Chapter 11
Swarm Intelligence

Xiaodong Li and Maurice Clerc

Abstract Swarm Intelligence (SI) is an Artificial Intelligence (AI) discipline that studies the collective behaviours of artificial and natural systems such as those of insects or animals. SI is seen as a new concept of AI and is becoming increasingly accepted in the literature. SI techniques are typically inspired by natural phenomena, and they have exhibited remarkable capabilities in solving problems that are often perceived to be challenging to conventional computational techniques. Although an SI system lacks a centralized control, the system at the swarm (or population) level reveals remarkable complex and self-organizing behaviours, often as the result of local interactions among individuals in the swarm as well as individuals with the environment, based on very simple interaction rules.

11.1 Introduction

Swarm Intelligence refers to a family of modern Artificial Intelligence techniques that are inspired by the collective behaviours exhibited by social insects and animals, as well as human societies. Many such phenomena can be observed in nature, such as ant foraging behaviours, bird flocking, fish schooling, animal herding, and many more. Even though individual ants are simple insects and do not exhibit

The original version of this chapter was revised: Acknowledgements section has been revised. The correction to this chapter is available at https://doi.org/10.1007/978-3-319-91086-4_19

X. Li (✉)
School of Science (Computer Science), RMIT University, Melbourne, VIC, Australia
e-mail: xiaodong.li@rmit.edu.au

M. Clerc
Independent Consultant, Groisy, France
e-mail: maurice.clerc@writeme.com

M. Gendreau, J.-Y. Potvin (eds.), *Handbook of Metaheuristics*,
International Series in Operations Research & Management Science 272,
https://doi.org/10.1007/978-3-319-91086-4_11

sophisticated behaviour, many ants working together can achieve fairly complex tasks. An SI system typically consists of a population of individuals. These individuals are usually very simple agents that on their own do not exhibit complex behaviours. However, complex global patterns may emerge from interactions between these agents and the agents with the environment. An intriguing property of an SI system is its ability to behave in a complex and self-organized way without any specific individual taking control of everything. In other words, even without any teleological principle, a common goal may nevertheless be reached.

One definition on Swarm Intelligence provided by Kennedy [44], the inventor of Particle Swarm Optimization (PSO), captures very nicely the essence of SI:

> "*Swarm intelligence refers to a kind of problem-solving ability that emerges in the interactions of simple information processing units. The concept of a swarm suggests multiplicity, stochasticity, randomness, and messiness, and the concept of intelligence suggests that the problem-solving method is somehow successful. The information processing units that compose a swarm can be animate, mechanical, computational, or mathematical; they can be insects, birds, or human beings; they can be array elements, robots, or standalone workstations; they can be real or imaginary. Their coupling can have a wide range of characteristics, but there must be interaction among the units.*"

SI techniques are problem solving techniques emulating this sort of social behaviours that are observed in nature. In essence, the problem solving ability of an SI technique is derived from the interactions among many simple information processing units (or agents). Given the distributed nature of the system, SI techniques tend to be more robust and scalable than conventional techniques. The term of *Swarm Intelligence* was first coined by Beni and Wang [1] in the context of cellular robotic systems. Since then, this term has been adopted in much broader research areas [9, 10].

The purpose of this chapter is to provide an introduction to SI and how it complements the traditional definition of Artificial Intelligence. Several biological examples as inspirations for SI techniques will be presented, as well as the SI metaphor of the human society. The application of SI principles to optimization is in particular prevalent among its many application areas. This chapter will focus on providing a detailed account on one of the most popular SI techniques, Particle Swarm Optimization (PSO). In particular, the chapter will present the canonical PSO and its variants, and provide an illustration of swarm dynamics through a simplified PSO. The chapter will also discuss several popular PSO application areas and its recent theoretical developments.

Traditionally intelligence has been considered as a trait of an individual. Kennedy et al. remarked [46]:

"*The early AI researchers had made an important assumption, so fundamental that it was never stated explicitly nor consciously acknowledged. They assumed that cognition is something inside an individuals head. An AI program was modeled on the vision of a single disconnected person, processing information inside his or her brain, turning the problem this way and that, rationally and coolly.*"

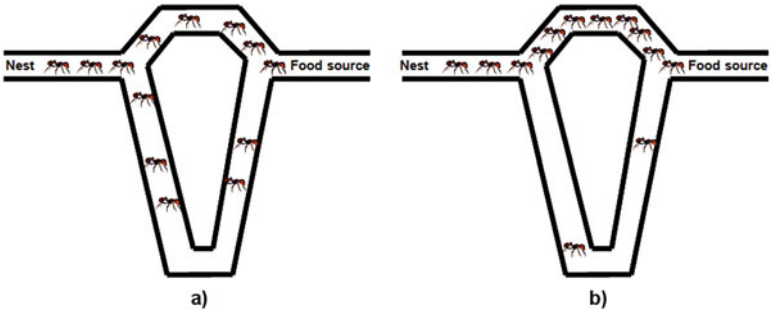

Fig. 11.1 The double-bridge experiment: ants find the shorter path of the two between the nest and the food source; (**a**) at the start, (**b**) after some period, more ants choose the shorter path

Fig. 11.2 Biological examples: flock of birds in flight (top left); fish schools (top right); domes built by termites (bottom left); and honey bees (bottom right)

SI can be regarded as a *broader concept of intelligence* since it emphasizes the fact that intelligence should be modeled in a social context, as a result of interaction with one another. Intelligence should be seen as a collective entity rather than a single isolated one.

11.2 Biological Examples

There is an abundance of social behaviour examples among insects and animals in nature that exhibit emergent intelligent properties [9]. A few examples include:

Ants exhibit interesting path-finding behaviours as they go out searching for food. A well known biological example is the double-bridge experiment, where two bridges of different lengths are placed between the ant nest and the food source (Fig. 11.1). The ants are set out to reach the food source and bring the food back to the nest. Since ants leave pheromone trails as they move around, the path with more ants crossing it will have a higher intensity level of pheromone than the other one. Although at the start of foraging, there is no pheromone on the two paths and there is a probability of 50% of going along either of the two bridges (Fig. 11.1a), after a certain period of time, as more ants come back via the shorter path, the intensity level of pheromone on the shorter path increases. Because ants tend to follow the path with a higher intensity level of pheromone, there will be more and more ants choosing the shorter path to reach the food source. Eventually almost all ants would converge onto the shorter path (Fig. 11.1b). It is remarkable that though no single ant knows about how to find the shorter path, many ants working together manage to achieve the task.

Birds fly in flocks to increase their chance of survival, finding food sources, and avoiding predators. By staying in a flock (Fig. 11.2 top left), birds gain several benefits. One major benefit is the so called "safety-in-numbers", since if a predator approaches the flock, it is more likely to be seen by at least some of the birds in the flock than if a bird is just on its own. The alarm message can be quickly passed onto other birds in the vicinity, and soon to the entire flock. Staying in a flock also serves as a distraction, as the predator may struggle to single out any specific bird. Birds in a flock are more efficient in foraging—if any bird spots the food location and dive towards it, this information can be passed onto others quickly, thus the whole flock benefits. Flying in a flock following a certain pattern also improves the efficiency of the flight, due to better aerodynamics.

Many species of fish swim in schools so as to minimize their energy consumption and to escape from predators' attack. Fish schooling (Fig. 11.2 top right) often refers to the fact that fish swim in groups in a highly coordinated manner, e.g., in the same direction. A fish school may appear to have a life of its own, as they move in unison like one single entity. It is amazing to see the direction or speed of hundreds of fish change almost at the same exact instant. By staying in a school, each individual fish can look out for one another, helping them to avoid a

predator's attack. By swimming in a certain formation following one another, fish can reduce their body friction with the water thereby keeping energy consumption at a very low level.

Termites build sophisticated domed structures as a result of decentralized control. Individual termites participate in building a dome by following some very simple rules (Fig. 11.2 bottom left). For example, termites carry dirt in their mouths, and move in the direction of the strongest pheromone intensity, and then deposit the dirt where the smell is the strongest. Initially termites move randomly and only a number of small pillars are built. These pillars also happen to be the places visited by a larger number of termites, thereby the pheromone intensity is higher there. As more termites deposit their loads in a place, the more attractive this place is to other termites, resulting in a positive feedback loop. Since the deposit tends to be made on the inner side of the pillars, more and more build-up is formed on the inward facing side, eventually resulting in an arch.

Honey bees perform waggle dances to inform other bees about the good sites of food sources. Honey bees use dance as a mechanism to convey information about the direction and distance of the food source (Fig. 11.2 bottom right). Dancing honey bees adjust both the duration and vigor of the dance to inform other bees about good sites of the food sources. The duration of the dance is measured by the number of waggle phases, while the vigor is measured by the time interval between waggle phases. The larger number of waggle phases, the more profitable the food source is, hence more bees will be attracted to it.

In human society we learn from each other. SI can be also observed in the human society. People learn from each other. Knowledge spreads from person to person. Culture emerges from populations. Human society has this remarkable ability to self-organize and adapt. A city like New York has several hundreds of bakeries to supply bread on a daily basis. No one dictates where exactly these bakeries should be located. Yet, these bakeries manage to do a good job of catering to the people living there. As a psychologist, Kennedy et al. [46] mention that the human society operates at three different levels, from individuals, to groups, and to cultures: (1) Individuals learn locally from their neighbours. People interact with their neighbours and share insights with each other; (2) Group-level processes emerge as a result of the spread of knowledge through social learning. Regularities in beliefs, attitudes and behaviours across populations can be observed. A society is a self-organizing entity, and its global properties cannot be predicted from its constituent individuals; (3) Culture optimizes cognition. Locally formed insights and innovations are transported by culture to faraway individuals. Combinations of various knowledge results in even more innovations.

SI principles have been applied to a wide range of problems, among which the most prominent one is probably optimization, SI has been the inspiration for developing many new optimization algorithms [8], including the two most representative examples, Particle Swarm Optimization [46] and Ant Colony Optimization [29]. The application of SI principles goes beyond just optimization though, e.g., in data mining [58] and swarm robotics [72]. As a new research field inspired by SI, *swarm robotics* studies physical robots which are designed in such a way that they can

collaboratively achieve tasks that are beyond the capability of any individual robot. This chapter will mainly focus on Particle Swarm Optimization (PSO), perhaps one of most well-known nature-inspired SI optimization technique. In addition, we will also briefly describe SI applications in data mining and swarm robotics. Readers interested in further more general SI techniques can find a wealth of information from some classic readings on the topic [9, 10, 46].

