Swarm Intelligence: Concepts, Models and Applications

Technical Report 2012-585

Hazem Ahmed
Janice Glasgow

School of Computing
Queen's University
Kingston, Ontario, Canada K7L3N6
{hazem, janice}@cs.queensu.ca

February 2012

Report Index

1. Introduction

A swarm is a large number of homogenous, simple agents interacting locally among themselves, and their environment, with no central control to allow a global interesting behaviour to emerge. Swarm-based algorithms have recently emerged as a family of nature-inspired, population-based algorithms that are capable of producing low cost, fast, and robust solutions to several complex problems [1][2]. Swarm Intelligence (SI) can therefore be defined as a relatively new branch of Artificial Intelligence that is used to model the collective behaviour of social swarms in nature, such as ant colonies, honey bees, and bird flocks. Although these agents (insects or swarm individuals) are relatively unsophisticated with limited capabilities on their own, they are interacting together with certain behavioural patterns to cooperatively achieve tasks necessary for their survival. The social interactions among swarm individuals can be either direct or indirect [3]. Examples of direct interaction are through visual or audio contact, such as the waggle dance of honey bees. Indirect interaction occurs when one individual changes the environment and the other individuals respond to the new environment, such as the pheromone trails of ants that they deposit on their way to search for food sources. This indirect type of interaction is referred to as stigmergy, which essentially means communication through the environment [4]. The area of research presented in this depth paper focuses on Swarm Intelligence. More specifically, this paper discusses two of the most popular models of swarm intelligence inspired by ants' stigmergic behaviour and birds' flocking behaviour.

In the past decades, biologists and natural scientists have been studying the behaviours of social insects because of the amazing efficiency of these natural swarm systems. In the late-80s, computer scientists proposed the scientific insights of these natural swarm systems to the field of Artificial Intelligence. In 1989, the expression "Swarm Intelligence" was first introduced by G. Beni and J. Wang in the global optimization framework as a set of algorithms for controlling robotic swarm [5]. In 1991, Ant Colony Optimization (ACO) [6][7][8] was introduced by M. Dorigo and colleagues as a novel nature-inspired metaheuristic for the solution of hard combinatorial optimization (CO) problems. In 1995, particle swarm optimization was introduced by J. Kennedy et al. [9][10], and was first intended for simulating the bird flocking social behaviour. By the late-90s, these two most popular swarm intelligence algorithms started to go beyond a pure scientific interest and to enter the realm of real-world applications. It is perhaps worth mentioning here that a number of years later, exactly in 2005, Artificial Bee Colony Algorithm was proposed by D. Karabago as a new member of the family of swarm intelligence algorithms [11][12].

Since the computational modeling of swarms was proposed, there has been a steady increase in the number of research papers reporting the successful application of

Swarm Intelligence algorithms in several optimization tasks and research problems. Swarm Intelligence principles have been successfully applied in a variety of problem domains including function optimization problems, finding optimal routes, scheduling, structural optimization, and image and data analysis [13][14]. Computational modeling of swarms has been further applied to a wide-range of diverse domains, including machine learning [15], bioinformatics and medical informatics [16], dynamical systems and operations research [17]; they have been even applied in finance and business [18].

The remainder of this paper is organized as follows: The next section presents an overview of two natural swarm systems (ant colonies and bird flocks), and also discusses and evaluates the two most popular swarm intelligence algorithms inspired by these natural swarms, namely, artificial ant colony optimization and particle swarm optimization. It further compares them with two of the most popular machine learning algorithms: Artificial Neural Networks and Genetic Algorithms. Then, a summary of the wide-range applications of swarm intelligence algorithms is presented in many different problem domains. The last section summarizes the advantages and limitations of swarm intelligence and provides some concluding remarks on the paper and open questions of the field.

2. Swarm Intelligence (SI) Models

Swarm intelligence models are referred to as computational models inspired by natural swarm systems. To date, several swarm intelligence models based on different natural swarm systems have been proposed in the literature, and successfully applied in many real-life applications. Examples of swarm intelligence models are: Ant Colony Optimization [29], Particle Swarm Optimization [9], Artificial Bee Colony [11], Bacterial Foraging [19], Cat Swarm Optimization [20], Artificial Immune System [21], and Glowworm Swarm Optimization [22]. In this paper, we will primarily focus on two of the most popular swarm intelligences models, namely, Ant Colony Optimization and Particle Swarm Optimization.

2.1 Ant Colony Optimization (ACO) Model

The first example of a successful swarm intelligence model is Ant Colony Optimization (ACO), which was introduced by M. Dorigo et al. [6][7][8], and has been originally used to solve discrete optimization problems in the late 1980s. ACO draws inspiration from the social behaviour of ant colonies. It is a natural observation that a group of *'almost blind'* ants can jointly figure out the shortest route between their food and their nest without any visual information. The following section presents some details about ants in nature, and shows how these relatively unsophisticated insects can cooperatively interact together to perform complex tasks necessary for their survival.

2.1.1 Ants in Nature

Since tens of millions of years ago, ants have survived different environments, climates and ages that dinosaurs, for example, did not. The secret of the remarkable ecological success of ants can be explained by a single word: sociality [23]. Ants have demonstrated exceptional social organization in several ways: They are inclined to live in organized societies made up of individuals that cooperate, communicate, and divide daily tasks. Ants have impressive abilities in finding their way, building their nests, and locating food supplies. They are not only efficient, but hard-working and thrifty creatures that can adapt to different ecosystems and survive harsh weather conditions.

2.1.1.1 Ants Stigmergic behaviour

Ants, like many other social insects, communicate with each other using volatile chemical substances known as pheromones, whose direction and intensity can be perceived with their long, mobile antennae [24]. The term "pheromone" was first introduced by P. Karlson and M. Lüscher in 1959, based on the Greek word pherein (means to transport) and hormone (means to stimulate) [25]. There are different types of pheromones used by social insects. One example of pheromone types is alarm pheromone

that crushed ants produce as an alert to nearby ants to fight or escape dangerous predators and to protect their colony [26]. Another important type of pheromone is food trail pheromone. Unlike flies, most ants live on the ground and make use of the soil surface to leave pheromone trails, which can be followed by other ants on their way to search for food sources. Ants that happened to pick the shortest route to food will be the fastest to return to the nest, and will reinforce this shortest route by depositing food trail pheromone on their way back to the nest. This route will gradually attract other ants to follow, and as more ants follow the route, it becomes more attractive to other ants as shown in Figure 1. This autocatalytic or positive feedback process is an example of a self-organizing behaviour of ants in which the probability of an ant's choosing a route increases as the count of ants that already passed by that route increases.

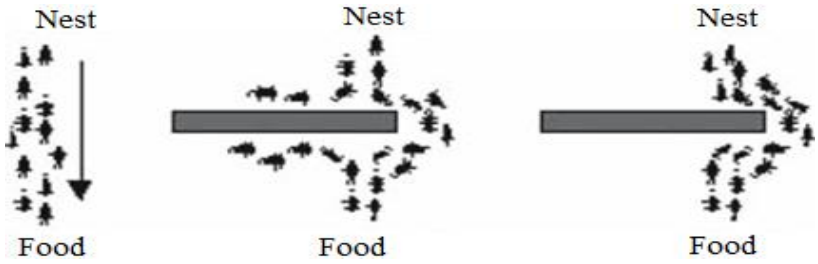

Figure 1: Ants' stigmergic behaviour in finding the shortest route between food and nest [49].

When the food source is exhausted, no new food pheromone trails are marked by returning ants and the volatile pheromone scent slowly evaporates. This negative feedback behaviour helps ants deal with changes in their environment. For instance, when an already established path to a food source is blocked by an obstacle, the ants leave the path to explore new routes. Such trail-laying, trail-following behaviour is called stigmergy (interaction through the environment), and can be considered as an indirect type of communication in which ants change the environment (soil surface) and the other ants detect and respond to the new environment. Stigmergy provides a general mechanism that relates individual (local) and colony-level (global) behaviours: individual behaviour modifies the environment (trail-laying), which in turn modifies the behaviour of other individuals (trail-following) [27].

2.1.1.2 The Double Bridge Experiment

The pheromone trail-laying and trail-following behaviour of ants has been studied in controlled experiments by several researchers. One simple, yet brilliant experiment is referred to as the double bridge experiment, which was designed and run by Goss, Deneubourg and colleagues in the late 1980s [28]. The experiment was simply made of a double bridge connecting a nest of ants and a food source as shown in Figure 2(a). Goss et al. considered different versions of the experimental setup over multiple experiment runs. In one version, the longer branch of the double bridge was twice as long as the short one and both branches are presented from the beginning of the experiment as

shown in Figure 2(a)(i). It was noted in this version that most ant traffic (80-100%) was eventually concentrated on the short branch in more than 90% of the experiment runs as shown in Figure 2(b)(i). Initially, ants left the nest to explore the environment; once they arrived at a decision point, they have to choose one of the two branches. Because the two unmarked branches initially looked identical to the ants (on individual-level behaviour), they were chosen randomly. However, quite surprising at first, the ants (on colony-level behaviour) appeared intelligent enough to eventually choose the shorter branch. This is because the lucky ants that happened to choose the short branch are the first to reach the food and to start their return to the nest. On their return way to the nest, these ants will be biased to pick the short branch over again (now probabilistically and not randomly), because of the higher level of pheromone they already left on the short branch. Returning ants will deposit pheromones once more on the short branch, which causes a faster accumulation of the pheromone trails on the short branch as opposed to the lower-level of pheromone on the not-yet-completed long branch. This stimulates more ants to choose the short branch until eventually it will be adopted by the majority of the ant colony. This explains the positive feedback process of ants, which is based on this simple, self-reinforcement rule: the more number of ants on a branch determines a greater amount of

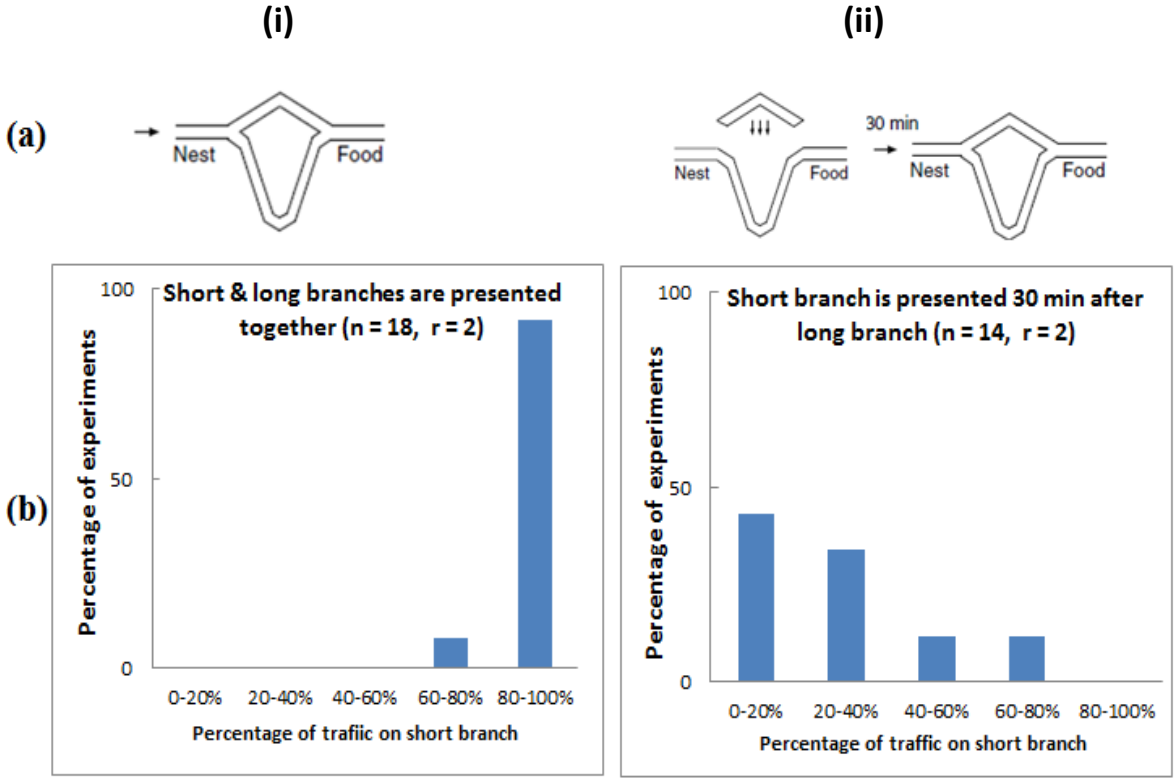

Figure 2: (a) In the first version of the experimental setup on Left (i), short and long branches are presented from the beginning of the experiment. In the second version of the experimental setup on right (ii), the short branch is presented to the colony 30 minutes after the long branch. **(b)** Distribution of the percentage of ants that selected the shorter branch over *n* experiments (r is the length ratio between the two branches). In both versions, the long branch was twice as long as the short branch. Adapted from Goss et al. [28].

pheromone, which influences even more ants to choose this branch [23]. Although the short branch dominated most ant traffic in an impressive *path-exploitation* behaviour of ants, it can be observed, however, from Figure 2(b)(i) that there is still a small percentage of ant traffic that took the longer branch. This may be interpreted as a type of *path-exploration* behaviour of ants [29].

