Washington, DC
Nelder JA, Mead R (1965) A Simplex Method for Function Minimization. Computer J 7:308–313
Ozcan E, Mohan CK (1998) Analysis of a simple particle swarm optimization system. Intelligent Engineering Systems Through Artificial Neural Networks 253–258
Ozcan E, Mohan CK (1999) Particle swarm optimization: surfing the waves. Proceedings of IEEE congress on evolutionary computation, Washington, DC
Parrott D, Li X (2004) A particle swarm model for tracking multiple peaks in a dynamic environment using speciation. In: Proceeding of the 2004 congress on evolutionary computation (CEC'04), IEEE Service Center, Piscataway, NJ, pp 98–103, 08855-1331
Parsopoulos KE, Vrahatis MN (2002) Initializing the particle swarm optimizer using the nonlinear simplex method. Advances in intelligent systems, fuzzy systems, evolutionary computation, pp 216–221
Parsopoulos KE, Vrahatis MN (2004) UPSO: a unified particle swarm optimization scheme. In: Lecture series on computer and computational sciences, Proceedings of international conference on

computational methods in sciences and engineering (ICCMSE 2004), VSP International Science Publishers, Zeist, The Netherlands, pp 868–873
Parsopoulos KE, Vrahatis MN (2005a) Unified Particle Swarm Optimization in Dynamic Environments. In: Rothlauf F et al (eds) EvoWorkshops 2005, LNCS 3449, pp 590–599
Parsopoulos KE, Vrahatis MN (2005b) Unified particle swarm optimization for tackling operations research problems. Proceedings swarm intelligence symposium SIS 2005, pp 53–59
Ratnaweera A, Halgamuge SK, Watson HC (2004) Self-organising hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans Evol Comput 8(3):240–255
Rechenberg I (1965) Cybernetic solution path of an experimental problem. Royal Aircraft Establishment, Library Translation No 1122, August
Reeves WT (1983) Particle systems – a technique for modeling a class of fuzzy objects. ACM Trans Graphics 2(2):91–108
Reynolds CW (1987) Flocks, herds, and schools: a distributed behavioral model. Computer Graphics 21(4):25–34 (Proc SIGGRAPH '87)
Riget J, Vesterstrøm JS (2002) A diversity-guided particle swarm optimizer – the ARPSO. EVALife Technical Report no 2002–2002
Shi Y, Eberhart RC (1998) A modified particle swarm optimizer. Proceedings of the IEEE international conference on evolutionary computation, pp 69–73. IEEE Press, Piscataway, NJ
Schutte JF, Reinbolt JA, Fregly BJ, Haftka RT, George AD (2004) Parallel Global Optimization with the Particle Swarm Algorithm. Int J Numer Meth Eng 61(13):2296–2315, John Wiley and Sons Ltd, Great Britain
Silva A, Neves A, Costa E (2002) An empirical comparison of particle swarm and predator prey optimization. In Proceedings of 13th Irish international conference on artificial intelligence and cognitive science 2464:103–110
Sipper M, Sanchez E, Mange D, Tomassini M, Pérez-Uribe A, Stauffer A (1998) An introduction to bio-inspired machines. In: Mange D, Tomassini M (eds) Bio-inspired computing machines towards novel computational architectures. Presses Polytechniques et Universitaires Romandes, Luasanne, Switzerland, pp 1–12
Storn R, Price K (1995) Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. Technical Report TR-95-012, ICSI Available from: http://http.icsi.berkeley.edu/~storn/litera.html
Trelea IC (2003) The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Lett 85:317–325
Turing AM (1952) The chemical basis of morphogenesis. Philosophical Trans Royal Society of London, series B 641:237
van den Bergh F, Engelbrecht AP (2002) A new locally convergent particle swarm optimizer. Proceedings of IEEE conference on systems, man and cybernetics, Hammamet, Tunisia
Veeramachaneni K, Peram T, Mohan CK, Osadciw LA (2003) Optimization using particle swarms with near neighbor interactions. Proceedings of genetic and evolutionary computation conference (GECCO), LNCS 2723, Chicago, IL, pp 110–121
Venter G, Sobieszczanski-Sobieski J (2005). A parallel particle swarm optimization algorithm accelerated by asynchronous evaluations. 6th world congresses of structural and multidisciplinary optimization. Rio de Janerio, Brazil, 30 May–03 June 2005
Wolpert DH, Macready WG (1997) No Free Lunch Theorems for Optimization. IEEE Trans Evol Comput 1(1):67–82
Xie XF, Zhang WJ, Yang ZL (2002) A dissipative particle swarm optimization. Congress on evolutionary computation, Honolulu, HI, USA, 2002:1456–1461
Zhang L, Yu H, Hu S (2005) Optimal choice of parameters for particle swarm optimization. J Zhejiang Univ Sci 528–534