11.3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) was originally proposed by Kennedy and Eberhart [45]. PSO is a meta-heuristic technique inspired by the social behaviours observed in animals, insects and humans. Since its inception, PSO has enjoyed a widespread acceptance among researchers and practitioners as a robust and efficient technique for solving various optimization problems. PSO was also based on a key insight into human social behaviours and cognition, as remarked by Kennedy: "people learn to make sense of the world by talking with other people about it" [44]. This simple observation allowed Kennedy and Eberhart to go on to design a computer program that encodes a population of candidate solutions that iteratively improve through interactions, that is, by sharing information with their neighbours and by making appropriate adjustments.

In PSO, each individual particle of a swarm represents a potential solution, which moves through the problem space, seeking to improve its own position by taking advantage of information collected by itself and its neighbours. The fact that, on the whole, the swarm often converges to an optimal solution (though not always) which is an emergent consequence of the interaction among particles. Basically the particles broadcast their current positions to neighbouring particles. Through some random perturbation, each particle adjusts its position according to its velocity (i.e., rate of change) and the difference between its current position and the best position found so far by its neighbours, as well as the best position found so far by the particle itself. As the PSO model iterates, the swarm converges towards an area of the search space containing high-quality solutions. The swarm as a whole mimics a flock of birds collectively searching for food. As time goes on, the flock gradually converges onto the food location. Locating a good solution in the search space is achieved by the collective effort of many particles interacting with each other.

Each particle's velocity is updated iteratively through its personal best position (i.e., the best position found so far by the particle) and the best position found by the particles in its neighbourhood. As a result, each particle searches around a region defined by its personal best position and the best position in its neighbourhood. If we use vector $\mathbf{v}_i$ to denote the velocity of the i-th particle in the swarm, $\mathbf{x}_i$ its position, $\mathbf{p}_i$ its personal best position, and $\mathbf{p}_g$ the best position found by particles in its neighbourhood (or the entire swarm), $\mathbf{v}_i$ and $\mathbf{x}_i$ in the original PSO algorithm are updated according to the following two equations [45]:

$$\mathbf{v}_i^{NEW} \leftarrow \mathbf{v}_i + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i) + \varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i), \tag{11.1}$$

$$\mathbf{x}_i^{NEW} \leftarrow \mathbf{x}_i + \mathbf{v}_i^{NEW}, \tag{11.2}$$

where $\varphi_1 = c_1\mathbf{R_1}$ and $\varphi_2 = c_2\mathbf{R_2}$. $\mathbf{R}_1$ and $\mathbf{R}_2$ are two separate functions, each returning a vector of random values uniformly generated in the range [0, 1]. c_1 and c_2 are acceleration coefficients. The symbol $\otimes$ denotes component-wise vector multiplication. Equation (11.1) shows that the velocity term $\mathbf{v}_i$ of a particle is determined by three components, the "momentum", "cognitive" and "social" parts. The "momentum" term $\mathbf{v}_i$, represents the previous velocity term which is used to carry the particle in the direction it has travelled so far; the "cognitive" part, $\varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i)$, represents the tendency of the particle to return to the best position it has found so far; and finally the "social" part, $\varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i)$, represents the tendency of the particle to be attracted towards the best position found by the entire swarm.

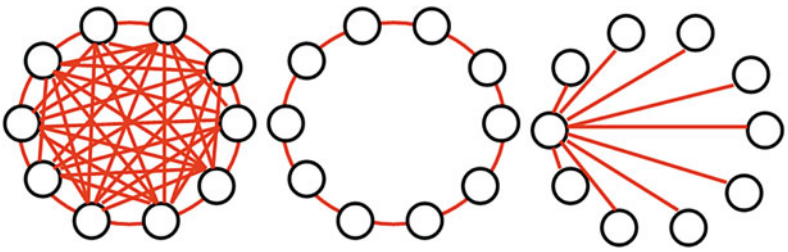

Fig. 11.3 Neighbourhood topologies: fully-connected, ring, and star (from left to right)

Algorithm 1 The PSO algorithm, assuming maximization

```
Randomly generate an initial population
REPEAT
  FOR each particle i
    IF f(x_i) > f(p_i) THEN p_i ← x_i;
    p_g ← max(p_neighbors);
    Update velocity (Eq. (11.1));
    Update position (Eq. (11.2));
  END
UNTIL termination criterion is met;
```

In practice, randomness is obtained with a Random Number Generator (RNG), but the same algorithm may perform differently depending on the chosen RNG. Furthermore, the best performance is not always obtained using the "best" generator. A detailed discussion and how to cope with this issue can be found in [24].

Neighbourhood topologies used in the "social" component can be exploited to control the speed of information propagation among particles. Representative examples of neighbourhood topologies include ring, star, and von Neumann (see Fig. 11.3). Restricted information propagation as a result of using small neighbour-

hood topologies, such as von Neumann usually works better on complex multimodal problems, whereas larger neighbourhoods would perform better on simpler unimodal problems [59]. A PSO algorithm choosing its $\mathbf{p}_g$ from within a restricted local neighbourhood is usually called a *lbest* PSO, whereas a PSO choosing $\mathbf{p}_g$ without any restriction (hence from the entire swarm) is commonly referred to as a *gbest* PSO. Algorithm 1 summarizes a basic PSO algorithm, where $f(.)$ refers to the fitness function, and function $max(\mathbf{p}_{neighbors})$ returns the best position among all the personal bests in the i-th particle's neighbourhood.

Note that the Algorithm 1 is a classical asynchronous model, usually the most efficient one. However, synchronous updates are also possible. In the case of parallel

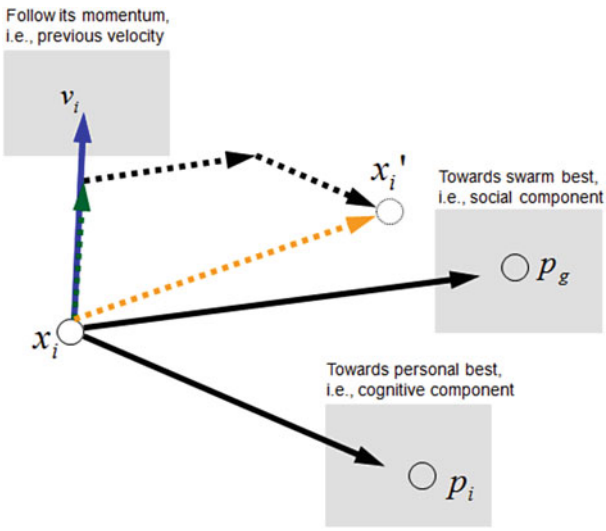

Fig. 11.4 The next movement of particle $\mathbf{x}_i$ is determined by three components, i.e., its previous velocity, its "cognitive" component, and the "social" component

computation, for example, one may use synchronous updates, in which all $\mathbf{p}_i$ are saved first, before updating $\mathbf{p}_g$, and then updating all particles' velocity and position values.

Figure 11.4 shows that particle position $\mathbf{x}_i$ is updated to its next position $\mathbf{x}_i'$ (denoted by a dashed circle), based on three components: the "momentum" term, i.e., the previous $\mathbf{v}_i$ scaled by the inertia weight w, the "cognitive" and "social" components. It is apparent that the new position $\mathbf{x}_i'$ is generated by a linear combination of these three vectors. The particle is shown to have moved to a position somewhere in the vicinity of the mean of the cognitive component $\mathbf{p}_i$ and social component $\mathbf{p}_g$. Note that due to the random coefficient used for each dimension, the "cognitive" and "social" components in Eq. (11.4) may be weighted differently for each dimension. This is indicated by the shaded areas in the figure. Note also that the inertia coefficient w is used to scale the previous velocity term, normally to reduce the "momentum" of the particle. More discussion on w will be provided in the next section.

Earlier studies showed that the velocity as defined in Eq. (11.1) has a tendency to diverge to a large value, resulting in particles flying past the boundaries of the search space, i.e., violating the boundary constraints. This is more likely to happen when a particle is far from its $\mathbf{p}_g$ or $\mathbf{p}_i$. To overcome this problem, a velocity clamping method can be adopted to clamp the maximum velocity value to V_{max} in each dimension of $\mathbf{v}_i$. This method does not necessarily prevent particles from leaving the search space, nor to converge. However, it does limit the particle step size, therefore restricting particles from further divergence.

Note there are many ways to take boundary constraints into account [22, 37]. When a particle tends to leave the search space, we can force it to go back, randomly or not, or to stop it at the boundary. We can even let it fly, but without re-evaluating its position outside the search space. As its memorised previous best position $\mathbf{p}_i$ is inside it, we are sure that it will be back, sooner or later [13].

11.3.1 Inertia Weighted and Constricted PSOs

The two most widely-used PSO models are probably the inertia weighted PSO and the constricted PSO, both representing a further refinement of the original PSO described in the previous section. Note that the $\mathbf{p}_i$ and $\mathbf{p}_g$ in Eq. (11.1) can be collapsed into a single term $\mathbf{p}$ without losing any information:

$$\mathbf{v}_i^{NEW} \leftarrow \mathbf{v}_i + \varphi \otimes (\mathbf{p} - \mathbf{x}_i), \tag{11.3}$$

where $\mathbf{p} = \frac{\varphi_1 \otimes \mathbf{p}_i + \varphi_2 \otimes \mathbf{p}_g}{\varphi_1 + \varphi_2}$, and $\varphi = \varphi_1 + \varphi_2$. Note that the division operator is a component-wise operator here. The vector $\mathbf{p}$ represents the stochastically weighted average of $\mathbf{p}_i$ and $\mathbf{p}_g$. Each particle oscillates around the point $\mathbf{p}$ as a result of Eq. (11.3), when its velocity $\mathbf{v}_i$ is adjusted at each iteration. As the particle gets farther away from $\mathbf{p}$, $\mathbf{v}_i$ becomes smaller until it reverses its direction (since $\mathbf{v}_i$ is iteratively reduced, and then increased in magnitude but in the opposite direction), and the particle then heads in the opposite direction. Note that the movement of the particle does not strictly follow a cyclic pattern, given that $\mathbf{p}$ is a stochastic average. This sort of stochasticity provides the needed variations that allow the particle to find new and better points.

The coefficient vector $\varphi_1 = c_1\mathbf{R_1}$ (or $\varphi_2 = c_2\mathbf{R_2}$) is a vector of randomly generated numbers in the range $[0, c_1]$. Thus, if a coefficient is equal to 0, the associated $\mathbf{p}$ (or $\mathbf{p}_g$) has no effect. If both coefficients are equal to 0, the velocity $\mathbf{v}_i$ will not change from its previous value. When both coefficients are close to 1, then $\mathbf{v}_i$ is likely to be greatly affected.