In another version of the experimental setup, initially only the long branch was presented to the colony, and then when a stable pheromone trail has formed on the initially only available long branch, the short branch was offered after 30 minutes, as shown in Figure 2(a)(ii). It is worth mentioning in this version that the longer branch was kept twice as long as the later-offered short branch. This version is deigned to examine what happens when the ant colony is offered, after convergence, a new better (i.e., shorter) path between the nest and the food. It was observed that the short branch was not frequently selected (e.g., only 0-20% of ant traffic took the newly-offered short branch in almost 50% of the experiment runs), and thus the colony largely remained trapped on the initially only-offered long branch as shown in Figure 2(b)(ii). The fact that the great majority of ants continued to choose the long branch can be explained by two reasons: The high pheromone concentration on the long branch and the slow evaporation of pheromone. Firstly, the high-level pheromone concentration of the already established trail on the long branch (compared to the zero-level pheromone-trail concentration on the short branch) led to an autocatalytic behaviour that continued to reinforce the long branch, even after a shorter one is offered. Secondly, the very slow rate of pheromone evaporation did not allow the ant colony to forget the suboptimal path to which they initially converged, preventing the new and shorter path to be discovered and learned [29]. In fact, the pheromone trails of most ant species were usually observed to be persistent for a long time-scale, ranging from at least several hours up to several months (depending on the ant species, the colony size, weather conditions, etc.) [27].

One of the lessons that can be learned from this experiment is that the pheromone evaporation rate is a key parameter in the convergence process, because it controls the trade-off between *path-exploration* of new (and hopefully better) paths and *path-exploitation* of the already established path. Therefore, in the field of artificial ant colony optimization, it is a common practice to set the pheromone evaporation to a sufficiently short time-scale [28]. This allows artificial ant colonies to favour the forgetting of errors (or bad choices) done in the past to allow a continuous improvement of the learned problem [29]. It also helps artificial ant colonies to avoid being trapped on a suboptimal solution and to reduce the risk of possibly stucking in local optima – one of the major concerns of optimization problems. In fact, the pheromone evaporation rate is an interesting example where there is a clear difference between real and artificial ants. The next section discusses the other differences between real and artificial ants, and illustrates the general framework used to move from a natural phenomenon to an artificial system.

2.1.1.3 Real Ants vs. Artificial Ants

Understanding a natural phenomenon and designing a nature-inspired algorithm are two related, yet different tasks. Understanding a natural phenomenon is constrained by observations and experiments, while designing a nature-inspired algorithm is only limited by one's imagination and available technology. Although the underlying principles of ant colony optimization metaheuristic are inspired by the social behaviour of ant colonies, some characteristics of artificial ants do not have to be identically the same as real ants. Table 1 summarizes the main differences between artificial ants and real ants. The artificial ant colony optimization metaheuristic just models the natural ant behaviour. Modeling serves as an interface between understanding nature and designing artificial systems. In other words, one starts from the observed natural phenomenon, tries to make a nature-inspired model of it, and then design an artificial system after exploring the model without constraints [27]. Figure 3 illustrates the framework that is generally used to move from a natural phenomenon to a nature-inspired algorithm. It is worth emphasizing that "memory" is the key difference between real and artificial ants; real ants have no memory, while artificial ants are offered a limited form of memory. The use of memory helps artificial ants to implement a number of useful behaviours that allow them to efficiently build solutions for more complex optimization problems than the simple double bridge experiment. One of such useful behaviours is that artificial ants evaluate the quality of the solutions generated, and use the solution quality in determining

Criteria	Real Ants	Artificial Ants
Pheromone Depositing Behaviour	Pheromone is deposited both ways while ants are moving (i.e. on their forward and return ways).	Pheromone is often deposited only on the return way after a candidate solution is constructed and evaluated.
Pheromone Updating Amount	The pheromone trail on a path is updated, in some ant species, with a pheromone amount that depends on the quantity and quality of the food [31].	Once an ant has constructed a path, the pheromone trail of that path is updated on its return way with an amount that is inversely proportional to the path length stored in its memory.
Memory Capabilities	Real ants have no memory capabilities.	Artificial ants store the paths they walked onto in their memory to be used in retracing the return path. They also use its length in determining the quantity of pheromone to deposit on their return way.
Return Path Mechanism	Real ants use the pheromone deposited on their forward path to retrace their return way when they head back to their nest	Since no pheromone is deposited on the forward path, artificial ants use the stored paths from their memory to retrace their return way.
Pheromone Evaporation Behaviour	Pheromone evaporates too slowly making it less significant for the convergence.	Pheromone evaporates exponentially making it more significant for the convergence.
Ecological Constraints	Exist, such as predation or competition with other colonies and the colony's level of protection.	Ecological constraints do not exist in the artificial/virtual world.

Table 1: Differences between Real Ants and Artificial Ants

the quantity of pheromone to deposit. That is why pheromone is deposited only on the return way after a full path (or, a candidate solution) is constructed and evaluated in terms of the path length (or, generally, the solution cost).

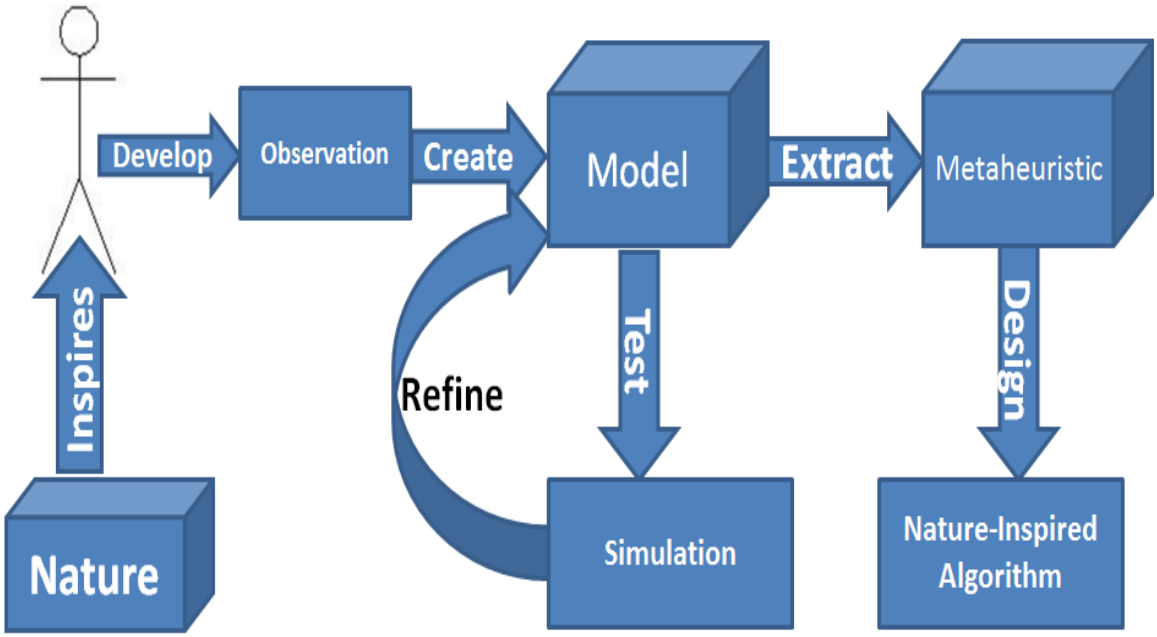

Figure 3: An illustration to the general framework used to move from a natural phenomenon to a nature-inspired algorithm. First, nature inspires humans to develop an observation of a particular natural phenomenon. Next, they create a model and test it using mathematical simulations, which help to refine the original model. Then, the refined model will be used to extract a metaheuristic that can be used as a basis to finally design and tune a nature-inspired algorithm.

2.1.2 Ant Colony Optimization Metaheuristic

ACO is based on pheromone laying/pheromone following behaviour of real ants that helps find the shortest route between their nest and a food source. ACO has been used to solve many optimization problems such as sequential ordering [32], scheduling [33], assembly line balancing [34], probabilistic Traveling Salesman Problem (TSP) [35], DNA sequencing [36], 2D-HP protein folding [37], and protein–ligand docking [38]. The main idea is to model the problem to be solved as a search for an optimal path in a weighted graph, called *construction graph*, and to use artificial ants to search for quality paths. A construction graph is a graph on which artificial ants iteratively deposit pheromone trails to help choose the graph nodes of quality paths that correspond to solution components. The behaviour of artificial ants simulates the behaviour of real ones in several ways: (i) artificial ants deposit pheromone trails on the nodes of quality paths to reinforce the most promising solution components of the construction graph, (ii) artificial ants construct solutions by moving through the

construction graph and choose their path with respect to probabilities, which depend on the pheromone trails previously deposited, and (iii) artificial pheromone trails decrease sufficiently quickly at each iteration simulating the slowly-evaporative pheromone trail phenomena observed in real ants [39]. A key point in the development of any ACO algorithm is to decide the fitness function based on which the components of a problem's construction graph will be rewarded with a high-level pheromone trail, and to determine how ants will exploit these promising components when constructing new solutions. The fitness function of ACO is often implicitly formulated as cost minimization of solution components, i.e., the goal of artificial ants is to walk on the construction graph and select the nodes that minimize the overall cost of the solution path.

Algorithm 1: Basic flow of ACO (adapted from [29])

1. Represent the solution space by a construction graph.
2. Set ACO parameters and initialize pheromone trails
3. Generate ant solutions from each ant's walk on the construction graph mediated by pheromone trails.
4. Update pheromone intensities.
5. Go to step 3, and repeat until convergence or termination conditions are met.

As shown in the basic flow of ACO above, the objective of ACO's third step is to construct ant solutions (i.e., find the quality paths on the problem's construction graph) by stochastically moving through neighbour nodes of the graph. Ants are driven by a probability rule to sequentially choose the solution components that make use of pheromone trail intensities and heuristic information. The solution of each ant is constructed when all solution components are selected by that ant (i.e., when the ant has completed a full tour/path on the construction graph). Once an ant has constructed a solution, or while the solution is being constructed, the ant evaluates the full (or partial) solution to be used by the ACO's next step (the pheromone updating step) in determining how much pheromone to deposit. The probability rule (*equation* 1) is called *Random-Proportional Action Choice* rule (or *State Transition* rule). It guides ant movement through a stochastic local decision policy that essentially depends on both pheromone information and heuristic information [40].

$$P_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}(t)]^{\alpha} * [\eta_{il}]^{\beta}}, & j \in \mathcal{N}_i^k \\ 0, & j \notin \mathcal{N}_i^k \end{cases} \quad (1)$$

Where:

- $P_{ij}^{k}(t)$ is the probability of the k^{th} ant to move from node i to node j at the t^{th} iteration/time step.
- $\mathcal{N}_i^k$ is the set of nodes in the neighborhood of the k^{th} ant in the i^{th} node.

- $P_{ij}^{k}(t) = 0, \forall\, j \notin \mathcal{N}_i^k$ means the ants are not allowed to move to any node not in their neighborhood. The neighborhood definition is problem-specific, for example, in the Traveling Salesman Problem (as discussed later in *Section 2.1.2.1*) the neighborhood is defined by the adjacent cities in the allowed list (the allowed list contains all unvisited cities), while in the Image Segmentation problem, the neighborhood can be defined as the 8-connected pixels surrounding each pixel on a two-dimensional square lattice.
- $[\tau_{ij}(t)]^{\alpha}$ is the pheromone amount on the arc connecting node i and node j, weighted by α (an application-depend constant). $\tau(t)$ is the pheromone information, or trail intensity value, that encodes a long-term memory about the whole ant search process. It is updated by all ants after each iteration *t* (sometimes, however, in more recent ACO versions it is updated by only some ants – the best one(s) that constructed the iteration-best or best-so-far solution).
- $[\eta_{ij}]^{\beta}$ is the heuristic value of the arc connecting node i and node j, weighted by β(an application-depend constant). η is the heuristic information, or path visibility, that represents *a priori information* about the problem instance definition, or *run-time information* provided by a different source other than ants. The heuristic value η_{ij} is usually a non-increasing function in the moving cost from node i to node j, and it often does not change during algorithm execution unless the moving cost is not static.
- α and β are weight parameters that control the relative importance of the pheromone versus heuristic information.
    - A high value for α means that pheromone information is very important; thus, ants are strongly biased to choose nodes previously chosen by other ants. This potentially leads to a *stagnation* situation in which all the ants would eventually follow the same path (usually suboptimal) and construct the same tour [29].
    - A low value of α makes the algorithm very similar to a stochastic multi-*greedy* algorithm with m starting points, as there is m number of ants that are initially randomly distributed over the construction graph.
    - When $\alpha = 0$, the ACO performs a typical stochastic greedy search strategy in which the next node (problem state) is selected only on the basis of its distance (cost) from the current node/state. As a result, the node with the minimum cost will be always favoured regardless of how many other ants have visited it, and how much its pheromone intensity is [29].

- When β = 0, the pheromone information is only used to guide the search process, which would reflect the way that ants do in real world (real ants do not use any heuristic information in their search process) [41].

The objective of ACO's fourth step is to update pheromone trails. At the very beginning, the pheromone trails of *all* arcs on the construction graph are initialized to a small constant value (τ_o). Then after a tour (or, a solution path) is constructed, the pheromone trails are updated in two ways, as shown in *equations 2 and 3*. Firstly, the pheromone trails of *all* arcs are decreased according to an evaporation rate (ρ) that allows ants to forget the suboptimal paths to which they previously converged. Pheromone evaporation rate is usually set to be sufficiently fast in order to favour the *exploration* of new areas of the search space, and avoid a premature convergence of the algorithm toward a local optimum. Secondly, the pheromone trail values of the *visited* arcs are increased with amounts inversely proportional to the cost of their tours (or, in other words, directly proportional to their tour quality). The pheromone depositing procedure implements a useful form of *exploitation* of quality paths by increasing their probability of being used again by future ants. The quality paths would include the solution components that were either used by many ants in the past, or that were used by at least one ant and which produced a high quality solution [40][42].

$$\tau_{ij(t+1)} \leftarrow (1-\rho) * \tau_{ij(t)} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t), \qquad \forall\, i,j \in A,\ 0 \le \rho < 1 \quad (2)$$

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} Q/C^{k}(t) & , if\ arc(i,j) \in T^{k}(t) \\ 0 & , otherwise \end{cases} \quad (3)$$

Where: Q is an application-specific constant, m is the number of ants, A represents *all* arcs of the problem's construction graph, $C^k(t)$ is the overall cost function of tour $T^k(t)$ constructed by the k^{th} ant at the t^{th} iteration, and $T^k(t)$ is the set of all arcs visited by ant k at the iteration t. Other variations of ACO, however, restrict pheromone depositing to the arcs of the best tour T^{best} only.

2.1.2.1 ACO Example: Traveling Salesman Problem and Ant System

The ACO metaheuristic is a general-purpose algorithmic framework on which many heuristic algorithms are based and applied to different optimization problems with a relatively few problem-specific modifications. Ant System (AS) was the first proposed ant-based example of ACO metaheuristic in the literature [8]. Illustrating AS as an example of an ACO algorithm in solving a particular optimization problem best explains how the ACO metaheuristic operates. The Traveling Salesman Problem (TSP) is

a well-studied combinatorial (discrete) optimization problem [45][46][47] that was first applied to the original AS in the early 1990s, and it has later often been used as a benchmark to test new ideas and algorithmic modifications [48]. The set of feasible solutions of combinatorial optimization problems is discrete, that is, each variable has a finite number of values. In TSP, for example, the goal is to find the shortest possible tour from the salesman's home city to a finite number of customer cities with only one constraint that each city must be visited just once before finally returning to the starting home city. That is why ants at each construction step are enforced to choose the next city from an *allowed* list that contains all unvisited cities. The TSP can be represented by a complete weighted graph *G = (V,E)*, where V is a finite set of cities (graph vertices) and E is the set of weighted edges fully connecting the vertices. Each edge has a weight d_{ij} representing the distance between cities i and j. The reason why TSP is the intuitive example first applied to AS is that the TSP problem is readily modeled as a weighted construction graph required by ACO metaheuristic to operate.

In AS, each edge is initialized by the same initial pheromone value τ_0 and each ant is initially put on a randomly chosen start city, making the number of ants *m* equals the number of cities *n* (or, in other words, the ant colony size = |V|). Each ant *k* traverses the construction graph and makes a probabilistic decision to move from city *i* to *j* according to a transition probability $P_{ij}^{k}(t)$ given by *equation 4*. While building the solutions, each ant iteratively stores the solution components (the graph vertices or cities selected from each ant step) in its memory until all cities have been visited. A construction step typically starts with one city in each ant's memory and terminates after each ant completes a tour of all cities.

$$P_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} * \left(\frac{1}{d_{ij}}\right)^{\beta}}{\sum_{l \in allowed_k} [\tau_{il}(t)]^{\alpha} * \left(\frac{1}{d_{il}}\right)^{\beta}}, & j \in allowed_k \\ 0, & otherwise \end{cases} \qquad (4)$$

Where: the pheromone trail $[\tau_{ij}(t)]^{\alpha}$ refers to the *desirability* of visiting city j directly after i at the construction step t, the heuristic information is inversely proportional to d_{ij} (the distance between cities i and j), and the $allowed_k$ list is a set of feasible neighbourhood cities yet to be visited by the k^{th} ant.

Next, the pheromone trails are updated, as discussed earlier, in two ways using the following equations provided in *5 and 6*:

$$\tau_{ij(t+1)} \leftarrow (1-\rho) * \tau_{ij(t)} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t), \qquad \forall\, i,j \in A \qquad (5)$$

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} {}^{1}/_{L^{k}(t)} & , if\ arc(i,j) \in T^{k}(t) \\ 0 & , \quad otherwise \end{cases} \qquad (6)$$

Where: $L^k(t)$ is the *overall* length of the tour $T^k(t)$ constructed by the ant k. It should be noted here that the pheromone deposit value, $\Delta\tau_{ij}^{k}(t)$, between cities i and j is set to be inversely proportional to the overall length of the complete tour $\left({}^{1}/_{L^{k}(t)}\right)$, and not merely the distance of the arc connecting the city i and the city j $\left({}^{1}/_{d_{ij}^{k}(t)}\right)$. That is, if the arc distance between city i and city j is relatively small but the distance of the complete tour $T^k(t)$ in which this arc was used is rather big, then the pheromone on this arc should NOT receive a big reinforcement and thus the pheromone deposit value should be small.

The values of the aforementioned parameters have a strong influence on the algorithm performance and the convergence behaviour. For example, If α is set to 0, the algorithm will always choose the closest city and behaves like a greedy search algorithm. Also, if the initial pheromone value τ_0 is ignored or set to 0, the search is quickly biased by the pheromone deposited during the first tours of '*less-experienced*' ants, which often does not direct the convergence process to optimal solutions. On the other hand, if the initial pheromone τ_0 is set to a too high value, the search is strongly biased by τ_0 (instead of $(\Delta\tau(t)_{ij}^{k})$ for many iterations until pheromone evaporation reduces enough pheromone values, and pheromone deposited by ants can actually start to influence the search process [29]. Typical parameter settings for TSP are: $m = n$ (i.e., number of ants = number of cities), $\alpha = 1$, β=2 to 5, ρ=0.5, and $\tau_0 = 10^{-6}$ [41].

2.1.2.2 ACO Variations: Ant System and its Extensions

Many different variations of the original AS [8] have been proposed in the literature, such as, Elitist AS [49], Ant-Q [50], Max-Min AS [51], Rank-Based AS [52], Ant Colony System [53], and Hyper-cube AS [54]. One of the principal differences between AS and its extensions lies in the pheromone updating procedure; whether a local pheromone update is required after each construction step, or the offline pheromone updating procedure performed at the end of the construction process is sufficient, whether the pheromone update should be done by all ants, or by the best one(s) that have constructed the best or best-so-far tour(s), among other differences. Discussing the details of ACO variations is out of the scope of this paper, but interested readers may refer to this [29][30] for more details about the differences between AS and its extensions. It should be noted that while ACO was originally proposed to solve combinatorial (discrete) optimization problems, some versions of ACO algorithms have

been recently proposed to handle not only continuous optimization problems [55][56], but also mixed optimization problems with both continuous and discrete variables [57].

2.1.2.3 ACO Discussion

A relatively close paradigm to ACO metaheuristic would be the artificial neural network (ANN), since both can be considered as a type of a *connectionist* system in which individual units (artificial ants of ACO or artificial neurons of ANN) are connected to each other according to a certain pattern [41]. Both ACO and ANN are quite similar in some aspects, for example: (i) The knowledge resulted from the learning process is numerically embedded in both of them (either in the weights of the network connections in case of ANN, or in the pheromone concentrations on the construction graph in case of the ACO metaheuristic). (ii) Both ACO and ANN are based on the same principle: reinforcement of portions of solutions that belong to good solutions either by adding more pheromone amounts in case of ACO or increasing network weights in case of ANN. (iii) in ACO the pathways to solutions are usually not predefined but emergent, and likewise the pathways to solutions in most ANNs are hidden in its black-box nature.

That being said, ACO differs from ANN in the following ways: (i) the individual units of ACO have a mobility feature, unlike ANN's individual units (neurons) do not have a mobility feature, but they should be preconfigured with a structure that does not change in run time. (ii) The dynamic nature of ACO's connectivity helps to continuously adapt to changes in real time, making ACO more applicable to dynamic problems such as urban transportation systems [58] and adaptive routing in telecommunication networks [59]. (iii) The two main learning approaches of ANN training (supervised and unsupervised learning[1]) do not get feedback from the environment, whereas ACO primarily depends on the feedback from the environment, which is used as a medium of coordination and indirect communication among ants (stigmergy).

Despite the advantages of ACO, it does have some limitations (as many other optimization algorithms) that do not let it to always work well [27]. For example, ACO does not work well when a large number of edges on the construction graph are equally likely to be part of good paths. This happens when many edges have similar cost and therefore similar probability of being selected as portions of good paths (e.g., a TSP problem whose cities are *uniformly* randomly distributed with a relatively equal distance from one another). Since ACO's objective is to reinforce all edges on the problem's construction graph that belong to good solutions/paths, a large number of edges will receive a relatively equal high amount of pheromone and will be equally likely selected. In this case, the original ACO would not perform well as it would take longer time to differentiate between such many good paths in an effort to eventually converge/select one of them [41].

[1] It is worth mentioning, however, that a third learning approach for ANN training called the "*reinforcement learning*" does use a feedback response from the environment [67].

2.2 Particle Swarm Optimization (PSO) Model

The second example of a successful swarm intelligence model is Particle Swarm Optimization (PSO), which was introduced by Russell Eberhart, an electrical engineer, and James Kennedy, a social psychologist, in 1995 [9][10]. PSO was originally used to solve non-linear continuous optimization problems, but more recently it has been used in many practical, real-life application problems. For example, PSO has been successfully applied to track dynamic systems [60], evolve weights and structure of neural networks [61], analyze human tremor [62], register 3D-to-3D biomedical image [63], control reactive power and voltage [64], even learning to play games [65] and music composition [66]. PSO draws inspiration from the sociological behaviour associated with bird flocking. It is a natural observation that birds can fly in large groups with no collision for extended long distances, making use of their effort to maintain an optimum distance between themselves and their neighbours. This section presents some details about birds in nature and overviews their capabilities, as well as their sociological flocking behaviour.

2.2.1 Birds in Nature

Vision is considered as the most important sense for flock organization [83]. The eyes of most birds are on both sides of their heads, allowing them to see objects on each side at the same time. The larger size of birds' eyes relative to other animal groups is one reason why birds have one of the most highly developed senses of vision in the animal kingdom [68]. As a result of such large sizes of birds' eyes, as well as the way their heads and eyes are arranged, most species of birds have a wide field of view [74]. For example, *Pigeons* can see 300 degrees without turning their head, and *American Woodcocks* have, amazingly, the full 360-degree field of view [75]. Birds are generally attracted by food; they have impressive abilities in flocking synchronously for food searching and long-distance migration. Birds also have efficient social interaction that enables them to be capable of: (i) flying without collision even while often changing direction suddenly, (ii) scattering and quickly regrouping when reacting to external threats, and (iii) avoiding predators [9].

2.2.1.1 Birds Flocking Behaviour

The emergence of flocking and schooling in groups of interacting agents (such as birds, fish, penguins, etc.) have long intrigued a wide range of scientists from diverse disciplines including animal behaviour, physics, social psychology, social science, and computer science for many decades [69][70][71][72][73]. Bird flocking can be defined as the social collective motion behaviour of a large number of interacting birds with a common group objective. The local interactions among birds (particles) usually emerge the shared motion direction of the swarm, as shown in Figure 4. Such interactions are based on the "*nearest neighbour principle*" where birds follow certain flocking rules to

adjust their motion (i.e., position and velocity) based only on their nearest neighbours, without any central coordination. In 1986, birds flocking behaviour was first simulated on a computer by Craig Reynolds [74]. The pioneering work of Reynolds proposed three simple flocking rules to implement a simulated flocking behaviour of birds: (i) *flock centering* (flock members attempt to stay close to nearby flockmates by flying in a direction that keeps them closer to the centroid of the nearby flockmates), (ii) *collision avoidance* (flock members avoid collisions with nearby flockmates based on their relative position), and (iii) *velocity matching* (flock members attempt to match velocity with nearby flockmates) [74].

Although the underlying rules of flocking behaviour can be considered simple, the flocking is visually complex with an overall motion that looks fluid yet it is made of discrete birds [74]. One should note here that *collision avoidance* rule serves to "establish" the minimum required separation distance, whereas *velocity matching* rule helps to "maintain" such separation distance during flocking; thus, both rules act as a complement to each other. In fact, both rules together ensure that members of a simulated flock are free to fly without running into one another, no matter how many they are. It is worth mentioning that the three aforementioned flocking rules of Reynolds are generally known as *cohesion*, *separation*, and *alignment* rules in the literature [76][77]. For example, according to the animal cognition and animal behaviour research, individuals of animals in nature are frequently observed to be attracted towards other individuals to avoid being isolated and to align themselves with neighbours [78][79]. Reynolds rules are also comparable to the *evaluation*, *comparison*, and *imitation* principles of the Adaptive Culture Model in the Social Cognitive Theory [80].

Figure 4: The flocking behaviour of a group of birds (adapted from [81]).

2.2.1.2 Birds' Physical Movement vs. Humans' Psychological Change

The Social Cognitive Theory, used in psychology, education, and communication, suggests that portions of knowledge acquired by humans can be directly influenced by their neighbours within the context of social interactions and experiences [82]. While being different, the birds flocking behaviour can be mapped to the human social behaviour, since the concept of bird's physical *movement* is generally analogous to the concept of psychological behaviour *change* in humans. But unlike birds, we tend to adjust our ideas, beliefs and attitudes, instead of just adjusting our physical positions, to conform to our social peers. Another obvious distinction between humans and birds in this context lies in the fact that the same attitudes and beliefs can be concurrently held by many individuals without banging with each other, but any two birds must occupy different positions in the 3D space to avoid collision [9]. In other words, birds move through a three-dimensional physical space, avoiding collisions, whereas humans psychologically change in an n-dimensional abstract space, collision-free, in addition to moving through a 3D physical space and avoiding collisions. It is worth emphasizing, however, that although we learn to avoid physical collision by an early age, decades of practice and experience are often required to learn how to efficiently navigate through such an abstract n-dimensional, psychological space [9].

2.2.2 Particle Swarm Optimization Metaheuristic

Particle Swarm Optimization (PSO) is a heuristic optimization technique introduced by Kennedy and Eberhart in 1995 [9][10]. It is inspired by the intelligent, experience-sharing, social flocking behaviour of birds that was first simulated on a computer by Craig Reynolds [74], and further studied by the biologist Frank Heppner [84]. PSO is a population-based search strategy that finds optimal solutions using a set of flying particles with velocities that are dynamically adjusted according to their historical performance, as well as their neighbours in the search space [85]. While ACO solves problems whose search space can be represented as a weighted *construction graph* (refer to *Section* 2.1.2), PSO solves problems whose solutions can be represented as a *set of points* in an n-dimensional solution space. The term "*particles*" refers to population members, which are fundamentally described as the swarm positions in the n-dimensional solution space. Each particle is set into motion through the solution space with a velocity vector representing the particle's speed in each dimension. Each particle has a memory to store its historically best solution (i.e., its best position ever attained in the search space so far, which is also called its *experience*).