From Eq. (11.1) it can be seen that the previous velocity term $\mathbf{v}_i$ tends to keep the particle moving in the same direction. A coefficient *inertia weight*, w, can be used to control this influence on the new velocity. The velocity update in Eq. (11.1) can now be revised as:

$$\mathbf{v}_i^{NEW} \leftarrow w\mathbf{v}_i + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i) + \varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i) \tag{11.4}$$

This so-called inertia weighted PSO can converge under certain conditions even without using V_{max} [25]. For $w > 1$, velocities increase over time causing particles to eventually diverge beyond the boundaries of the search space. For $0 < w < 1$, velocities decrease over time eventually reaching 0, thus resulting in convergence. Eberhart and Shi suggested to use a time-varying inertia weight that gradually decreases from 0.9 to 0.4 (with $\varphi = 4.0$) [30].

A more general PSO model employing a *constriction coefficient* χ was introduced by Clerc and Kennedy [25]. Several variants were suggested, among which Constriction Type 1 PSO is shown to be algebraically equivalent to the inertia-weighted PSO. In Constriction Type 1 PSO, the velocity update in Eq. (11.4) can be rewritten as:

$$\mathbf{v}_i^{NEW} \leftarrow \chi(\mathbf{v}_i + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i) + \varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i)), \tag{11.5}$$

where $\chi = \frac{2}{\left|2-\varphi'-\sqrt{\varphi'^2-4\varphi'}\right|}$, and $\varphi' = c_1 + c_2, \varphi' > 4$ (note that φ' is a scaler). If φ' is set to 4.1, and $c_1 = c_2 = 2.05$, then the constriction coefficient χ will be 0.7298. Applying χ in Eq. (11.5) results in the previous velocity to be scaled by 0.7298 and the "cognitive" and "social" parts multiplied by 1.496 (i.e., 0.7298 multiplied by 2.05). Both theoretical and empirical results suggest that the above configuration using a constriction coefficient $\chi = 0.7298$ ensures convergent behaviour [30] without using V_{max}. However, early empirical studies by Eberhart and Shi suggested that it may still be a good idea to use velocity clamping together with the constriction coefficient, which showed improved performance on certain problems.

11.3.2 Memory-Swarm vs. Explorer-Swarm

In PSO, interactions among particles have a significant impact on the particles' behaviour. A distinct feature of PSO, which sets it apart from a typical evolutionary algorithm, is that each particle has its own memory, i.e., its personal best $\mathbf{p}_i$. As remarked by Clerc [23], a swarm can be viewed as comprising two sub-swarms with different functionalities. The first group, *explorer-swarm*, is composed of particles moving around in large step sizes and more frequently, each strongly influenced by its velocity and its previous position (see Eqs. (11.2) and (11.1)). The explorer-swarm explores the search space more broadly. The second group, *memory-swarm*, consists of the personal bests of all particles. This memory-swarm is more stable than the explorer-swarm because personal bests represent the best positions found so far by individual particles. The memory-swarm is more effective in retaining the best positions found so far by the swarm as a whole. Meanwhile, each of these positions can be further improved by the more exploratory particles in the explorer-swarm.

We can use a "graph of influence" to illustrate the sender and receiver of influence for each particle in a swarm. A swarm of seven particles following a ring neighbourhood topology is shown in Fig. 11.5. Here, a particle that informs another particle is called an "informant". The explorer-swarm consists of particles labeled from 1 to 7, and the memory-swarm consists of particles labeled from $m1$ to $m7$, which are the personal bests of particles 1–7. Each particle has three informants: the memories of two neighbouring particles and its own memory. The memory of each particle has also three informants: the two neighbouring particles and the particle itself.

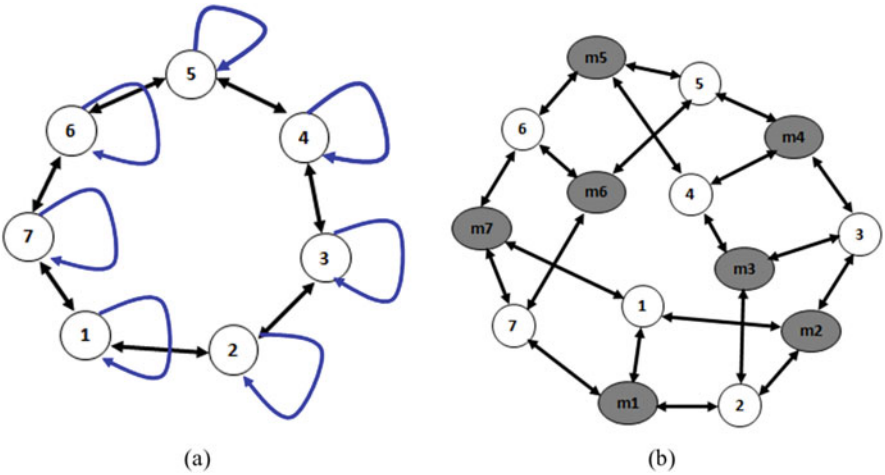

Fig. 11.5 Graphs of influence for a swarm of seven particles on a ring topology [23]; (**a**) each particle interacting directly with its immediate left and right-sided neighbours plus itself; (**b**) still the same swarm as in (**a**), but showing both the *explorer-swarm* (not shaded) and the *memory swarm* (shaded)

11.3.3 Particle Dynamics Through a Simplified Example

It is worth noting that the interactions among particles have a huge impact on the performance of the PSO model. To gain a better understanding of the consequences of such interactions, we study a simplified PSO in this section, where a swarm is reduced to only one or two particles, with just one dimension. We then examine the dynamics of this simplified PSO. Although it is a very simple model, we hope to provide a glimpse of how and why PSO works. We use an example based on [23] to demonstrate the dynamics of such a simplified PSO.

In this simplified PSO, we assume that there is no stochastic component, only one dimension, and the initial position and velocity are pre-specified. With these

assumptions, Eqs. (11.4) and (11.2) can be simplified as follows:

$$v_i^{NEW} \leftarrow wv_i + c_1(p_i - x_i) + c_2(p_g - x_i), \tag{11.6}$$

$$x_i^{NEW} \leftarrow x_i + v_i^{NEW}. \tag{11.7}$$

Note that the subscript i can also be removed in the above equations when there is only one particle. In the following case studies, we also set w, c_1 and c_2 to 0.7 for simplification purposes.

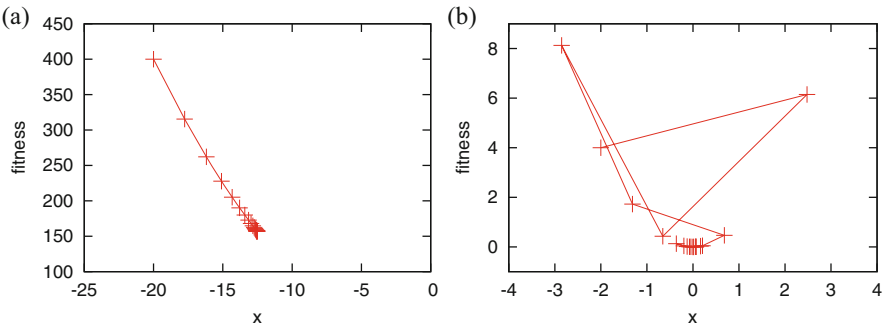

Fig. 11.6 The dynamics of the system is different depending on the initial x and v values: (**a**) when $x = -20$ and $v = 3.2$, the particle prematurely converges to one point, which is not the minimum; (**b**) when $x = -2$ and $v = 6.4$, the particle oscillates around the minimum several times before converging to it

11.3.3.1 One Particle

Let us first assume that there is only a single particle in the swarm. Note that even when there is only one particle, we actually know the information about two positions, its current position x and its personal best position p (since there is only one particle, p_g is the same as p). Let us now consider a simple minimization problem using the one-dimensional Parabola function: $f(x) = x^2$, where $x \in [-20, 20]$. If the initial x and v values are provided, then the future x values can be computed deterministically by iteratively calling Eqs. (11.6) and (11.7). There are two possible scenarios:

- **Case 1:** the initial x and v positions are on the same side of the minimum, e.g., when $x = -20$, $v = 3.2$ (see Fig. 11.6a);
- **Case 2:** The initial x and v positions are on both sides of the minimum, e.g., $x = -2$, $v = 6.4$ (see Fig. 11.6b).

Figure 11.6 shows two startling different dynamics depending on the initial x and v values. In Fig. 11.6a, it can be noted that p is always equal to x, essentially turning Eq. (11.6) into $v \leftarrow wv$. Since w=0.7, v gradually approaches 0 over iterations. As a

result, with increasingly small step sizes, x prematurely converges to a point which is insufficient to reach the minimum. In contrast, Fig. 11.6b shows that when the particle oscillates around the minimum, it manages to converge very closely to the minimum. This time, the iteratively updated p is not always equal to x, resulting in a much better convergence behaviour.

The better convergence behaviour of the particle can be further illustrated in Fig. 11.7 in the phase spaces of v and x. Figure 11.7b shows that the particle oscillates around the minimum with several changes in the direction of velocity v, before the particle converges to the minimum following a spiral trajectory. In contrast, Fig. 11.7a shows that the prematurely converged particle never manages to change the direction of its velocity v.

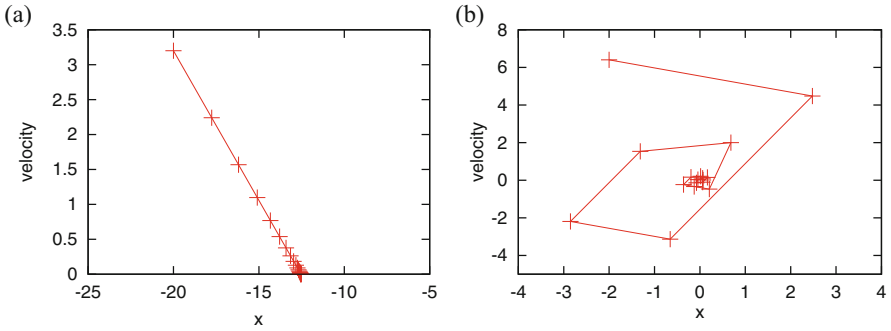

Fig. 11.7 Phase spaces for the two examples used in Fig. 11.6: (**a**) velocity v approaches 0 from a positive number; (**b**) velocity v takes both positive and negative values, approaching 0 through a spiral trajectory

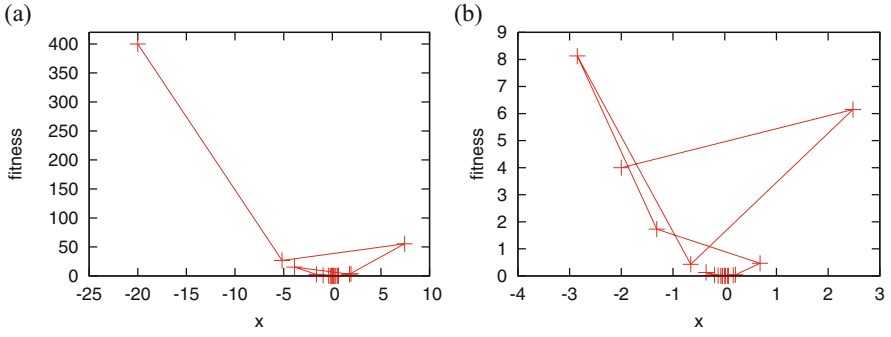

Fig. 11.8 A swarm of two particles based on two different cases: $x = -20$, v $= 3.2$ and $x = -2$, $v = 6.4$. Now the two particles interact with each other: (**a**) particle 1's convergence benefits from the information provided by particle 2; (**b**) particle 2's convergence behaviour is unaffected, since no useful information comes from particle 1

11.3.3.2 Two Particles

Now, let us consider a swarm of two particles. In this case, we know four positions: the two particles' current positions and their two personal bests (i.e., memories). Here, each particle informs only its memory, but gets informed by both its own memory and the other particle's memory.