The secret of the PSO success lies in the experience-sharing behaviour in which the experience of each particle is continuously communicated to part or the whole swarm, leading the overall swarm motion towards the most promising areas detected so far in the search space [17]. Therefore, the moving particles, at each iteration, evaluate their current position with respect to the problem's fitness function to be optimized, and compare the

current fitness of themselves to their historically best positions, as well as to the other individuals of the swarm (either locally within their neighbourhood as in the local version of the PSO algorithm, or globally throughout the entire swarm as in the global version of the algorithm). Then, each particle updates its *experience* (if the current position is better than its historically best one), and adjusts its velocity to imitate the swarm's global best particle (or, its local superior neighbour, i.e., the one within its neighbourhood whose current position represents a better solution than the particle's current one) by moving closer towards it. Before the end of each iteration of PSO, the index of the swarm's global best particle (or, the local best particle in the neighbourhood) is updated if the most recent update of the position of any particle in the entire swarm (or, within a predetermined neighbourhood topology) happened to be better than the current position of the swarm's global best particle (or, the local best particle in the neighbourhood).

2.2.2.1 The Original PSO Algorithm

The original PSO was designed as a global version of the algorithm [9], that is, in the original PSO algorithm, each particle globally compares its fitness to the entire swarm population and adjusts its velocity towards the swarm's global best particle. There are, however, recent versions of local/topological PSO algorithms, in which the comparison process is locally performed within a predetermined neighbourhood topology [80][86][87]. Unlike the original version of ACO (refer to *Section* 2.1.2), the original PSO is designed to optimize real-value continuous problems, but the PSO algorithm has also been extended to optimize binary or discrete problems [88][89][90]. The original version of the PSO algorithm is essentially described by the following two simple "*velocity*" and "*position*" update equations, shown in 7 and *8* respectively.

$$v_{id(t+1)} = v_{id(t)} + c_1 \boldsymbol{R}_1 (p_{id(t)} - x_{id(t)}) + c_2 \boldsymbol{R}_2 (p_{gd(t)} - x_{id(t)}) \quad (7)$$

$$x_{id(t+1)} = x_{id(t)} + v_{id(t+1)} \quad (8)$$

Where:

- v_{id} represents the rate of the position change (velocity) of the i^{th} particle in the d^{th} dimension, and t denotes the iteration counter.
- x_{id} represents the position of the i^{th} particle in the d^{th} dimension. It is worth noting here that $\mathbf{X}_i$ is referred to as the i^{th} particle itself, or as a vector of its positions in all dimensions of the problem space. The n-dimensional problem space has a number of dimensions that equals to the numbers of variables of the desired fitness function to be optimized.
- p_{id} represents the historically best position of the i^{th} particle in the d^{th} dimension (or, the position giving the best ever fitness value attained by $\mathbf{X}_i$).

- p_{gd} represents the position of the swarm's global best particle ($\mathbf{X}_g$) in the d^{th} dimension (or, the position giving the *global* best fitness value attained by *any* particle among the entire swarm).
- **R**1 and **R**2 are two n-dimensional vectors with random numbers uniformly selected in the range of [0.0, 1.0], which introduce useful randomness for the search strategy. It worth noting that each dimension has its own random number, *r*, because PSO operates on each dimension independently [17].
- c_1 and c_2 are positive constant weighting parameters, also called the *cognitive* and *social* parameters, respectively, which control the relative importance of particle's private experience versus swarm's social experience (or, in other words, it controls the movement of each particle towards its individual versus global best position [91]). It is worth emphasizing that a single weighting parameter, c, called the acceleration constant or the learning factor, was initially used in the original version of PSO, and was typically set to equal 2 in some applications (i.e., it was initially considered that $c_1 = c_2 = c = 2$). But, to better control the search ability, recent versions of PSO are now using different weighting parameters which generally fall in the range of [0,4] with $c_1 + c_2 = 4$ in some typical applications [17]. The values of c_1 and c_2 can remarkably affect the search ability of PSO by biasing the new position of $\mathbf{X}_i$ toward its historically best position (its own private experiences, $\mathbf{P}_i$), or the globally best position (the swarm's overall social experience, $\mathbf{P}_g$):
    - High values of c_1 and c_2 can provide new positions in relatively distant regions of the search space, which often leads to a better global exploration [17], but it may cause the particles to diverge.
    - Small values of c_1 and c_2 limit the movement of the particles, which generally leads to a more refined local search around the best positions achieved [17].
    - When $c_1 > c_2$, the search behaviour will be biased towards particles' historically best experiences.
    - When $c_1 < c_2$, the search behaviour will be biased towards the swarm's globally best experience.

The velocity update *equation* in (7) has three main terms: (i) The first term, $v_{id(t)}$, is sometimes referred to as "*inertia*", "*momentum*" or "*habit*" [17]. It ensures that the velocity of each particle is not changed abruptly, but rather the previous velocity of the particle is taken into consideration [85]. That is why the particles generally tend to continue in the same direction they have been flying, unless there is a really major difference between the particle's current position from one side, and the particle's historically best position or the swarm's globally best position from the other side (which means the particle starts to move in the wrong direction). This term has a particularly

important role for the swarm's globally best particle, $\mathbf{x}_g$, This is because if a particle, $\mathbf{x}_i$, discovers a new position with a better fitness value than the fitness of swarm's globally best particle, then it becomes the global best (i.e., g←i). In this case, its historically best position, p_i, will coincide with both the swarm's global best position, p_g, and its own position vector, $\boldsymbol{x}_i$, in the next iteration (i.e., $p_i = \boldsymbol{x}_i = p_g$) [17]. Therefore, the effect of last two terms in *equation* (7) will be no longer there, since in this special case $p_{id(t)} - x_{id(t)} = p_{ig(t)} - x_{id(t)} = 0, \ \forall\, d$. This will prevent the global best particle to change its velocity (and thus its position), so it will keep staying at its same position for several iterations, as long as there was no way to offer an inertial movement and there has been no new best position discovered by another particle. Alternatively, when the previous velocity term is included in the velocity updating *equation* (7), the global best particle will continue its exploration of the search space using the inertial movement of its previous velocity [17]. (ii) The second term, $\boldsymbol{(\, p_{id(t)} - x_{id(t)}\,)}$, is the "*cognitive*" part of the equation that implements a linear attraction towards the historically best position found so far by each particle [91]. This term represents the private-thinking or the self-learning component from each particle's flying experience [85], and is often referred to as "*local memory*", "*self-knowledge*", "*nostalgia*" or "*remembrance*" [17]. (iii) The third term, $\boldsymbol{(\, p_{gd(t)} - x_{id(t)}\,)}$, is the "*social*" part of the equation that implements a linear attraction towards the globally best position ever found by any particle [91]. This term represents the experience-sharing or the group-learning component from the overall swarm's flying experience [85], and is often referred to as "*cooperation*", "*social knowledge*", "*group knowledge*" or "*shared information*" [17].

According to the aforementioned *equations* (7) and (8), the basic flow of the original PSO algorithm can be described as shown below.

Algorithm 2: Basic flow of PSO (adapted from [85])

1) Initialize the swarm by randomly assigning each particle to an arbitrarily initial velocity and a position in each dimension of the solution space.
2) Evaluate the desired fitness function to be optimized for each particle's position.
3) For each individual particle, update its historically best position so far, $\boldsymbol{P_i}$, if its current position is better than its historically best one.
4) Identify/Update the swarm's globally best particle that has the swarm's best fitness value, and set/reset its index as $\boldsymbol{g}$ and its position at $\boldsymbol{P_g}$.
5) Update the velocities of all the particles using *equation* (7).
6) Move each particle to its new position using *equation* (8).
7) Repeat steps 2–6 until convergence or a stopping criterion is met (e.g., the maximum number of allowed iterations is reached; a sufficiently good fitness value is achieved; or the algorithm has not improved its performance for a number of consecutive iterations).

2.2.2.2 The Refinements and Extensions to the Original PSO

The PSO algorithm showed sufficiently good performance on the simple optimization problems firstly applied to its original and early versions, but some limitations later appeared when PSO was applied to harder optimization problems with large search spaces and multiple local optima [17]. As a result, a number of parameter/methodology refinements are considered in the later versions of PSO in order to (i) prevent what is known as "*swarm explosion*" by limiting the maximum velocity, (ii) facilitate the convergence in harder optimization problems by introducing an inertia weight, and (iii) handle optimization problems with multiple local optima by defining a neighbourhood topology for a local version of PSO [91].

(i) Limiting the Maximum Velocity

In the velocity update equation (7), when $x_{id}(t) << p_{id}(t)$ and $x_{id}(t) << p_{gd}(t)$, the new velocity, $v_{id}(t+1)$, will have a very large *+ve* value, and the algorithm will enforce the i[th] particle's current position to be significantly adjusted forward to become closer to its historically best position and the swarm's global best position. On the other hand, when $x_{id}(t) >> p_{id}(t)$ and $x_{id}(t) >> p_{gd}(t)$, the new velocity, $v_{id}(t+1)$, will have a very large *-ve* value, and the algorithm will enforce the i[th] particle's current position to be significantly adjusted back to its historically best position and the swarm's global best position. It has been observed, however, that too much increase or decrease to the values of particles' velocities has often led to what is known as "*swarm explosion*" in the early versions of PSO. Swarm explosion refers to the uncontrolled increase of the magnitude of particle velocities, $|v_{id}(t+1)|$, which could lead to swarm divergence (especially when the problem's search space is very large) [17]. This issue was addressed by defining a problem-dependent maximum velocity threshold ($v_{max} > 0$) for the velocity magnitude to avoid the particles taking extremely large shifts from their current position, realistically simulating the incremental change of human learning [92], as described below:

$$|v_{id}(t+1)| \leq v_{max}, \qquad i = 1, 2, \ldots, N \text{ (particles)} \quad \text{and} \quad d = 1, 2, \ldots, n \text{ (dimensions)}$$

If, at any iteration, t, the result of the velocity update equation presented at (7) violates the rule above, i.e., $|v_{id}(t+1)| > v_{max}$, then the values of the violating particles' velocities are clamped, as follows:

$$v_{id}(t+1) = \begin{cases} v_{max}, & if\ v_{id}(t+1) > v_{max}, \\ -v_{max}, & if\ v_{id}(t+1) < -v_{max} \end{cases} \qquad (9)$$

The value of the parameter, V_{max}, is important because it remarkably affects the algorithm behaviour. For example, if V_{max} is sufficiently big, the particles fly far past the target region and could discover even better positions than what they originally set out for. This improves the global exploration ability of the algorithm as the particle would be able to take sufficiently large steps to escape from local optima. On the other hand, a small value of V_{max} could cause the particles to be trapped into local optima, and prevent them from discovering better solution areas [85]. If necessary, the value of the maximum velocity could be not only problem-dependent, but also dimension-dependent according to the problem's space dimensions [17]. Nevertheless, in order to ensure uniform velocity throughout all dimensions, Abido [93] has proposed an equation to govern the maximum velocity value, as shown below in *equation* (10):

$$v_{max} = \frac{x_d^{max} - x_d^{min}}{K} \quad (10)$$

Where: x_d^{max} and x_d^{min} are the maximum and minimum position values found so far by the particles in the d^{th} dimension, and *K* is a user-defined parameter that controls the shift intervals (or, the particles' steps in each dimension of the search space), with k = 2 being a common choice (i.e., velocities are clamped to *at most 50%* of the range on each dimension [17]).

(ii) Introducing an Inertia Weight

The inertia weight is introduced to control the global exploration ability of PSO, and provide a balance between the global and local search abilities. It has been observed that PSO produces better results when its global *exploration* ability is *more* favoured in the *early* optimization stages to allow the exploration of as many promising areas of the search space as possible. Then, towards the end of the optimization process, the local *exploitation* ability of the algorithm should be promoted, instead, to allow for a more refined search around the best areas previously *roughly* detected [17]. This is possible by reducing the position shifts (or, the velocity) of the particles in the later search stages. This means the effect of the previous velocity term, $v_{id}(t)$, of *equation* (7) (which is known as "inertia" or "momentum" as discussed in *Section* 2.2.2.1) will gradually fade over PSO iterations for each particle. Therefore, a linearly decreasing inertia weight, ω, multiplied to that previous velocity term was introduced by Shi and Eberhart [94], as shown in *equation* (11). Intuitively, the linearly decreasing inertia weight is initially set to a high value, ω_{hi}, around 1.0 (typically, from 0.9 to 1.2) in order to allow the particles to move freely, and quickly explore the global optimum neighbourhood [17]. Towards the later optimization stages, when the optimal regions are roughly identified, the value of the inertia weight is decreased to a small amount, ω_{low}, around 0.2 (typically, from 0.1 to

0.4) in order to refine the search, and shift the optimization process from an *exploratory* mode to an *exploitative* mode [17][91].

$$v_{id(t+1)} = \boldsymbol{\omega}\, v_{id(t)} + c_1 \boldsymbol{R}_1 (p_{id(t)} - x_{id(t)}) + c_2 \boldsymbol{R}_2 (p_{gd(t)} - x_{id(t)}) \quad (11)$$

$$x_{id(t+1)} = x_{id(t)} + v_{id(t+1)} \quad (12)$$

The rest of the parameters of *equations* (11) and (12) remain the same as for the equations of the original PSO version presented in *equations* (7) and (8). Since the inertia weight is selected such that the effect of $v_{id(t)}$ *gradually* fades during the execution of the algorithm, a *linearly* decreasing scheme for the inertia weight is often utilized [17]. One possible definition of such linearly decreasing scheme can be mathematically described, as shown in *equation* (13).

$$\omega(t) = \omega_{hi} - (\omega_{hi} - \omega_{low}) \frac{t}{T_{max}} \quad (13)$$

Where: t is the iteration counter; ω_{hi} *and* ω_{low} are the desired higher and lower bounds of the inertia weight, respectively; T_{max} is the maximum allowed number of iterations after which the algorithm shall terminate. The definition of that scheme produces a linearly decreasing time-dependent inertia weight with initial value, ω_{hi}, at the first iteration, $t = 0$, and final value, ω_{low}, at the last possible iteration, T_{max} [17]. It is worth noting that the concept of a linearly decreasing value of the inertia weight could be considered quite analogous to the concept of *simulated annealing* [2] that is often used in global optimization problems [95].

(iii) Defining a Neighbourhood Topology

Despite its aforementioned benefits, the introduction of a linearly-decreasing inertia weight has a disadvantage; that is, once the inertia weight is faded, the swarm's exploration ability is almost lost and cannot be recovered [91]. This means no further exploration is possible and the particles can only perform local search around their convergence point, which most likely exists close to the swarm's global best position. The *instant* information-sharing of the swarm's global best position can be attributed to this disadvantage, because each particle always knows and *instantly* shares the global best position at each iteration [17]. If this information, however, is not instantly shared, but rather slowly propagated throughout several *local neighbourhoods* before affecting the entire swarm, the particles' exploration ability will generally be retained longer to explore more areas in the search space, which solves that disadvantage and decreases the

[2] Simulated Annealing (SA), inspired by the annealing in solids [96], simulates the process of material cooling in a heat bath, which is known as the process of physical annealing. SA is a stochastic search technique with good abilities to escape local optima by taking a random walk through the search space at successively lower temperatures, following a Boltzmann distribution.

chance of premature convergence [97]. The main idea of using the concept of local neighbourhood is as follows: the information of the swarm's best particle position is initially shared only to its neighbours and successively to the rest of the entire swarm's particles through their neighbours, allowing the wisdom to gradually emerge instead of trying to impose it. It is worth mentioning, however, that while the aforementioned disadvantage did not particularly appear for simple optimization problems with unimodal or convex fitness functions, it remarkably appeared for high-dimensional, multimodal, and complex optimization problems [17].

Neighbourhoods are either based on randomly assigned indices to the particles, the actual distances of the particles in the search space, or a particular predefined topological neighbourhood structure [17][87][98]. The original global version of PSO can be considered a special case of the local version with its neighbourhood being defined as the entire swarm. The neighbourhood of the global version of PSO can, therefore, be conceptualized as a fully connected network in which each particle has access to the information of all other particles in the swarm (**Figure 5**(a)), as opposed to just its immediate local neighbours in a predefined neighbourhood topology. The two most common local neighbourhood topologies are ring (or, circle) and star (or, wheel) topologies [91]. As for the ring topology, particles are arranged in a ring-like structure in which each particle is directly connected with its *two* immediate neighbours to its right and left (**Figure 5**(b)). Whereas, for the star topology, particles are not *directly* connected to one another; rather they are *all* connected to a selected one particle called the focal point to which all the swarm information is shared and communicated, as shown in **Figure 5** (c). There are also many other regular predefined neighbourhood topologies, such as the pyramid topology and the von Neumann topology. As its name implies, the particles in the pyramid topology are arranged in a pyramid-like structure in which each particle is *directly* connected to its *three* immediate neighbours, as shown in **Figure 5** (d). Whereas, in the Von Neumann structure, particles are arranged in a grid-like structure or a two-dimensional lattice network where each particle is connected to at most *four* of its immediate neighbours (above, below, right and left), as shown in **Figure 5**(e).

The choice of neighbourhood topology has a significant effect on the propagation of the best solution found by the swarm. For example, in the global version of PSO, the propagation of the best solution is very fast, since the global best solution is instantly shared among all the particles [97]. However, in the ring and Von Neumann topologies, on the other hand, the best solution is slowly propagated throughout several *local neighbourhoods* before reaching all particles in the swarm. Kennedy et al. suggested that the global version of PSO converges fast but it may get trapped in local optimum or increase the chance of premature convergence, while the local version results in a larger diversity and increases the chances to find the global optimal solution, although with slower convergence rate [80]. Kennedy and Mendes tested and evaluated PSO performance with all aforementioned regular topologies, shown in **Figure 5,** as well as PSO

performance with *randomly generated* neighbours [87]. In their experiments, with a fixed swarm size of 20 particles, they observed that the best performance occurred in the case of randomly generated neighbours with an average size of *five* neighbouring particles. As for regular shaped topologies, the authors recommended *Von Neumann* topology over all other regular shaped topologies, since it consistently performed better, in their experiments, compared to all other topologies, including the global and the local (e.g., ring) version [87]. It is worth emphasizing, however, that selecting the most efficient neighbourhood topology largely depends on the type of problem. One topology may perform more effectively on specific types of problems; however, it could have a worse performance on other problems [91]. Kennedy believed that regular shaped topologies with fewer connections might perform better on highly multimodal problems, while highly interconnected topologies would be better and faster for unimodal problems [86].

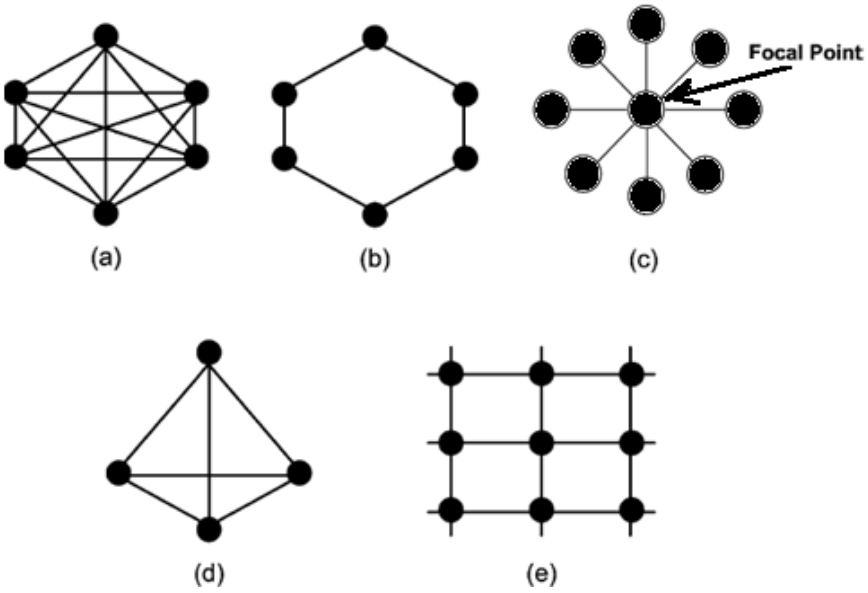

Figure 5: Common regular shaped neighbourhood topologies: (a) the fully connected network topology (PSO's original global version), (b) the ring (circle) topology, (c) the star (wheel) topology, (d) the pyramid topology, and (e) the Von Neumann topology (the generally recommended neighbourhood topology). Adapted from [91].

In addition to the three previously-discussed core refinements to the original PSO [9], several different variations and extensions to the original PSO have been proposed in the literature, such as Constriction Coefficient/Canonical PSO [99][100], Dissipative PSO [101], Stretching PSO [102], Gaussian PSO [103], PSO with Mutation [104], Fully-informed PSO [105], Species-based PSO [106], Self-organizing Hierarchical PSO [107], Cooperative PSO [108] and Comprehensive Learning PSO [109], among many other variations. The details of PSO variations are beyond the scope of this paper, but interested readers may refer to these references [80][85][91] for more information.

2.2.2.3 PSO Discussion

This section summarizes the strengths and limitations of PSO, as well as it highlights the similarities and dissimilarities between PSO, as one of the most competitive emerging computing techniques versus Genetic Algorithms (GA) as an example of evolutionary computing techniques. Lastly, it discusses the advantages of using PSO over GA.

- ***PSO Strengths:***
  - PSO uses memory to store the particle's historically best position and the swarm's global best position, which helps not only each particle to keep track of its own individual experience, but also helps the most superior particle to communicate its social experience to the other particles. This generally directs the convergence to the most promising areas on the search space and accelerates the optimization process towards the optimal solution [80].
  - PSO is not only characterized by its fast convergence behaviour, but also by its simplicity. The core mathematical equations of PSO (namely, *velocity update, position update, and memory update*) are easily calculated. Thus, the implementation of PSO procedure is simple and generally requires just a relatively few lines of code [9].
  - PSO has an inherit potential to adapt to a changing environment, which can expand its ability from just locating optima in static environments to further track them in dynamic environments [111].
- ***PSO Limitations:***
  - The typical PSO problems are those whose solutions can be represented as a set of points in an n-dimensional Cartesian coordinate system, as it would be easy, in such problems, to determine the previous and next positions for each point (i.e., particle). On the other hand, PSO fails to work if the problem representation does not offer a clear way to uniquely define what the next and previous particle positions are to help search in the solution space [112].
  - The original PSO assumes all particles of the entire swarm are completely homogenous, and therefore employs the same value settings of inertia weight, cognitive and social parameters (c_1 and c_2) for the entire swarm. This assumption, however, ignores the internal differences among birds of the same swarm in real life, such as ages, catching skills, flying experiences, and muscles' stretching. It also neglects the relative flying position within the swarm, although it provides an important influence on particles. For example, particles flying in the outer side of the swarm often make more choices than those in the swarm center, and thus should receive more attention [123].
  - The original PSO fails to locate multiple optima, since the idea of the original PSO was to adjust the swarm direction closer to the swarm's global best particle to guide the entire swarm to converge to a single optimum. However, many variations of the original PSO have been proposed in the literature to overcome such a limitation. For example, Li proposed a species-based PSO (SPSO), which divides the swarm into multiple species (groups of particles sharing similar characteristics) and enables them to concurrently search for multiple optima [106].

Although PSO and GA are based on totally different philosophical metaphors (namely, the evolution metaphor for GA, and the bird flocking metaphor for PSO), both PSO and GA share some common features, besides many other different characteristics. The comparison between PSO and GA has generally been popular in the literature [114]-[118], as it highlights what the novelty of the PSO metaphor is and what's new PSO can offer compared to the other metaphorical models. The comparison also demonstrates the fact that the general SI metaphor is not merely giving new names to existing operations, but there are fundamental differences in the core optimization processes/functions between it and other metaphorical models.

Before going to the comparison details, let us first briefly overview the basics of Genetic Algorithms (GA). In the mid-1970s, John Holland was the first to rigorously present the main concepts of GA [119], drawing inspiration from the evolution metaphor of the Darwinian Theory, and following basic genetics principles. GA employ three operators to propagate its population from one generation to another: Selection, Crossover and Mutation. (i) The selection operator mimics the natural selection's principal (*Survival of the Fittest*), in which the most fitted population individuals are selected for future generations over weaker, less-fit individuals. (ii) The crossover operator mimics the reproduction behaviour observed in biological populations. It propagates the good characteristics/chromosomes of the current generation to future ones by allowing fit individuals to produce more offspring than less-fit individuals, which help improve the average fitness of new generations as the algorithm progresses. (iii) The mutation operator promotes the exploration ability of the algorithm by introducing useful diversity in population characteristics, which acts as necessary randomness to reduce the probability of getting tapped into local optima. The details of GA are beyond the scope of this paper, but interested readers may refer to these references [120]-[122] for more information.

- ***PSO Similarities to GA***
  - *Initialization Mechanism*: Both PSO and GA are stochastic population-based algorithms that start with a number of randomly generated individuals/particles.
  - *Fitness Function:* Both PSO and GA use a specific fitness function (that is desired to be optimized) to evaluate the population members (i.e., either individuals' genetic encodings in GA or particles' positions in PSO), and accordingly assign fitness values to them.
  - *Nature-inspired Properties*: Both PSO and GA update their population according to a number of nature-inspired properties. For instance, the velocity update equation in PSO and the arithmetic crossover operator in GA are both nature-inspired properties that can actually be considered quite analogous to each other [113].
  - *Parameter Tuning*: Both GA and PSO have several numerical parameters that remarkably affect the convergence process, and therefore need to be carefully selected. For example, population size, crossover and mutation rates are required to be carefully selected in GA. Also, swarm size, inertia weight, cognitive and social parameters (c_1 and c_2) need to be cleverly decided upon in PSO [114].

- ***PSO Dissimilarities to GA***
  - *Different Conceptual Bases*: The conceptual bases of PSO and GAs are intrinsically different: GAs are based on the intelligence of natural selection, whereas PSO algorithms are based on the intelligent social behaviour of swarms in nature.
  - *Cooperation vs. Competition*: PSO algorithms choose the path of "cooperation", i.e., convergence is driven through learning from cooperative peers/particles, while GAs choose the path of "competition", i.e., the convergence is driven through learning from competitive individuals (following the survival of the fittest principle) [123].
  - *Selection Mechanism*: The objective of the selection mechanism in GA is to apply natural selection's principle (survival of the fittest), in such a way that the best individuals with the highest performance on the optimization problem are selected and individuals with poor performance are discarded. On the other hand, PSO does not explicitly include a selection mechanism for its convergence strategy; rather it relies on each particle's memory of its historically best position and the swarm's global/local best position. It is worth noting that the particle's best position (the individual experience) in PSO largely resembles the parent's role in GA with the distinction that no new individuals in PSO are created, but instead are updated relative to their own individual experience, or for analogy purposes, their own parents [115].
  - *Population Adapting vs. Population Replacement:* In PSO, instead of explicitly using genetic operators like crossover and mutation, each particle adjusts its velocity (and therefore position) according to its own flying experience, as well as the flying experience of its peers, so the changes are driven through learning from peers and not through genetic recombination and mutations [94]. In other words, PSO iteratively uses a velocity update equation through a process of "*adapting*" the *current* population (so, the convergence is performed by attracting the particles to positions with good solutions), while GAs use crossover and mutation operators through a process of "*replacing*" the *previous* population with a new one (resembling the death and birth of successive generations in nature). In contrast, PSO population is more stable, as its particles are not destroyed or created, but rather they are just influenced by the best performance of themselves and their peers [123].
  - *Conscious Mutation vs. Random Mutation*: The position update equation in PSO, which adds the velocity to the current position to generate the new/next position, is quite analogous to the arithmetic mutation operation in GA. However, the "mutation" process in PSO is not randomly performed (as in GA); rather it is guided by particle's own flying experience and the flying experience of its peers. In other words, the position update equation of PSO performs some sort of *conscious* mutation, as opposed to the *random* mutation performed in GA (using a predefined mutation operator and rate) [94].
  - *Memory Capabilities*: Since the original PSO has a built-in memory capability, each particle in PSO benefits from its previous experience. In contrast, individuals in GA do not benefit from their history because the standard GA has no memory [91], plus the population in each iteration of GAs

replace itself, anyway, in a number of generations that are successively destroyed and created.