Figure 11.8 shows the convergence behaviours of the two interacting particles in a swarm. In this example, $m2$ is always better than $m1$, hence particle 2 does not benefit from the presence of particle 1. The convergence behaviour of particle 2 as shown in Fig. 11.8b is unaffected and is identical to the case illustrated in Fig. 11.6b. On the other hand, the trajectory of particle 1 in Fig. 11.8a shows that it benefits from the presence of particle 2. Since $m2$ is better than $m1$, this information is used to improve the convergence behaviour of particle 1 towards the minimum.

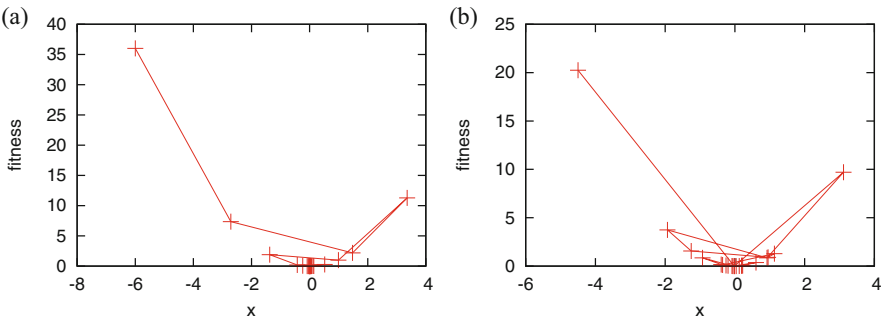

Fig. 11.9 A swarm of two particles both benefiting from interaction with each other. (**a**) particle 1's convergence trajectory; (**b**) particle 2's convergence trajectory

Figure 11.9 shows a more general case where each particle is influenced by the memory of the other. Both particles benefit from receiving the memory to the other particle. Improved convergence for each particle is evident.

11.4 PSO Variants

Apart from the canonical PSO models such as the inertia weighted and constriction based PSO, a few other PSO variants have been increasingly accepted in the optimization research community.

11.4.1 Fully Informed PSO

It can be noted that Eq. (11.3) suggests that a particle tends to converge towards a point determined by $\mathbf{p} = \frac{\varphi_1 \otimes \mathbf{p}_i + \varphi_2 \otimes \mathbf{p}_g}{\varphi_1 + \varphi_2}$, where $\varphi = \varphi_1 + \varphi_2$. In the Fully Informed Particle Swarm (FIPS) proposed by Mendes et al. [59], $\mathbf{p}$ can be further generalized to any number of terms:

$$\mathbf{p} = \frac{\sum_{k \in N_i} \mathbf{R}[0, \frac{c_{max}}{|N_i|}] \otimes \mathbf{p}_k}{\sum_{k \in N_i} \varphi_k}, \tag{11.8}$$

where $\mathbf{p}_k$ denotes the best position found by the k-th particle in N_i, which is a set of neighbours that includes the particle i, and c_{max} denotes the acceleration coefficient which is usually shared among all $|N_i|$ neighbours. $\mathbf{R}[0, \frac{c_{max}}{|N_i|}]$ is a function returning a vector of numbers randomly and is uniformly generated in the range $[0, \frac{c_{max}}{|N_i|}]$. Note again that the division is a component-wise operator.

If we set $k = 2$ and $\mathbf{p}_1 = \mathbf{p}_i$, and $\mathbf{p}_2 = \mathbf{p}_g$, with both $\mathbf{p}_i, \mathbf{p}_g \in N_i$, then the Constriction Type 1 PSO is just a special case of the more general PSO— FIPS defined in Eq. (11.8). A significant implication of Eq. (11.8) is that it allows us to think more freely about other terms of influence than just $\mathbf{p}_i$ and $\mathbf{p}_g$ [43, 59].

11.4.2 Bare-Bones PSO

Kennedy proposed a PSO variant which does not use the velocity term $\mathbf{v}_i$, so called bare-bones PSO [42]. Each dimension $d = 1, \ldots, D$ of the new position of a particle is randomly selected from a Gaussian distribution, with a mean defined by the average of $p_{i,d}$ and $p_{g,d}$ and a standard deviation set to the distance between $p_{i,d}$ and $p_{g,d}$:

$$x_{i,d} \leftarrow N(\frac{p_{i,d} + p_{g,d}}{2}, ||p_{i,d} - p_{g,d}||). \tag{11.9}$$

Note that no velocity term is used in Eq. (11.9). The new particle position is simply generated via the Gaussian distribution. Other sampling distributions may also be employed [23, 70]. For example, Richer and Blackwell [70] employed a Lévy distribution instead of the Gaussian. The Lévy distribution is also bell-shaped like the Gaussian but with fatter tails. A parameter α can be tuned to obtain a series of different shapes between the Cauchy and Gaussian distributions. Richer and Blackwell found that the bare-bones PSO using the Lévy distribution with $\alpha = 1.4$ was able to reproduce the performance of the canonical PSO [70].

11.4.3 Binary PSO

Although the canonical PSO was designed for continuous optimization, it can be extended to operate on binary search spaces. Kennedy et al. [46] developed a simple binary PSO by using a sigmoid function $s(.)$ to transform the velocity term in the canonical PSO into a probability threshold to determine if the d-th element of a binary string representing $\mathbf{x}_i$ should be 0 or 1:

$$s(v_{id}) = \frac{1}{1+exp(-v_{id})}, \tag{11.10}$$

and

$$x_{id} = \begin{cases} 1 & \text{if } R \leq s(v_{id}) \\ 0 & \text{otherwise} \end{cases}. \tag{11.11}$$

That is, if a uniformly drawn random number R from [0, 1] is smaller than $s(v_{id})$, then x_{id} is set to 1, otherwise it is set to 0. Equation (11.10) is iterated over each dimension for each particle to see if x_{id} results in a better fitness than p_{id}, and if so, p_{id} is updated.

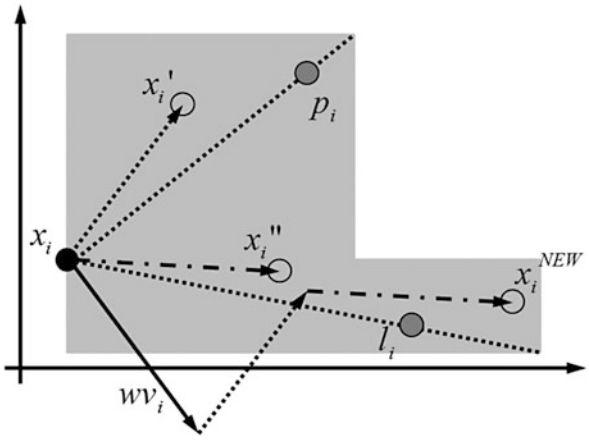

Fig. 11.10 The points x_i' and x_i'' are chosen at random inside two hyper-parallelepipeds parallel to the coordinate axes

11.4.4 Discrete PSO

PSO can also be extended to solve discrete or mixed (continuous and discrete) optimization problems such as knapsack, quadratic assignment, and traveling salesman problems. PSO can be adapted to work with discrete variables by simply discretizing the values obtained after computing the velocity and position update equations, or using combinatorial methods (what is usually done for knapsack, quadratic as-

signment, and traveling salesman problems) [21, 23]. In the latter case, designing a domain specific velocity operator following the general PSO principle is critical, i.e., each particle has a velocity, has knowledge of the best position visited so far, and the best position in the swarm (Fig. 11.4). For example, Goldbarg et al. [34] used customized local search operators (e.g., swap operator) and the path-relinking procedure for effectively solving the traveling salesman problem. An empirical study on several such discrete PSOs on the traveling salesman problem [57] shows that they can be competitive with ACO algorithms.

11.4.5 SPSO-2011

A major shortcoming of both inertia weighted and constricted PSO is that they are not "rotation invariant" [74, 75], meaning that their performances depend on the orientation of the coordinate axes. Note that it is rarely the case in real-world situations that we need to rotate the search space of a problem. But, the appeal of such a "rotation invariant" algorithm is that its behaviour does not depend on the orientation of the search space, hence it is more likely to perform more consistently. The rotation of the search space here merely introduces variable interaction, often making the problem more difficult to handle.

Figure 11.10 provides an example to illustrate this issue with the canonical PSO. Here, the cognitive and social components of PSO (c.f., either Eq. (11.4) or (11.5)) can be seen as parts of the following two hyper-parallelepipeds (each is the Euclidean product of D real intervals):

$$x_i' = \bigotimes_{d=1}^{D} [x_{i,d}, x_{i,d} + c(p_{i,d} - x_{i,d})], \tag{11.12}$$

$$x_i'' = \bigotimes_{d=1}^{D} [x_{i,d}, x_{i,d} + c(l_{i,d} - x_{i,d})], \tag{11.13}$$

where $l_{i,d}$ denotes the neighbourhood best for the i-th particle. Its new position at the next iteration will then be:

$$\mathbf{x}_i^{NEW} \leftarrow w\mathbf{v}_i + (\mathbf{x}_i' - \mathbf{x}_i) + (\mathbf{x}_i'' - \mathbf{x}_i). \tag{11.14}$$

As can be seen in Fig. 11.10, both x_i' and x_i'' are sampled uniformly from the two hyper-parallelepipeds with their sides parallel to the coordinate axes. It shows that the newly generated position $\mathbf{x}_i^{NEW}$ depends on the orientation of the coordinate axes. If we consider the distribution of all possible next positions (DNPP), its support is a D-rectangle whose density is not uniform (denser near the center). A more complete analysis of this phenomenon is given in [75].