- *Information Sharing Mechanism*: In GAs, chromosomes *mutually* share their genetic information with each other through a genetic recombination process known as crossover. In PSO, however, only the global/local best particle communicates its position information to other particles in a *one-way* information sharing mechanism [123].
- *Problems Types*: The standard GA is an inherently discrete algorithm, i.e., it encodes its design variables into bits of 0's and 1's, making it generally suitable for discrete/binary problems [116]. In contrast, the original PSO is an inherently continuous algorithm, but it was later modified to handle discrete/binary problems. It has been observed that the binary PSO is generally faster, more robust and performs better than GAs, particularly on high dimensional problems [117][118].

➢ ***PSO Advantages over GA***

- The key advantage of PSO over GA is that it is algorithmically simpler, yet more robust and generally converges faster than GA [97][116]. In fact, the simplicity of PSO allowed scientists from different backgrounds, not necessarily related to computer science or programming skills, to use PSO as an efficient optimization tool to a wide-range of application domains.
- PSO is more able to control convergence than GA. Although manipulating rates of crossover and mutation can have an effect on controlling GA's convergence, such controlling effect is not as significant compared to the level of control that can be achieved in PSO through manipulating its inertia weight [114]. For example. It has been shown that the decrease of inertia weight dramatically increases the swarm's convergence [124].
- Because of the various studies available in the literature to address the parameter selection issue in PSO, the PSO parameters are now more easily selected and more robustly tuned/controlled than GA parameters [94][125].
- PSO has an impressive ability to perform well without having a large swarm size. In fact, it has been observed that PSO with smaller swarm sizes perform comparably to GAs with larger populations [97]. It has also been observed that the PSO performance is not too sensitive to the population size, as long as the population size is not too small. This observation with first suggested by Shi and Eberhart [94], and then verified by Løvberg and Krink [126][127].

Hybrid approaches combining PSO and GA were attempted by Veeramachaneni et al. in 2003 [128]. The main idea of their work is to take the population of one algorithm (when there has been no fitness improvement) and use it as the starting population for the other one, instead of just employing the traditional random initialization mechanism [97]. Two versions were proposed in this study: GA-PSO and PSO-GA. In GA-PSO, the GA population is used to initialize the PSO population, while in PSO-GA, the PSO population is used to initialize the GA population. The study results showed that: **GA & GA-PSO << PSO < PSO-GA**, i.e., PSO-GA was the best-performing version, and it even had a slightly better performance than PSO. Furthermore, both PSO and PSO-GA performed remarkably better than both GA and GA-PSO [97][128].

3. SI Applications

The purpose of this section is to present some real-life problems and applications in which the use of swarm-based optimization algorithms has been successfully made in the literature, but not providing readers with an extensive survey on swarm intelligence applications. More details on SI applications can, however, be found here [1][2][14][16][17][29][80][85][91]. The impressive performance of SI algorithms in discrete and continuous optimization problems has increased the attention of many researchers with different backgrounds to apply SI algorithms into their own research areas. As a result, there has been an almost exponential increase in the number of research papers reporting the successful application of SI-based algorithms in a wide range of domains, including combinatorial optimization problems, function optimization, finding optimal routes, scheduling, structural optimization, image analysis, data mining, machine learning, bioinformatics, medical informatics, dynamical systems, industrial problems, operations research, and even finance and business.

The potential of SI is yet far from being exhausted with many interesting applications still to be explored, especially in bioinformatics. In the past few years, there has been a slow, yet steady increase in the number of research papers that have successfully applied SI algorithms in bioinformatics. This is because several tasks in bioinformatics involve optimization of different criteria (such as, energy, alignment score, overlap strength, etc.), and the various applications of SI algorithms proved them to be efficient, robust and computationally inexpensive optimization techniques, which made their applications in bioinformatics more obvious and appropriate [16].

It is worth noting, however, that SI-based algorithms do not fully show their competitive edge over other optimization techniques on static problems whose characteristics and conditions do not change over time. Nevertheless, they are often more competitive to deterministic approaches[3] in dealing with uncertainty, as well as general-purpose heuristics (e.g., hill climbing and simulated annealing approaches) in dealing with stochastic time-varying problem domains, due to their inherit adaptive capabilities [27].

3.1 ACO Applications

The scientific community has raised remarkable interest in ACO algorithms due to their amazing capabilities. According to a recent (2010) book chapter surveying ACO applications [48], there are hundreds of implementations of the ACO metaheuristic successfully applied to numerous optimization problems in various domains, including famous NP-hard combinatorial optimization problems. While ACO was initially

[3] Deterministic approaches are those with no parameter tuning or randomness involved, such as the least square optimization.

introduced with an application to the TSP as a proof-of-concept/classical application, ACO algorithms have later been successfully applied to a wide-range of optimization problems. This ranges from fundamental combinatorial problems, such as sequential ordering problems, assignment problems, scheduling problems, the maximum clique problem, graph coloring, assembly line balancing and vehicle routing problems, to more recent continuous, multi-objective or dynamic problems in machine learning, data mining, telecommunication networks and bioinformatics [141].

3.1.1 ACO Relevance to Bioinformatics

Despite a number of differences [138], most of the ordering problems in bioinformatics, (e.g., sequence alignment, fragment assembly problem, and gene mapping) are quite similar to the TSP, one of the most classical and famous ordering problem [16]. As previously discussed how TSP can be efficiently solved by the ant systems (*Section 2.1.2.2*), ACO can be (and has actually been) efficiently applied to many of such ordering problems in bioinformatics [141]. Thus, bioinformatics and biomedical fields, in particular, have shown a steady growing interest in ACO [16]. Examples of ACO applications in bioinformatics and biomedical problems include: DNA sequencing by hybridization [36], the 2D and 3D hydrophobic polar protein folding [37], protein–ligand docking [38], constructing phylogenetic trees [139], multiple sequence alignment [142], DNA fragment assembling [143], and the prediction of major histocompatibility complex (MHC) class II binders [144].

3.2 PSO Applications

The first practical application of PSO was in the field of neural networks, in 1995, when PSO was able to train and adjust the weights of a feed-forward multilayer perceptron neural network as effectively as the conventional error back-propagation approach [9]. Since then, a nearly-exponential growing number of PSO applications have been explored in several domains due to their simplicity, efficiency and fast-convergence nature. A comprehensive technical report in [145] has made an extensive review of over ***1,100*** PSO publications. Among those over one thousand PSO publications, the review report considered around 350 papers as proposals for improvements and extensions to the original "*1995-version*" of PSO. Such large proposed number of PSO variations and extensions has made PSO capable of solving several optimization problems ranging from unconstrained, single-objective or static problems to constrained, multi-objective or dynamic problems. The report further considered the remaining ~700 papers as PSO applications, although many of them also introduced different customizations and extensions to PSO method to fit their particular application. Of those ~700 papers, PSO applications have been classified into 26 different categories. The massive number and scope of successful PSO applications fall under a broad domain of research areas, ranging from combinatorial optimization problems to computational intelligence applications, from electrical and electromagnetic applications to signal processing and graphics, from image analysis and robotics to bioinformatics and medical applications.

Among the 26 proposed categories of PSO applications, the "*Image and Video Analysis*" application category was the biggest one with about 9% of the total surveyed PSO applications at the time of the report. Examples of image analysis applications include: IRIS recognition [146], fruit quality grading [147], face detection and recognition [148], image segmentation [149], synthetic aperture radar imaging [150], locating treatment planning landmarks in orthodontic x-ray images [151], image classification [152], inversion of ocean colour reflectance measurements [153], traffic stop-sign detection [154], defect detection [155], image retrieval [156], human detection in infrared imagery [157], image registration [63], pixel classification [158], detection of objects [159], pedestrian detection and tracking [160], texture synthesis [161], microwave imaging [162], scene matching [163], photo time-stamp recognition [164], contrast enhancement [165], 3D recovery with structured beam matrix [166], auto cropping for digital photographs [167], character recognition [168], shape matching [169], image noise cancellation [170]. Examples of video analysis applications include: MPEG optimization [171], motion estimation [172], object tracking [173], body posture tracking [174], and traffic incident detection [175].

3.2.1 PSO Relevance to Bioinformatics

The key challenge of bioinformatics problems lies in the huge amount of their data, and thereby their computational complexity. As a result, many bioinformatics problems do not always need the exact optimal solution; an approximation to the solution is often used instead. Bioinformatics problems, therefore, require optimal or even near-optimal solutions that are computationally inexpensive, and can be produced in a fast and robust means, which PSO algorithms are actually distinguished by. That is why PSO algorithms have been efficiently applied in many bioinformatics problems. Furthermore, the laboratory operations on DNA, for instance, inherently involve errors and uncertainty, which are more tolerable in PSO algorithms compared to deterministic algorithms. Actually, these errors may be considered beneficial in PSO to some extent, as they may introduce useful randomness and contribute to population diversity – a desirable property for PSO convergence [17].

The review report in [145] has surveyed more than 25 different applications in bioinformatics, biomedical and pharmaceutical problems. Applications include: human tremor analysis for the diagnosis of Parkinson's disease [62], inference of gene regulatory networks [176], human movement biomechanics optimization [177], phylogenetic tree reconstruction [178], cancer classification [179], cancer survival prediction [180], DNA motif detection [181], gene clustering [182], identification of transcription factor binding sites in DNA [183], biomarker selection [184], protein structure prediction [185], protein–ligand docking [186], drug design [187], radiotherapy planning [188], analysis of brain MEG data [189], RNA secondary structure determination [190], EEG analysis [191], and biometrics [192].

4. Summary and Concluding Remarks

Because of its many advantages, SI is now being considered as one of the most promising AI techniques with steady growing scientific attention. This can be supported by the increasing produced number of successful SI research output, as well as the rapidly expanded conferences and journals dedicated for Swarm Intelligence [110].

4.1 SI General Advantages

- *Scalability*: SI systems are highly scalable; their impressive abilities are generally maintained when using groups ranging from just sufficiently few individuals up to millions of individuals. In other words, the control mechanisms used in SI systems are not too dependent on swarm size, as long as it is not too small [3].
- *Adaptability*: SI Systems respond well to rapidly changing environments, making use of their inherit auto-configuration and self-organization capabilities. This allows them to autonomously adapt their individuals' behaviour to the external environment dynamically on the run-time, with substantial flexibility [3].
- *Collective Robustness*: SI Systems are robust as they collectively work without central control, and there is no single individual crucial for the swarm to continue to function (due to the redundancy of their individuals). In other words, the fault-tolerance capability of SI systems is remarkably high, since these systems have no single point of failure. A single point of failure is a part of any system that puts the entire system into risk of a complete failure, if it ceased to function [129].
- *Individual Simplicity*: SI systems consist of a number of simple individuals with fairly limited capabilities on their own, yet the simple behavioural rules at the individual level are practically sufficient to cooperatively emerge a sophisticated group behaviour [129].

4.2 SI General Limitations

The potential of swarm intelligence is indeed fast-growing and far-reaching. It offers an alternative, untraditional way of designing complex systems that neither require centralized control nor extensive pre-programming. That being said, SI systems still have some limitations, such as:

- *Time-Critical Applications*: Because the pathways to solutions in SI systems are neither predefined nor pre-programmed, but rather emergent, SI systems are not suitable for time-critical applications that require (i) on-line control of systems, (ii) time critical decisions, and (iii) satisfactory solutions within very restrictive time frames, such as the elevator controller and the nuclear reactor temperature controller. It remains to be useful, however, for non-time critical applications that involve numerous repetitions of the same activity [3].
- *Parameter Tuning:* Tuning the parameters of SI-inspired optimization techniques is one of the general drawbacks of swarm intelligence, like in most stochastic optimization methods, and unlike deterministic optimization methods. In fact, however, since many parameters of SI systems are problem-dependent, they are

often either empirically pre-selected according to the problem characteristics in a trial-and-error manner [130], or even better adaptively adjusted on run time (as in the adaptive ACO [131] and the fuzzy adaptive PSO [132]).

- *Stagnation*: Because of the lack of central coordination, SI systems could suffer from a *stagnation* situation or a premature convergence to a local optimum (e.g., in ACO, stagnation occurs when all the ants eventually follow the same suboptimal path and construct the same tour [29]). This limitation, however, can be controlled by carefully setting algorithm parameters, e.g., the parameter α in ACO (*Section 2.1.2),* or the parameter $\boldsymbol{\omega}$ of PSO (*Section 2.2.2.2 (ii)*). Different variations of ACO and PSO algorithms could further reduce the probability of that limitation (e.g., by explicitly or implicitly limiting the amount of pheromone trials, as proposed in Max-Min AS [51] or Ant Colony Systems [53], as well as by varying inertia weight, $\boldsymbol{\omega}$, exponentially (rather than linearly), as lately proposed in a recent PSO variation called Exponential PSO [133]).

4.3 Comparison between the two discussed SI Models: ACO *vs.* PSO

The objective of this paper was to present the main principles and concepts of Swarm Intelligence, with a particular focus on two of the most popular SI models, namely, ACO (*Section 2.1.2*) and PSO (*Section 2.2.2*). Despite both models are principally similar in their inspirational origin (the intelligence of swarms), and are based on nature-inspired properties, they are fundamentally different in the following aspects.

Criteria	ACO	PSO
Communication Mechanism	ACO uses an indirect communication mechanism among ants, called stigmergy, which means interaction through the environment.	The communication among particles in PSO is rather direct without altering the environment.
Problem Types	ACO was originally used to solve combinatorial (discrete) optimization problems, but it was later modified to adapt continuous problems.	PSO was originally used to solve continuous problems, but it was later modified to adapt binary/ discrete optimization problems.
Problem Representation	ACO's solution space is typically represented as a weighted graph, called construction graph.	PSO's solution space is typically represented as a set of n-dimensional points.
Algorithm Applicability	ACO is commonly more applicable to problems where source and destination are predefined and specific.	PSO is commonly more applicable to problems where previous and next particle positions at each point are clear and uniquely defined.
Algorithm Objective	ACO's objective is generally searching for an optimal path in the construction graph.	PSO's objective is generally finding the location of an optimal point in a Cartesian coordinate system.
Examples of Algorithm Applications	Sequential ordering [32], scheduling [33], assembly line balancing [34], probabilistic TSP [35], DNA sequencing [36], 2D-HP protein folding [37], and protein–ligand docking [38].	Track dynamic systems [60], evolve NN weights [61], analyze human tremor [62], register 3D-to-3D biomedical image [63], control reactive power and voltage [64], and even play games [65].

Table 2: Comparison between ACO and PSO

4.4 Other SI Models

While ACO and PSO are two of the most common examples of optimization techniques inspired by swarm intelligence, there are several other optimization techniques based on SI principles have been proposed in the literature, including Artificial Bee Colony [11], Bacterial Foraging [19], Cat Swarm Optimization [20], Artificial Immune System [21] and Glowworm Swarm Optimization [22], among many others. All these SI models intrinsically share the principal inspirational origin of the intelligence of different swarms in nature, such as swarms of *E. coli* bacteria as in Bacterial Foraging, swarms of cells and molecules as in Artificial Immune System, and the amazing swarms of honey bees as in the Artificial Bee Colony System.

What is amazing about honey bee colonies is that they are very efficient in exploiting the best food sources (in terms of distance and quality) based on a group of forager bees. When a forager bee (recruiter) decides to attract more bee mates to a newly discovered good food source, it returns to the hive and starts performing what is known as the "waggle dance" to communicate spatial and profitability information about the discovered food source, and recruit more honey bees (dancer followers) to exploit it. The language of waggle dance and its orientation patterns were first deciphered by von Frisch in 1967 [137]. The waggle dance consists of a series of *waggle phases*. A waggle phase starts when the recruiter bee vigorously shakes its body from side to side [134]. The time interval between each waggle phase is called a *return phase*, in which the recruiter bee makes an abrupt turn to the left or right before starting another waggle phase. The waggle dance encodes both (a) spatial and (b) profitability information of the target food source to dance followers [140].

(a) As for spatial information, the waggle dance encodes two important pieces of information: (i) the direction and (ii) the distance to the target. (i) Direction information is encoded in the *waggle dance orientation* [140]. During the waggle dance, the recruiter bees amazingly align their body with an angle representing the direction of food location relative to current sun direction. This means food sources located directly in line with the current sun direction are represented by a series of waggle phases oriented to the upward/vertical direction. If food sources, however, are located with an angle to the right or left of the sun, their direction is encoded in the waggle dance orientation by a corresponding angle to the right or left of the upward direction. What is more astounding is that recruiter bees have an internal clock that helps them adjust the angles of their dances relative to the sun directional changes throughout the day, even after they have been in their almost dark hive for extended time [137]. (ii) On the other hand, distance information is encoded in the *waggle phase duration*, i.e., dances for close targets have short waggle phases, while dances for remote targets have long waggle phases. Dance followers need both direction and distance information to reach the target food source, which could be several kilometers away from the hive, as they fly in a three-dimensional space, unlike most ants that just normally walk on the ground searching for nearby food sources.

(b) As for profitability information, it is encoded in the overall *waggle dance duration* and the *return phase duration* (or the time interval between waggle phases). The larger the number of waggle phases (or the longer the overall duration of waggle dance), and the shorter time interval between waggle phases, the more profitable the target food

source [140]. Even more astoundingly, the nervous system of even beginner recruiter bees has been internally calibrated to assess the profitability of food sources based on different factors: (i) the sugar content of their nectar, (ii) their distance from the colony, and (iii) the ease with which nectar (or pollen) can be collected. After recruiter bees assess these factors, they decide on two things: firstly, if the food source worth foraging for (by themselves), and secondly if it worth recruiting more honey bees [135].

The foraging behaviour of a honey bee colony can be summarized as follows: When a forager bee finds a food source, it first returns to the hive and relinquishes its nectar to worker bees to store it in the hive. At that point, the forager bee has three options/decisions to take: (i) it can become a recruiter bee and performs a waggle dance to recruit more bees (the dance followers) to join it in foraging for the food source, if it is worthwhile, (ii) it can remain as a forager bee by just going back to the food source and continue foraging there by itself, if it is not really worth advertising for, or (iii) it can become an uncommitted follower by abandoning the food source when it is completely exhausted – in this case, the uncommitted-follower bee starts to watch for any waggle dances being performed by other recruiter bees and potentially become a dance-follower bee [136]. The details of natural honey bee colonies and the Artificial Bee Colony optimization algorithm, as well as other SI-inspired optimization techniques are beyond the scope of this paper, but interested readers may refer to these references [11][12][19]-[22] for more information.

4.5 Concluding Remarks and Open Questions

In this paper, the main concepts and principles of Swarm Intelligence are presented, with a particular focus on two of the most successful and popular SI-inspired optimization techniques: Ant Colony Optimization and Particle Swarm Optimization. The aim here, in this last section, is suggesting the concluding remarks on the topic, as well as presenting the current open research questions of the field. It was both challenging and interesting to study and research on this topic. On one hand, it was completely new to me, as well as relatively recent and interdisciplinary. On the other hand, it was very interesting to learn how astoundingly-intelligent the social collective behaviour of swarms in nature, and see how amazing is it to uncover some of nature's secrets, as well as realize how knowledge from different disciplines (such as, animal behaviour, physics, social psychology and social sciences) can actually work in harmony together, and practically be used in computer science and beyond.

- **Concluding Remarks:**
  - Nature is a rich inspirational source and there is still much to learn from.
  - We can take advantage of the social collective behaviour of swarms to solve our real-life problems, by observing how these swarms have survived and solved their own challenges in nature.
  - Several simple agents interacting locally among themselves can eventually emerge a sophisticated global behaviour.
  - Different SI-based computational models are fast-growing, as they are generally computationally inexpensive, robust, and simple.
  - SI-based optimization techniques are far-reaching in many domains, and have a wide-range of successful applications on different areas.

- Swarm intelligence is an active field in Artificial Intelligence and Emerging Computing, and its potential is still far from being exhausted, with many studies are exponentially growing and going on.

➢ **Open Questions:**

- Should the individual agents of artificial swarms remain simple? If not, how complex should they be?
- Should the individual agents remain identical or homogenous? If not, how different should they be?
- Should the individual agents have the ability to learn on their own?
- How local should their knowledge of the environment be?
- How to efficiently tune the parameters of SI systems. Despite different studies in the literature had tried to solve this problem [99][131][132] (e.g., by adaptively changing the parameters of SI systems on run time), this is still a current open research question.
- Should SI approaches remain bottom-up, and the pathways to their solutions remain emergent (i.e., not predefined)? If not, is it possible to clearly define the pathways linking between the lower-level individual interactions and the upper-level emergent group behaviour?

References

[1] B. K. Panigrahi, Y. Shi, and M.-H. Lim (eds.): Handbook of Swarm Intelligence. Series: Adaptation, Learning, and Optimization, Vol 7, Springer-Verlag Berlin Heidelberg, 2011. ISBN 978-3-642-17389-9.

[2] C. Blum and D. Merkle (eds.). Swarm Intelligence – Introduction and Applications. Natural Computing. Springer, Berlin, 2008.

[3] M. Belal, J. Gaber, H. El-Sayed, and A. Almojel, Swarm Intelligence, In Handbook of Bioinspired Algorithms and Applications. Series: CRC Computer & Information Science. Vol. 7. Chapman & Hall Eds, 2006. ISBN 1-58488-477-5.

[4] M. Dorigo, E. Bonabeau, and G. Theraulaz, Ant algorithms and stigmergy, Future Gener. Comput. Syst., Vol. 16, No. 8, pp. 851–871, 2000.

[5] G. Beni and J. Wang, Swarm intelligence in cellular robotic systems. In NATO Advanced Workshop on Robots and Biological Systems, Il Ciocco, Tuscany, Italy, 1989.

[6] M. Dorigo, V. Maniezzo, and A. Colorni, Positive feedback as a search strategy, Tech. Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1991.

[7] M. Dorigo, Optimization, learning and natural algorithms (in Italian), Ph.D. Thesis, Dipartimento diElettronica, Politecnico di Milano, Italy, 1992.

[8] A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian. Ant System for Job-shop Scheduling. Belgian Journal of Operations Research, Statistics and Computer Science, 34(1):39-53, 1994.

[9] J. Kennedy and R. C. Eberhart. Particle Swarm Optimization. In Proceedings of IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942–1948, 1995.

[10] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp. 39–43, 1995.

[11] D. Karaboga, An Idea Based On Honey Bee Swarm for Numerical Optimization, Technical Report-TR06,Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[12] D. Karaboga and B. Basturk, A Powerful And Efficient Algorithm For Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm, Journal of Global Optimization, Springer Netherlands, Vol. 39, No. 3, pp: 459-471, 2007.

[13] A. P. Engelbrecht (ed.), Computational Intelligence: An Introduction. John Wiley & Sons, England, 2002.

[14] C. P. Lim, L. C. Jain, and S. Dehuri, Innovations in Swarm Intelligence: Studies in Computational Intelligence, Vol. 248, Springer, 2009.

[15] S. Das, B. K. Panigrahi, and S. S. Pattnaik, Nature-Inspired Algorithms for Multi-objective Optimization, Handbook of Research on Machine Learning Applications and Trends: Algorithms Methods and Techniques, Hershey, New York, Vol. 1, pp. 95–108, 2009.

[16] S. Das, A. Abraham, and A. Konar, Swarm Intelligence Algorithms in Bioinformatics, Studies in Computational Intelligence. Vol. 94, pp. 113–147, 2008.
[17] K. E. Parsopoulos and M N. Vrahatis, Particle Swarm Optimization and Intelligence: Advances and Applications, Information Science Reference, Hershey, Pennsylvania, 2010.
[18] E. Bonabeau, C. Meyer, Swarm Intelligence: A Whole New Way to Think About Business, Harvard Business Review, Vol.79, No.5, pp. 106-114, 2001.
[19] K. M. Passino, Biomimicry of Bacteria Foraging for Distributed Optimization and Control, IEEE Control Systems Magazine, Vol. 22, 52–67, 2002.
[20] S.-C. Chu, P.-W. Tsai and J.-S. Pan, Cat swarm optimization, Proc. of the 9th Pacific Rim International Conference on Artificial Intelligence, LNAI 4099, pp. 854-858, 2006.
[21] M. Bakhouya and J. Gaber, An Immune Inspired-based Optimization Algorithm: Application to the Traveling Salesman Problem, Advanced Modeling and Optimization, Vol. 9, No. 1, pp. 105-116, 2007.
[22] K.N. Krishnanand and D. Ghose, Glowworm swarm optimization for searching higher dimensional spaces. In: C. P. Lim, L. C. Jain, and S. Dehuri (eds.) Innovations in Swarm Intelligence. Springer, Heidelberg, 2009.
[23] L. Keller and E. Gordon, The Lives of Ants, Oxford University Press, Oxford 2009.
[24] D. E. Jackson, F. L. Ratnieks, Communication in ants, Current Biology, Vol. 16, No. 15, pp. R570–R574, 2006.
[25] P. Karlson and M. Lüscher, Pheromones: a new term for a class of biologically active substances. Nature, Vol. 183, pp. 55–56, 1959.
[26] C. Lloyd, The alarm pheromones of social insects: A review, Technical report, Colorado State University, 2003.
[27] E. Bonabeau, M. Dorigo, and G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, Oxford, 1999.
[28] S. Goss, S. Aron, J.-L. Deneubourg, and J. M. Pasteels, Self-Organized Shortcuts in the Argentine Ant, Naturwissenchaften, Vol. 76, pp. 579-581, 1989.
[29] M. Dorigo and T. Stützle, Ant Colony Optimization. MIT Press, Cambridge, 2004. ISBN: 978-0-262-04219-2.
[30] N. Zhao, Z. Wu, Y. Zhao, and T. Quan, Ant colony optimization algorithm with mutation mechanism and its applications. Expert Systems with Applications, Vol. 37, No. 7, pp. 4805-4810, 2010.
[31] R. Beckers, J.-L. Deneubourg, and S. Goss, Modulation of trail laying in the ant Lasius niger (hymenoptera: Formicidae) and its role in the collective selection of a food source, Journal of Insect Behavior, Vol. 6, No. 6, pp. 751–759, 1993.
[32] L. M. Gambardella, M. Dorigo, Ant colony system hybridized with a new local search for the sequential ordering problem. INFORMS J. Comput, Vol. 12, No. 3, pp. 237–255, 2000.
[33] C. Blum, Beam-ACO—Hybridizing ant colony optimization with beam search: An application to open shop scheduling, Comput. Oper. Res., Vol. 32, No. 6, pp. 1565–1591, 2005.