To address this problem, Clerc proposed SPSO-2011 [20], where the velocity term is modified in a "geometrical" way that does not depend on the system of coordinates. The key idea is to define the center of gravity $\mathbf{G}_i$ from three existing

points: the current position $\mathbf{x}_i$, a point a bit beyond the best personal best position $\mathbf{p}'_i$, and a point a bit beyond the best local neighbourhood point $\mathbf{l}'_i$, as follows:

$$\mathbf{G}_i = \frac{\mathbf{x}_i + \mathbf{p}'_i + \mathbf{l}'_i}{3}, \tag{11.15}$$

where $\mathbf{p}'_i = \mathbf{x}_i + c_1\varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i)$ and $\mathbf{l}'_i = \mathbf{x}_i + c_1\varphi_1 \otimes (\mathbf{l}_i - \mathbf{x}_i)$.

A random point $\mathbf{x}'_i$ can then be selected (may be according to a uniform distribution) in the hypersphere $H_i(\mathbf{G}_i, ||\mathbf{G}_i - \mathbf{x}_i||)$ of center $\mathbf{G}_i$ with a radius $||\mathbf{G}_i - \mathbf{x}_i||$. The velocity update equation is now:

$$\mathbf{v}_i^{NEW} \leftarrow w\mathbf{v}_i + \mathbf{x}'_i - \mathbf{x}_i, \tag{11.16}$$

and the position update equation is:

$$\mathbf{x}_i^{NEW} \leftarrow w\mathbf{v}_i^{NEW} + \mathbf{x}'_i \tag{11.17}$$

Figure 11.11 shows how such a new point $\mathbf{x}'_i$ is selected from the hypersphere H_i, which is now rotation invariant.

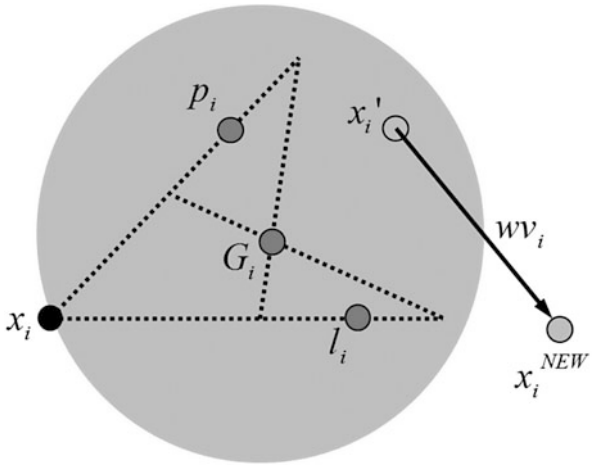

Fig. 11.11 The points x'_i is chosen at random inside the hypersphere $H_i(G_i, ||G_i - x_i||)$

SPSO-2011 also adopts an adaptive random topology (which was previously defined in SPSO-2006 [23]). It means that at the beginning, and after each unsuccessful iteration (no improvement of the best known fitness value), the graph of influence is modified: each particle informs at random K particles, including itself. The parameter K is usually set to 3, but may follow some distribution to allow occasionally the selection of a larger number of particles. Thus, on average, each particle is informed by K other particles, but may be informed by a much larger number of particles (possibly the entire swarm) with a small probability.

SPSO-2011 represents a major enhancement to PSO. The incorporation of these two features, i.e., *adaptive random topology* and *rotational invariance*, has resulted in competitive performances against other state-of-the-art meta-heuristic algorithms on the CEC'2013 real-parameter optimization test function suite [84].

11.4.6 Other PSO Variants

Many PSO variants have been developed since it was first introduced. In particular, to tackle the problem of premature convergence often encountered when using the canonical PSO, several PSO variants incorporate some diversity maintenance mechanisms. For example, ARPSO (attractive and repulsive PSO) uses a diversity measure to trigger alternating phases of attraction and repulsion [71]; a PSO with self-organized criticality was also developed [56]; another PSO variant based on fitness-distance-ratio (FDR-PSO) was developed to encourage interactions among particles with high fitness and close to each other [80]. This FDR-PSO can be seen as using a dynamically defined neighbourhood topology. Various neighbourhood topologies have been adopted to restrict particle interactions. In particular, the von Neumann neighbourhood topology has been shown to provide good performance across a range of test functions [59, 76]. In [40], a H-PSO (Hierarchical PSO) was proposed, where a hierarchical tree structure is adopted to restrict the interactions among particles. Each particle is influenced only by its own personal best position and by the best position of the particle that is directly above it in the hierarchy. Another highly successful PSO variant is CLPSO (Comprehensive Learning PSO) [55], where more historical information about particles' personal best is harnessed through a learning method to better preserve swarm diversity. The Gaussian distribution was employed as a mutation operator to create more diversity in a hybrid PSO variant [38]. A cooperative PSO, similar to coevolutionary algorithms, was also proposed in [78]. It should be noted that many more PSO variants can be found in the literature.

11.5 PSO Applications

One of the earliest PSO applications was the optimization of neural network structures [46], where PSO replaced the traditional back-propagation learning algorithm in a multilayer *Perceptron*. Due to the fast convergence property of PSO, using it to train a neural network can potentially save a considerable amount of computational time as compared to other optimization methods. There are numerous examples of PSO applications for a wide range of optimization problems, from classical problems such as scheduling, traveling salesman problem, neural network training, to highly specialized problem domains such as reactive power and voltage control [83], biomedical image registration [81], and even music composition [4]. PSO is also a

popular choice for multiobjective optimization [69] dynamic optimization [63], and multimodal optimization problems [49], which will be described in more detail in the subsequent sections.

11.5.1 Multiobjective Optimization

Multiobjective optimization problems represent an important class of real-world problems. Typically such problems involve trade-offs. For example, a car manufacturer wants to maximize its profit, but at the same time wants to minimize its production cost. These objectives are usually conflicting with each other, e.g., a higher profit would increase the production cost. Generally speaking, there is no single optimal solution. Often the manufacturer needs to consider many possible "trade-off" solutions before choosing the one that suits its need. The curve or surface (for more than two objectives) describing the optimal trade-off solutions between objectives is known as the Pareto front. A multiobjective optimization algorithm is required to locate solutions as closely as possible to the Pareto front, and at the same time maintaining a good spread of these solutions along the Pareto front. Several questions must first be answered before one can apply PSO to multiobjective optimization:

- How to choose $\mathbf{p_g}$ (i.e., a swarm leader) for each particle? The PSO model needs to favor non-dominated particles over dominated ones, and propels the swarm to spread towards different parts of the Pareto front, not just towards a single point. This would require particles to be led by different swarm leaders.
- How to identify non-dominated particles with respect to all particles' current positions and personal best positions? and how to retain these solutions during the search? One strategy is to combine all particles' personal best positions ($\mathbf{p}_i$) and current positions ($\mathbf{x}_i$), and then extract the non-dominated solutions from this combined population.
- How to maintain particle diversity so that a set of well-distributed solutions can be found along the Pareto front? Some classic niching methods (e.g., crowding [27] or sharing [33]) can be adopted for this purpose.

The earliest work on PSO for solving multiobjective optimization was proposed by Moore and Chapman [60], where an *lbest* PSO was used, and $\mathbf{p_g}$ was chosen from a local neighbourhood using a ring topology. All personal best positions were kept in an archive. At each particle update, the current position is compared with solutions in this archive to see if the current position can be considered as a non-dominated solution. Then the archive is subsequently updated (at each iteration) to ensure it retains only non-dominated solutions.

It was not until 2002 that the next research work on multiobjective PSO appeared—Coello and Lechuga [26] proposed MOPSO (Multiobjective PSO) which

also uses an external archive to store non-dominated solutions. The diversity of solutions is maintained by keeping only one solution within each hypercube specified by the user in the objective space. Parsopoulos and Vrahatis adopted the classical weighted-sum approach in [65]. By using a set of gradually changing weights, their approach was able to find a diverse set of solutions along the Pareto front. In [32], Fieldsend and Singh proposed a PSO using a *dominated tree* structure to store non-dominated solutions. The selection of leaders was also based on this structure. To maintain a better diversity, a *turbulence* operator was adopted to function as a 'mutation' operator in order to perturb the velocity value of a particle. To make effective extraction of non-dominated solutions from a PSO population, NSPSO (Non-dominated Sorting PSO) was proposed in [47], which follows the main idea of the well-known genetic algorithm NSGA II [28]. In NSPSO, instead of comparing solely a particle's personal best with its new position, all particles' personal bests and their new positions are first combined to form a temporary population. The dominance comparisons are performed over all individuals in this temporary population. This strategy allows more non-dominated solutions to be discovered and in a much faster way than early multiobjective PSO algorithms.

Many more multiobjective PSO algorithms have been proposed over the years. A survey in 2006 showed that there were 25 different PSO variants at that time for handling multiobjective optimization problems [69]. Multiobjective PSO has also been combined with classic MCDM (Multi-Criteria Decision Making) user prefer-

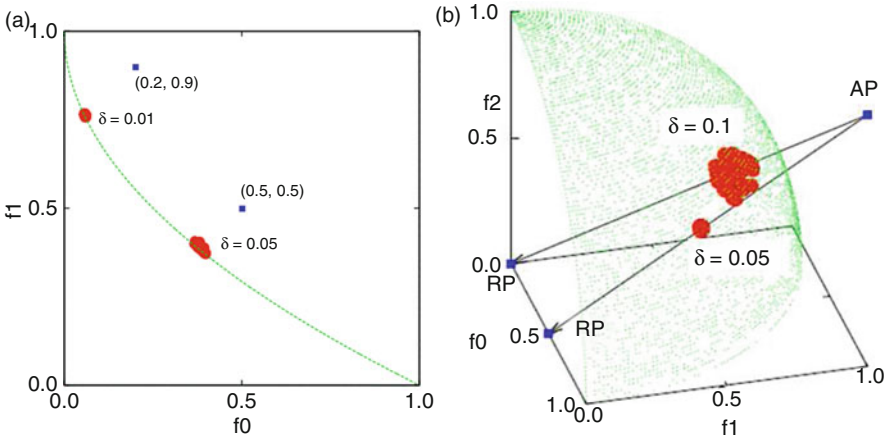

Fig. 11.12 Particles of a multiobjective PSO model have converged only around the preferred regions (of the Pareto front), as indicated by the reference points supplied by a decision maker, e.g., points (0.2, 0.9) and (0.5, 0.5) in the 2-objective space, as shown in (**a**); and AP (Aspiration Point) and RP (Reservation Point) in the 3-objective space, as shown in (**b**). Note that δ denotes a control parameter specifying the coverage of the preferred region

ences based techniques to allow a decision maker to specify a preferred region of convergence before running the algorithm. By using this preference information,

a multiobjective PSO can focus its search effort more effectively on the preferred region in the objective space [82]. This could save a substantial amount of computational time especially when the number of objectives is large. An example of convergence to a preferred region of the Pareto-front by this multiobjective PSO is presented in Fig. 11.12. This multiobjective PSO has also been shown to be an efficient optimizer for a practical aerodynamic design problem [17, 18].

11.5.2 Optimization in Dynamic Environments

Many real-world optimization problems are dynamic by nature, and require optimization algorithms to adapt to the changing optima over time. For example, traffic conditions in a city change dynamically and continuously. What might be regarded as an optimal route at one time might not be optimal a bit later. In contrast to optimization towards a static optimum, the goal in a dynamic environment is to track as closely as possible the dynamically changing optima.

A defining characteristic of PSO is its fast convergent behaviour and inherent adaptability. Particles can adaptively adjust their positions based on their dynamic interactions with other particles in the population. This makes PSO especially appealing as a potential solution to dynamic optimization problems. Several studies showed that the canonical PSO must be adapted to meet the additional challenges of dynamic optimization problems [5, 6, 15, 16, 31, 39, 51, 63]. In particular, the following questions need to be answered: (1) How do we detect that a change has actually occurred? (2) Which response strategies are appropriate once a change is detected? (3) How do we handle the issue of "out-of-date" memory as particles personal best positions become invalid once the environment has changed? (4) How do we handle the trade-off issue between convergence (in order to locate optima) and diversity (in order to relocate changed optima)?

An early work on the application of PSO for dynamic optimization was carried out by Eberhart and Shi [31], where an inertia-weighted PSO was used to track the optimum of a unimodal parabolic function whose maxima changed at regular interval. It was found that, under certain circumstances, the performance of PSO was comparable to or even better than that of evolutionary algorithms.

To detect changes, one could use a randomly chosen *sentry particle* at each iteration [15]. The sentry particle can be evaluated before each iteration, comparing its fitness with its previous fitness value. If the two values are different, suggesting that the environment has changed, then the whole swarm gets alerted and several possible responses can then be triggered. Another simple strategy is to re-evaluate $\mathbf{p}_g$ and a second-best particle to detect if a change has occurred [39].

Various response strategies have been proposed. To deal with the issue of "out-of-date" memory as the environment changes, we can periodically replace all personal best positions by their corresponding current positions when a change has been detected [16]. This allows particles to forget their past experience and use only up-to-date knowledge about the new environment. Re-randomizing different propor-

tions of the swarm was also suggested in order to maintain some degree of diversity and better track the optima after a change [39]. However, this approach suffers from possible information loss since the re-randomized portion of the population does not retain any, potentially useful, information from the past iterations. Another idea is to introduce the so called "charged swarms" [3], where mutually repelling *charged* particles orbit around the nucleus of neutral particles (conventional PSO particles) [6, 53]. Whereas charged particles allow the swarm to better adapt to changes in the environment, neutral particles are used to converge towards the optimum.

A multi-population based approach can be promising if used with charged particles. The multi-swarm PSO [53] aims at maintaining multiple swarms on different peaks. These swarms are prevented from converging to the same optimum by randomizing the worse of two swarms that come too close. The multi-swarm PSO also replaces the charged particles with quantum particles, whose positions are solely based on a probability function centered around the swarm attractor. This multi-swarm approach is particularly attractive because of its improved adaptability in a more complex multimodal dynamic environment where multiple peaks exist and need to be tracked. Along this line of research, a species-based PSO was also developed to locate and track multiple peaks in a dynamic environment [48, 63], where a speciation algorithm [66] was incorporated into PSO and a local "species seed" was used to provide the local $\mathbf{p_g}$ to particles whose positions are within a user-specified radius of the seed. This encourages swarms to converge toward multiple local optima instead of a single global optimum, hence performing search with mul-

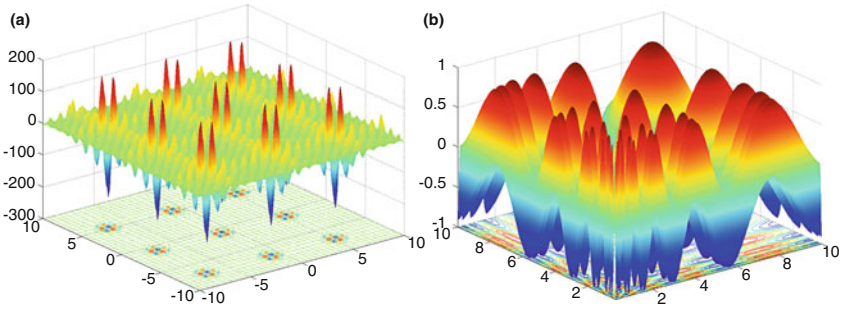

Fig. 11.13 Two multi-modal test functions from the CEC 2013 multi-modal optimization benchmark test function suite [54]: (**a**) Shubert 2D function and (**b**) Vincent 2D function

tiple swarms in parallel. It was also demonstrated in [7] that the quantum particle model can be integrated into the species-based PSO to improve its optima-tracking performance for the moving peaks problem [12].

11.5.3 Multimodal Optimization

The two canonical PSOs, inertia weighted PSO and constricted PSO, were designed for locating a single global solution. The swarm typically converges to one final solution by the end of an optimization run. However, many real-world problems are "multimodal" by nature, that is, multiple satisfactory solutions exist. For such an optimization problem, it may be desirable to locate all global optima and/or some local optima which are considered to be sufficiently good. Figure 11.13 shows the fitness landscapes of two multi-modal test functions, each with multiple global peaks (or solutions). In the early development of genetic algorithms during the 1970s and 1980s, several techniques have been designed specifically for locating multiple optima (global or local), which are commonly referred to as "niching" methods. The most well-known niching methods include *fitness sharing* [33] and *crowding* [27]. Subsequently, other niching methods were also developed, including *restricted tournament selection* [36], *clearing* [66], and *speciation* [52]. Since PSO is also population-based, a niching method can be easily incorporated into PSO, to promote formation of multiple subpopulations within a swarm, allowing multiple optima to be found in the search space.

One early PSO niching model was based on a "stretching method" proposed by Parsopoulos and Vrahatis [64], where a potentially good solution is isolated once it is found. Then, the fitness landscape is "stretched" to keep other particles away from this area of the search space. The isolated particle is checked to see if it is a global optimum, and if it is below the desired threshold, a small population is generated around this particle to allow a finer search in this area. The main swarm continues its search in the rest of the search space for other potential global optima. Another early PSO niching method NichePSO was proposed by Brits et al. [14]. It uses multiple subswarms produced from a main swarm to locate multiple optimal solutions. Subswarms can merge together, or absorb particles from the main swarm. NichePSO monitors the fitness of a particle by tracking its variance over a number of iterations. If there is little change in a particle's fitness over a number of iterations, a subswarm is created with the particle's closest neighbour.

A speciation-based PSO (SPSO) model based on the notion of species was developed in [48]. Here, the definition of species depends on a parameter r_s, which denotes the radius measured in Euclidean distance from the center of a species to its boundary. The center of a species, the so-called species seed, is always the best-fit individual in the species. All particles that fall within distance r_s from the species seed are classified as the same species. A procedure for determining species seeds can be applied at each iteration step. As a result, different species seeds are identified for multiple species, with each seed adopted as the $\mathbf{p_g}$ (similar to a neighbourhood best in an *lbest* PSO) of a different species. In SPSO, a niche radius must be specified in order to define the size of a niche (or species). Since this knowledge may not always be available *a priori*, it may be difficult to apply this algorithm to some real-world problems. To tackle this problem, population statistics or a time-based convergence measure could also be used for adaptively determining the niching parameters during a run [2].

A simple ring neighbourhood topology could be also used for designing a PSO niching method [49]. This PSO niching method (which belongs to the class of *lbest* PSOs) makes use of the inherent characteristics of PSO and does not require prespecification of the niching parameters, hence may offer advantages over previous methods. The ring topology-based niching PSO algorithm makes use of its individual particles' local memories (i.e., the memory-swarm) to form a stable network retaining the best positions found so far, while still allowing these particles to explore the search space more broadly. Given a reasonably large population uniformly distributed in the search space, the ring topology-based niching PSO is able to form stable niches across different local neighbourhoods, eventually locating multiple global/local optima. Of course, the neighbourhood is not necessarily defined only in the topological space. For example, LIPS (Locally Informed PSO) induces a niching effect by using information on the nearest neighbours to each particle's personal best, as measured in the decision space [68].

It is noteworthy here that the intrinsic properties of PSO can be harnessed to design highly competitive niching algorithms. In particular, local memory and slow communication topology seem to be two key components for constructing a competent PSO niching method. Further information on PSO niching methods can be found in [50].

11.6 PSO Theoretical Works

Since PSO was first introduced by Kennedy and Eberhart [45], several studies have been carried out on understanding the convergence properties of PSO. Although particles in isolation and the update rules are simple, the dynamics of the whole swarm of multiple interacting particles can be rather complex. A direct analysis of the convergence behavior of a swarm would be a very challenging task. As a result, many of these works focused on studying the convergence behavior of a simplified PSO system.

Kennedy [41] provided the first analysis of a simplified particle behavior, where particle trajectories for a range of variable choices were given. Ozcan and Mohan [61] showed that the behaviour of one particle in a one-dimensional PSO system, with its $\mathbf{p_i}$, $\mathbf{p_g}$, φ_1 and φ_2 kept constant, follows the path of a sinusoidal wave, where the amplitude and frequency of the wave are randomly generated.

A formal theoretical analysis of the convergence properties of a simplified PSO was provided by Clerc and Kennedy [25], by assuming that the system consists of only one particle, which is one-dimensional, with the best positions being stagnant, and deterministic. The PSO was represented as a dynamic system in state-space form. By simplifying the PSO to a deterministic dynamic system (i.e., removing all its stochastic components), its convergence can be shown based on the eigenvalues of the state transition matrix. If the eigenvalue is less than 1, the particle will converge to equilibrium. Through this study, Clerc and Kennedy were able to derive a general PSO model which employs a constriction coefficient (see Eq. (11.5)).

The original PSO and the inertia weighted PSO can be treated as special cases of this general PSO model. This study also led to suggestions of PSO parameter settings that would guarantee convergence. A similar work was also carried out by van den Bergh [77] where regions of the parameter space that guarantee convergence are identified. The conditions for convergence derived from both studies [25, 77] are: $w < 1$ and $w > \frac{1}{2}(c_1 + c_2) - 1$. In another work by van den Bergh and Engelbrecht [79], the above analysis was generalized by including the inertia weight w. A more formal convergence proof of particles was provided using this representation. Furthermore, the particle trajectory was examined with a relaxed assumption that allowed stochastic values for $\boldsymbol{\varphi_1}$ and $\boldsymbol{\varphi_2}$. They demonstrated that a particle can exhibit a combination of convergent and divergent behaviors with different probabilities when different values of $\boldsymbol{\varphi_1}$ and $\boldsymbol{\varphi_2}$ are used.

In another important theoretical work, Poli [67] suggested a method to build Markov chain models of stochastic optimizers and approximate them on continuous problems to any degree of precision. By using discretization, it allows an easy computation of the transition matrix and represents more precisely the behaviour of PSO at each iteration. Poli [67] was able to overcome the limitations of previous theoretical PSO studies and model the bare-bones PSO without any simplification, that is, the stochastic elements as well as the dynamics of a population of particles are included in the model.

It is worth noting that many analyses are still based on a stagnation assumption, i.e., the best positions are not supposed to move. Although one recent work does not make this assumption [11], the best positions in this case move according to a probabilistic distribution whose expectation is known, which is not more realistic than stagnation (it depends on the problem and may not even exist, e.g., when Lévy flights are used). Hence, although of theoretical interest, this approach cannot yet derive better parameter guidelines than the previous work. Nevertheless, along with a theoretical analysis, recent works also suggest a useful empirical approach [19].

11.7 Other SI Applications

There are many SI applications apart from just optimization. Below we provide two such examples.

11.7.1 Swarm Robotics

Swarm robotics is a new emerging SI research area concerned with the design, control and coordination of multi-robot systems, especially when the number of robots is large. The focus is on the physical embodiment of SI individuals and their interactions with each others and with the environment in a realistic setting. A more formal definition of swarm robotics is provided below [72]:

"Swarm robotics is the study of how a large number of relatively simple physically embodied agents can be designed such that a desired collective behavior emerges from the local interactions among agents and between the agents and the environment."

A swarm robotic system should exhibit the following three functional properties observed in nature: *robustness, flexibility*, and *scalability*. Robustness refers to the system still being able to function despite disturbances from the environment or some individuals being malfunctional; flexibility means that individuals of the swarm can coordinate their behaviours to tackle various tasks; finally scalability means that the swarm is able to operate under different group sizes and with a large number of individuals.

One successful demonstration of the above characteristics of a swarm robotic system is provided in a *swarm-bot* study [35], where a swarm-bot consists of multiple mobile robots capable of self-assembling into task-oriented teams to accomplish tasks that individual robots would not be able to achieve independently. These team-oriented tasks include "crossing a hole", "object transport", and "navigation over a hill" [35]. Readers are referred to [73] for further information on swarm robotics and many application examples.

11.7.2 Swarm Intelligence in Data Mining

Apart from being used as optimization methods, SI techniques can be also used for typical data mining tasks. Data instances in the feature space can be checked and sorted by swarms so that similar data instances are grouped into suitable clusters.

Popular SI techniques such as ACO (Ant Colony Optimization) and PSO have been extensively studied for their capabilities to do data mining tasks. A survey by Martens et al. [58] shows that ACO has been used for both supervised learning such as classification tasks as well as unsupervised learning such as clustering. One most notable example is AntMiner [62], which is an ant-colony-based data miner capable of extracting classification rules from data. In AntMiner, a directed graph is constructed to allow each variable to have multiple paths, each leading to one node associated with one possible value for that variable. Multiple variables in sequence form the entire directed graph. Ants build their paths by sequencing the variables, and by doing so they implicitly construct a rule. Compared with GA-based approaches, rules produced by AntMiner are simpler and can be better understood than other machine learning methods such as neural networks or support vector machines.

11.8 Conclusion

This chapter provides an introduction to Swarm Intelligence (SI), a research paradigm that has shown tremendous growth and popularity in the past decade. SI techniques have been successfully applied to many application domains. In particular, many SI techniques have been developed for solving optimization problems which are challenging for conventional computational and mathematical techniques. Two representative examples are Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO). This chapter focused on the canonical PSOs, their variants, and their common application areas. There is a growing list of new real-world applications based on SI techniques. Nevertheless, SI is still a relatively young field as compared with classic Artificial Intelligence techniques. There is still a huge potential for developing better and more efficient SI algorithms. Many open research questions still remain. Clerc's PSO book [23] provides many pointers to hot research topics on PSO. Blum and Merkle's edited book [9] provides some interesting examples of SI applications. Readers are encouraged to look at the references listed in the bibliography section to gain more in-depth understanding of this fast growing research field.

Acknowledgements This chapter is a further extension to an early EOLSS online article by the first author (Li, X. "Swarm intelligence" Computational Intelligence (6.44.40-50 UNESCO Encyclopedia of Life Support Systems), EOLSS Publishers, Oxford, UK, Vol. II, pp. 87–112, 2015), with new sections and references included to reflect the more recent developments on this topic. The authors would also like to thank Prof. Jean-Yves Potvin for his valuable feedback, which has substantially improved the quality of this chapter.

References

1. G. Beni, J. Wang, Swarm intelligence in cellular robotic systems, in *Robots and Biological Systems: Towards a New Bionics?* ed. by P. Dario, G. Sandini, P. Aebischer (Springer, Berlin, 1993), pp. 703–712
2. S. Bird, X. Li, Adaptively choosing niching parameters in a PSO, in *Proceedings of Genetic and Evolutionary Computation Conference*, July 2006, ed. by M. Cattolico (ACM Press, New York, 2006), pp. 3–10
3. T.M. Blackwell, P. Bentley, Dynamic search with charged swarms, in *Proceedings of Workshop on Evolutionary Algorithms Dynamic Optimization Problems* (2002), pp. 19–26
4. T.M. Blackwell, P.J. Bentley, Improvised music with swarms, in *Proceedings of Congress on Evolutionary Computation*, ed. by D.B. Fogel, M.A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, M. Shackleton (IEEE Press, Piscataway, 2002), pp. 1462–1467
5. T.M. Blackwell, J. Branke, Multi-swarm optimization in dynamic environments, in *Applications of Evolutionary Computing, LNCS 3005* (Springer, Berlin, 2004), pp. 489–500
6. T.M. Blackwell, J. Branke, Multi-swarms, exclusion and anti-convergence in dynamic environments. IEEE Trans. Evol. Comput. **10**(4), 459–472 (2006)
7. T.M. Blackwell, J. Branke, X. Li, Particle swarms for dynamic optimization problems, in *Swarm Intelligence: Introduction and Applications*, ed. by C. Blum, D.D. Merkle (Springer, Berlin, 2008), pp. 193–217

8. C. Blum, X. Li, Swarm intelligence in optimization, in *Swarm Intelligence: Introduction and Applications*, ed. by C. Blum, D. Merkle (Springer, Berlin, 2008), pp. 43–85
9. C. Blum, D. Merkle, *Swarm Intelligence: Introduction and Applications*. Natural Computing Series (Springer, Berlin, 2008)
10. E. Bonabeau, M. Dorigo, G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems* (Oxford University Press, New York, 1999)
11. M.R. Bonyadi, Z. Michalewicz, Stability analysis of the particle swarm optimization without stagnation assumption. IEEE Trans. Evol. Comput. **20**(5), 814–819 (2016)
12. J. Branke, *Evolutionary Optimization in Dynamic Environments* (Kluwer Academic, Norwell, 2002)
13. D. Bratton, J. Kennedy, Defining a standard for particle swarm optimization, in *IEEE Swarm Intelligence Symposium* (June 2007), pp. 120–127
14. R. Brits, A.P. Engelbrecht, F. van den Bergh, A niching particle swarm optimizer, in *Proceedings of 4th Asia-Pacific Conference on Simulated Evolution and Learning* (2002), pp. 692–696
15. A. Carlisle, G. Dozier, Adapting particle swarm optimization to dynamic environments, in *Proceedings of International Conference on Artificial Intelligence*, Las Vegas, NV (2000), pp. 429–434
16. A. Carlisle, G. Dozier, Tracking changing extrema with adaptive particle swarm optimizer, in *Proceedings of World Automation Congress*, Orlando, FL (2002), pp. 265–270
17. R. Carrese, X. Li, Preference-based multiobjective particle swarm optimization for airfoil design, in *Springer Handbook of Computational Intelligence*, ed. by J. Kacprzyk, W. Pedrycz (Springer, Berlin, 2015), pp. 1311–1331
18. R. Carrese, A. Sobester, H. Winarto, X. Li, Swarm heuristic for identifying preferred solutions in surrogate-based multi-objective engineering design. Am. Inst. Aeronaut. Astronaut. J. **49**(7), 1437–1449 (2011)
19. C.W. Cleghorn, Particle Swarm Optimization: Empirical and Theoretical Stability Analysis, Ph.D. thesis, University of Pretoria, 2017
20. M. Clerc, Standard particle swarm optimisation. 15 pages (2012)
21. M. Clerc, Discrete particle swarm optimization, illustrated by the traveling salesman problem, in *New Optimization Techniques in Engineering* (Springer, Heidelberg, 2004), pp. 219–239
22. M. Clerc, Confinements and biases in particle swarm optimisation, Technical report, Open archive HAL (2006). http://hal.archives-ouvertes.fr/, ref. hal-00122799
23. M. Clerc, *Particle Swarm Optimization* (ISTE Ltd, Washington, DC, 2006)
24. M. Clerc, *Guided Randomness in Optimization* (ISTE (International Scientific and Technical Encyclopedia)/Wiley, Washington, DC/Hoboken, 2015)
25. M. Clerc, J. Kennedy, The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. Evol. Comput. **6**(1), 58–73 (2002)
26. C.A.C. Coello, M. Salazar Lechuga, MOPSO: a proposal for multiple objective particle swarm optimization, in *Proceedings of Congress on Evolutionary Computation*, Piscataway, NJ, May 2002, vol. 2, pp. 1051–1056
27. K.A. De Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Ph.D. thesis, University of Michigan, 1975
28. K. Deb, A. Pratap, S. Agrawal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
29. M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents. Trans. Syst. Man Cybern. B **26**(1), 29–41 (1996)
30. R.C. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in *Proceedings of IEEE International Conference Evolutionary Computation* (2000), pp. 84–88
31. R.C. Eberhart, Y. Shi, Tracking and optimizing dynamic systems with particle swarms, in *Proceedings of Congress on Evolutionary Computation* (IEEE Press, 2001), pp. 94–100
32. J.E. Fieldsend, S. Singh, A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence, in *Proceedings of U.K. Workshop on Computational Intelligence*, Birmingham, September 2002, pp. 37–44

33. D.E. Goldberg, J. Richardson, Genetic algorithms with sharing for multimodal function optimization, in *Proceedings of Second International Conference on Genetic Algorithms*, ed. by J.J. Grefenstette, pp. 41–49 (1987)
34. E.F.G. Goldbarg, G.R. De Souza, M.C. Goldbarg, Particle swarm for the traveling salesman problem, in *Evolutionary Computation in Combinatorial Optimization: Proceedings of the 6th European Conference, EvoCOP 2006*, ed. by J. Gottlieb, G. Raidl, R. Günther. LNCS, vol. 3906 (Springer, Berlin, 2006), pp. 99–110
35. R. Groß, M. Bonani, F. Mondada, M. Dorigo, Autonomous self-assembly in swarm-bots. IEEE Trans. Robot. **22**(6), 1115–1130 (2006)
36. G.R. Harik, Finding multimodal solutions using restricted tournament selection, in *Proceedings of Sixth International Conference on Genetic Algorithms*, ed. by L. Eshelman (Morgan Kaufmann, San Francisco, 1995), pp. 24–31
37. S. Helwig, R. Wanka, Particle swarm optimization in high-dimensional bounded search spaces, in *Proceedings of IEEE Swarm Intelligence Symposium*, April 2007 (IEEE Press, Honolulu, 2007), pp. 198–205
38. N. Higashi, H. Iba, Particle swarm optimization with Gaussian mutation, in *Proceedings of IEEE Swarm Intelligence Symposium* (2003), pp. 72–79
39. X. Hu, R.C. Eberhart, Adaptive particle swarm optimisation: detection and response to dynamic systems, in *Proceedings of Congress on Evolutionary Computation* (2002), pp. 1666–1670
40. S. Janson, M. Middendorf, A hierarchical particle swarm optimizer and its adaptive variant. IEEE Trans. Syst. Man Cybern. B **35**(6), 1272–1282 (2005)
41. J. Kennedy, The behaviour of particle, in *Proceedings of 7th Annual Conference Evolutionary Programming*, San Diego, CA (1998), pp. 581–589
42. J. Kennedy, Bare bones particle swarms, in *Proceedings of IEEE Swarm Intelligence Symposium*, Indianapolis, IN (2003), pp. 80–87
43. J. Kennedy, In search of the essential particle swarm, in *Proceedings of IEEE Congress on Evolutionary Computation* (IEEE Press, 2006), pp. 6158–6165
44. J. Kennedy, Swarm intelligence, in *Handbook of Nature-Inspired and Innovative Computing: Integrating Classical Models with Emerging Technologies*, ed. by A.Y. Zomaya (Springer, Boston, 2006), pp. 187–219
45. J. Kennedy, R.C. Eberhart, Particle swarm optimization, in *Proceedings of IEEE International Conference on Neural Networks*, vol. 4 (IEEE Press, Piscataway, 1995), pp. 1942–1948
46. J. Kennedy, R.C. Eberhart, Y. Shi, *Swarm Intelligence* (Morgan Kaufmann, San Francisco, 2001)
47. X. Li, A non-dominated sorting particle swarm optimizer for multiobjective optimization, in *Proceedings of Genetic and Evolutionary Computation Conference, Part I*, ed. by Erick Cantú-Paz et al. LNCS, vol. 2723 (Springer, Berlin, 2003), pp. 37–48
48. X. Li, Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization, in *Proceedings of Genetic and Evolutionary Computation Conference*, ed. by K. Deb. LNCS, vol. 3102 (2004), pp. 105–116
49. X. Li, Niching without niching parameters: particle swarm optimization using a ring topology. IEEE Trans. Evol. Comput. **14**(1), 150–169 (2010)
50. X. Li, Developing niching algorithms in particle swarm optimization, in *Handbook of Swarm Intelligence* ed. by B. Panigrahi, Y. Shi, M.-H. Lim. Adaptation, Learning, and Optimization, vol. 8 (Springer, Berlin, 2011), pp. 67–88
51. X. Li, K.H. Dam, Comparing particle swarms for tracking extrema in dynamic environments, in *Proceedings of Congress on Evolutionary Computation* (2003), pp. 1772–1779
52. J.P. Li, M.E. Balazs, G.T. Parks, P.J. Clarkson, A species conserving genetic algorithm for multimodal function optimization. Evol. Comput. **10**(3), 207–234 (2002)
53. X. Li, J. Branke, T. Blackwell, Particle swarm with speciation and adaptation in a dynamic environment, in *Proceedings of Genetic and Evolutionary Computation Conference*, ed. by M. Cattolico (ACM Press, New York, 2006), pp. 51–58

54. X. Li, A. Engelbrecht, M.G. Epitropakis, Benchmark functions for CEC'2013 special session and competition on niching methods for multimodal function optimization, Technical report, Evolutionary Computation and Machine Learning Group, RMIT University, 2013
55. J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans. Evol. Comput. **10**(3), 281–295 (2006)
56. M. Lovbjerg, T. Krink, Extending particle swarm optimizers with self-organized criticality, in *Proceedings of Congress on Evolutionary Computation* (IEEE Press, 2002), pp. 1588–1593
57. A. Mah, S.I. Hossain, S. Akter, A comparative study of prominent particle swarm optimization based methods to solve traveling salesman problem. Int. J. Swarm Intell. Evol. Comput. **5**(3), 1–10 (2016)
58. D. Martens, B. Baesens, T. Fawcett, Editorial survey: swarm intelligence for data mining. Mach. Learn. **82**(1), 1–42 (2011)
59. R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better. IEEE Trans. Evol. Comput. **8**(3), 204–210 (2004)
60. J. Moore, R. Chapman, *Application of Particle Swarm to Multiobjective Optimization* (Department of Computer Science and Software Engineering, Auburn University, 1999)
61. E. Ozcan, C.K. Mohan, Analysis of a simple particle swarm optimization system, in *Intelligent Engineering Systems through Artificial Neural Networks* (1998), pp. 253–258
62. R.S. Parpinelli, H.S. Lopes, A.A. Freitas, Data mining with an ant colony optimization algorithm. IEEE Trans. Evol. Comput. **6**(4), 321–332 (2002)
63. D. Parrott, X. Li, Locating and tracking multiple dynamic optima by a particle swarm model using speciation. IEEE Trans. Evol. Comput. **10**(4), 440–458 (2006)
64. K. Parsopoulos, M. Vrahatis, Modification of the particle swarm optimizer for locating all the global minima, in *Artificial Neural Networks and Genetic Algorithms*, ed. by V. Kurkova, N. Steele, R. Neruda, M. Karny (Springer, Berlin, 2001), pp. 324–327
65. K. Parsopoulos, M. Vrahatis, Particle swarm optimization method in multiobjective problems, in *Proceedings of ACM Symposium on Applied Computing*, Madrid (ACM Press, New York, 2002), pp. 603–607
66. A. Pétrowski, A clearing procedure as a niching method for genetic algorithms, in *Proceedings of 3rd IEEE International Conference on Evolutionary Computation* (1996), pp. 798–803
67. R. Poli, Mean and variance of the sampling distribution of particle swarm optimizers during stagnation. IEEE Trans. Evol. Comput. **13**(4), 712–721 (2009)
68. B.Y. Qu, P.N. Suganthan, S. Das, A distance-based locally informed particle swarm model for multimodal optimization. IEEE Trans. Evol. Comput. **17**(3), 387–402 (2013)
69. M. Reyes-Sierra, C.A.C. Coello, Multi-objective particle swarm optimizers: a survey of the state-of-the-art. Int. J. Comput. Intell. Res. **2**(3), 287–308 (2006)
70. T. Richer, T. Blackwell, The L*é*vy particle swarm, in *Proceedings of Congress on Evolutionary Computation* (2006), pp. 808– 815
71. J. Riget, J. Vesterstroem, A diversity-guided particle swarm optimizer - the ARPSO, Technical Report 2002-02, Department of Computer Science, University of Aarhus, 2002
72. E. Şahin, Swarm robotics: from sources of inspiration to domains of application, in *Swarm Robotics: SAB 2004 International Workshop (Revised Selected Papers)*, ed. by E. Şahin, W.M. Spears (Springer, Berlin, 2005), pp. 10–20
73. E. Şahin, S. Girgin, L. Bayindir, A.E. Turgut, Swarm robotics, in *Swarm Intelligence: Introduction and Applications*, ed. by C. Blum, D. Merkle (Springer, Berlin, 2008), pp. 87–100
74. R. Salomon, Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions - a survey of some theoretical and practical aspects of genetic algorithms. Biosystems **39**(3), 263–278 (1996)
75. W.M. Spears, D.T. Green, D.F. Spears, Biases in particle swarm optimization. Int. J. Swarm. Intell. Res. **1**(2), 34–57 (2010)
76. P.N. Suganthan, Particle swarm optimiser with neighbourhood operator, in *Congress on Evolutionary Computation (CEC 1999)*, Washington (1999), pp. 1958–1962
77. F. van den Bergh, Analysis of Particle Swarm Optimizers, Ph.D. thesis, Department of Computer Science, University of Pretoria, Pretoria, 2002

78. F. van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization. IEEE Trans. Evol. Comput. **8**(3), 225–239 (2004)
79. F. van den Bergh, A.P. Engelbrecht, A study of particle swarm optimization particle trajectories. Inform. Sci. **176**, 937–971 (2006)
80. K. Veeramachaneni, T. Peram, C. Mohan, L. Osadciw, Optimization using particle swarm with near neighbor interactions, in *Proceedings of Genetic and Evolutionary Computation Conference*, Chicago, IL (2003), pp. 110 – 121
81. M. Wachowiak, R. Smolikova, Y. Zheng, J. Zurada, A. Elmaghraby, An approach to multimodal biomedical image registration utilizing particle swarm optimization. IEEE Trans. Evol. Comput. **8**(3), 289–301 (2004)
82. U.K. Wickramasinghe, X. Li, Using a distance metric to guide PSO algorithms for many-objective optimization, in *Proceedings of Genetic and Evolutionary Computation Conference* (ACM Press, New York, 2009), pp. 667–674
83. H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, Y. Nakanishi, A particle swarm optimization for reactive power and voltage control considering voltage security assessment. IEEE Trans. Power Syst. **15**(4), 1232–1239 (2001)
84. M. Zambrano-Bigiarini, M. Clerc, R. Rojas, Standard particle swarm optimisation 2011 at CEC-2013: a baseline for future PSO improvements, in *Proceedings of Congress on Evolutionary Computation* (2013), pp. 2337–2344