[34] C. Blum, Beam-ACO for simple assembly line balancing. INFORMS J. Comput. Vol. 20, No. 4, pp. 618–627, 2008.
[35] P. Balaprakash, M. Birattari, T. Stützle, Z. Yuan, and M. Dorigo, Estimation-based ant colony optimization algorithms for the probabilistic travelling salesman problem. Swarm Intell., 3(3), 223–242, 2009.
[36] C. Blum, M. Yabar, and M. J. Blesa, An ant colony optimization algorithm for DNA sequencing by hybridization. Comput. Oper. Res. Vol. 35, No. 11, pp. 3620–3635, 2008.
[37] A. Shmygelska and H. H. Hoos, An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem. BMC Bioinformat. Vol. 6, No. 30, 2005.
[38] O. Korb, T. Stützle, and T. E. Exner, An ant colony optimization approach to flexible protein ligand docking. Swarm Intelli. Vol. 1, No. 2, pp. 115–134, 2007.
[39] S. Sorlin, C. Solnon, and J.-M. Jolion, A Generic Graph Distance Measure Based on Multivalent Matchings, Studies in Computational Intelligence (SCI), vol. 52, pp. 151–182. Springer, 2007.
[40] K. Jones and A. Bouffet, Comparison of Ant Colony Optimisation and Differential Evolution, International Conference on Computer Systems and Technologies – CompSysTech'07, 2007.
[41] E. Bonabeau, M. Dorigo, and G. Theraulaz, Inspiration for optimization from social insect behaviour, Nature, Vol. 406, pp. 39–42, 2000.
[42] M. Dorigo and K. Socha, An Introduction to Ant Colony Optimization. In T. F. Gonzalez (ed.), Approximation Algorithms and Metaheuristics, CRC Press, 2007.
[43] M. Dorigo and L. M. Gambardella, Ant colonies for the traveling salesman problem. BioSystems, Vol. 43, No. 2, pp. 73–81, 1997.
[44] M. Dorigo and L. M. Gambardella, Ant Colony System: A cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, pp. 53–66, 1997.
[45] E. L. Lawler, J. K. Lenstra, A. H. Rinnooy Kan, and D. B. Shmoys, The Travelling Salesman Problem, John Wiley & Sons, Chichester, UK, 1985.
[46] G. Reinelt. The Traveling Salesman: Computational Solutions for TSP Applications, Lecture Notes in Computer Science, Springer Verlag, Berlin, Germany, Vol 840, 1994.
[47] D. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, The Traveling Salesman Problem: A Computational Study, Princeton University Press, Princeton, NJ, 2006.
[48] M. Dorigo and T. Stützle, Ant Colony Optimization: Overview and Recent Advances. In M. Gendreau and Y. Potvin (eds.), Handbook of Metaheuristics, 2nd edition, International Series in Operations Research & Management Science, Springer Verlag, New York, Vol. 146, pp. 227-263, 2010.
[49] M. Dorigo, V. Maniezzo and A. Colorni, Ant System: Optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics-Part B, Vol. 26, No. 1, pp. 29-41, 1996
[50] L. M. Gambardella and M. Dorigo, Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem. Proceedings of ML-95, 12th International

Conference on Machine Learning, Tahoe City, CA, In A. Prieditis and S. Russell (eds.), Morgan Kaufmann, pp. 252-260,1995.
[51] T. Stùtzle and H. H. Hoos, Improving the Ant System: A detailed report on the MAX-MIN Ant System. Technical report AIDA-96-12, FG Intellektik, FB Informatik, TU Darmstadt, Germany, 1996.
[52] B. Bullnheimer, R. F. Hartl and C. Strauss, A new rank-based version of the Ant System: A computational study, Central European Journal for Operations Research and Economics, Vol. 7, No. 1, pp. 25–38, 1999.
[53] M. Dorigo and L. M. Gambardella, Ant Colony System: A cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, pp. 53–66, 1997.
[54] C. Blum, A. Roli and M. Dorigo, HC–ACO: The hyper-cube framework for Ant Colony Optimization, In Proceedings of MIC'2001—Metaheuristics International Conference, Vol. 2, pp. 399–403, 2001.
[55] K. Socha and M. Dorigo, Ant colony optimization for continuous domains, European Journal of Operations Research, Vol. 185, No. 3, pp. 1155–1173, 2008.
[56] S. Tsutsui, Ant colony optimisation for continuous domains with aggregation pheromones metaphor, In Proceedings of the 5th International Conference on Recent Advances in Soft Computing (RASC-04), pp. 207–212, 2004.
[57] K. Socha, ACO for continuous and mixed-variable optimization. In M. Dorigo, L. Gambardella, F. Mondada, T. Stùtzle, M. Birratari, and C. Blum (eds.), Proceedings of the 4th International Conference on Swarm Intelligence (ANTS '04), Lecture Notes in Computer Science 3172, pp. 25–36, 2004.
[58] B. J. Vitins and K.W. Axhausen, Optimization of large transport networks using the ant colony heuristic, Computer-Aided Civil and Infrastructure Engineering, Vol. 24, No. 1, pp. 1-14, 2009.
[59] G. Di Garo and M. Dorigo, An Adaptive Multi-Agent Routing Algorithm Inspired by Ants Behavior, Proc. of 5th Annual Australasian Conf. Para. & Real-Time Sys., 1998.
[60] R. C. Eberhart and Y. Shi, Tracking and optimizing dynamic systems with particle swarms, Proc. Congress on Evolutionary Computation 2001, Seoul, Korea, 2001.
[61] C. Zhang, H. Shao, and Y. Li, Particle Swarm Optimisation for Evolving Artificial Neural Network, In the 2000 IEEE International Conference on Systems, Man, and Cybernetics, vol.4, pp.2487-2490, 2000.
[62] R. C. Eberhart and X. Hu, Human tremor analysis using particle swarm optimization, Proc. Congress on Evolutionary Computation 1999, Washington, DC, pp. 1927-1930, 1999.
[63] M. P. Wachowiak, R. Smolíková, Y. Zheng, J. M. Zurada, and A. S. Elmaghraby, An approach to multimodal biomedical image registration utilizing particle swarm optimization, IEEE Transactions on Evolutionary Computation, 2004.
[64] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi, A particle swarm optimization for reactive power and voltage control considering voltage security assessment, IEEE Transactions on Power Systems, Vol. 15, No. 4, pp. 1232-1239, 2000.

[65] L. Messerschmidt, A. P. Engelbrecht, Learning to play games using a PSO-based competitive learning approach, IEEE Transactions on Evolutionary Computation, 2004.
[66] T. Blackwell and P. J. Bentley, Improvised music with swarms, In David B. Fogel, Mohamed A. El-Sharkawi, Xin Yao, Garry Greenwood, Hitoshi Iba, Paul Marrow, and Mark Shackleton (eds.), Proceedings of the 2002 Congress on Evolutionary Computation CEC 2002, pages 1462–1467, IEEE Press, 2002.
[67] M. H. Hassoun, Fundamentals of Artificial Neural Networks, MIT Press, Cambridge, 1995.
[68] B. MacKinnon, R. Snowden, and S. Dudley (eds.), Sharing the skies: an aviation guide to the management of wildlife hazards, Transport Canada, Ontario, Canada, 2001.
[69] J. T. Emlen, Flocking behaviour in birds, The Auk Journal. Vol. 69, No. 2, pp. 160-170, 1952.
[70] E. Shaw, Fish in schools, Natural History, Vol. 84, No. 8, pp. 40–45, 1975.
[71] J. K. Parrish, S. V. Viscido, and D. Grunbaum, Self-organized fish schools: an examination of emergent properties, Biol. Bull., Vol. 202, pp. 296–305, 2002.
[72] J. Toner and Y. Tu, Flocks, herds, and schools: A quantitative theory of flocking, Physical Review E, Vol. 58, No. 4, pp. 4828–4858, 1998.
[73] A. Okubo, Dynamical aspects of animal grouping: swarms, schools, flocks. and herds, Adv. Biophysics, Vol. 22, pp. 1–94, 1986.
[74] C. W. Reynolds, Flocks, herds, and schools: a distributed behavioural model, Computer Graphics (ACM SIGGRAPH '87 Conference Proceedings), Vol. 21, No. 4, pp. 25–34, July 1987.
[75] M. Jones, K. Pierce Jr., and D. Ward, Avian vision: a review of form and function with special consideration to birds of prey, Journal of Exotic Pet Medicine, Vol.16, No.2, pp.69–87, 2007.
[76] R. Olfati-Saber, Flocking for multi-agent dynamic systems: Algorithms and theory, IEEE Trans. Automatic Control, Vol. 51, No. 3, pp. 401–420, 2006.
[77] V. Gazi and K. M. Passino, Swarm Stability and Optimization, Springer 1st edition, 2011. ISBN: 978-3-642-18040-8.
[78] J. Krause and G. D. Ruxton, Living in Groups, Oxford University Press, Oxford, UK, 2002.
[79] I. D. Couzin, J. Krause, R. James, G. D. Ruxton, and N. R. Franks, Collective Memory and Spatial Sorting in Animal Groups, J. Theor. Biol., Vol. 218, pp. 1–11, 2002.
[80] J. Kennedy, R. C. Eberhart, and Y. Shi, Swarm Intelligence, Morgan Kaufmann, San Francisco, CA, 2001.
[81] N. Xiong, J. He, Y. Yang, Y. He, T. Kim, and C. Lin, A Survey on Decentralized Flocking Schemes for a Set of Autonomous Mobile Robots (Invited Paper). Journal of Communications, Vol. 5, No. 1, pp. 31-38, 2010.
[82] A. Bandura, Social Foundations of Thought and Action: A Social Cognitive Theory, Prentice-Hall, Englewood Cliffs, NJ, 1986.
[83] C. Grosan, A. Abraham, and C. Monica, Swarm Intelligence in Data Mining, Studies in Computational Intelligence, Vol. 34, pp. 1–16. Springer, Heidelberg 2006.

[84] F. Heppner and U. Grenander, A stochastic nonlinear model for coordinated bird flocks. In S . Krasner, Ed., The Ubiquity of Chaos. AAAS Publications, Washington, DC, 1990.
[85] Y. Shi, Feature article on particle swarm optimization, IEEE Neural Network Society, Feature Article, pp. 8–13, Feb. 2004.
[86] J. Kennedy, Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance, In Proceeding of the 1999 Conference on Evolutionary Computation, pp. 1931-1938, 1999.
[87] J. Kennedy and R. Mendes, Population structure and particle swarm performance, Proceeding of the 2002 Congress on Evolutionary Computation, Honolulu, Hawaii, May 2002.
[88] J. Kennedy and R. C. Eberhart, A discrete binary version of the particle swarm algorithm, in Proceeding of the 1997 Conference on Systems, Man, and Cybernetics, pp. 4104-4109, 1997.
[89] C. K. Mohan and B. Al-kazemi, Discrete particle swarm optimization, Proceedings of the Workshop on Particle Swarm Optimization, Indianapolis, IN, 2001.
[90] D. K. Agrafiotis and W. Cedeño, Feature selection for structure-activity correlation using binary particle swarms, Journal of Medicinal Chemistry, Vol. 45, pp. 1098-1107, 2002.
[91] Y. Valle, G. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, R. G. Harley, Particle swarm optimization: basic concepts, variants and applications in power systems. IEEE Trans. on Evolutionary Computation, Vol. 12, No. 2, pp. 171–195, 2008.
[92] J. Kennedy, The particle swarm: Social adaptation of knowledge, in Proc. IEEE Int. Conf. Evolutionary Computation, pp. 303–308, 1997.
[93] A. Abido, Optimal power flow using particle swarm optimization, International Journal of Electrical Power Energy System, Vol. 24, No. 7, pp. 563–571, 2002.
[94] Y. Shi and R. Eberhart, Parameter selection in particle swarm optimization, Evolutionary Programming VII, Lecture Notes in Computer Science, Springer, Vol. 1447, pp. 591–600, 1998.
[95] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Optimization by simulated annealing, *Science*, Vol. 220, No. 4598, pp. 671–680, 1983.
[96] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, Equation of State Calculations by Fast Computing Machines, The Journal of Chemical Physics, Vol. 21, No. 6, p. 1087, 1953.
[97] M. Omran, Particle swarm optimization methods for pattern recognition and image processing, PhD Thesis, University of Pretoria, 2005.
[98] P. Suganthan, Particle Swarm Optimizer with Neighborhood Operator, In Proceedings of the Congress on Evolutionary Computation, pp. 1958-1962, 1999.
[99] M. Clerc, The swarm and the queen: towards a deterministic and adaptive particle swarm optimization, Proc. I999 ICEC, Washington, DC, pp. 1951 – 1957, 1999.
[100] M. Clerc and J. Kennedy, The Particle Swarm: Explosion, Stability, and Convergence in a Multi-dimensional Complex Space, IEEE Transactions on Evolutionary Computation, Vol. 6, pp. 58–73, 2002.

[101] X. Xie, W. Zhang, and Z. Yang, A dissipative particle swarm optimization, In Proc. IEEE Congr. Evol. Comput., Vol. 2, pp. 1456–1461, May 2002.
[102] K. Parsopoulos and M. Vrahatis, Recent approaches to global optimization problems through particle swarm optimization, Natural Computing, Vol. 1, pp. 235–306, May 2002.
[103] B. Secrest and G. Lamont, Visualizing particle swarm optimization— Gaussian particle swarm optimization, In Proc. IEEE Swarm Intell. Symp., pp. 198–204, Apr. 2003.
[104] N. Higashi and H. Iba, Particle Swarm Optimization with Gaussian Mutation, In Proceedings of the IEEE Swarm Intelligence Symposium'03 (SIS 2003), Indianapolis, Indiana, USA. pp. 72-79, 2003.
[105] R. Mendes, J. Kennedy, and J. Neves, The Fully Informed Particle Swarm: Simpler, may be Better, IEEE Transactions on Evolutionary Computation, Vol. 8, pp. 204–210, 2004.
[106] X. Li, Adaptively choosing neighborhood bests using species in a particle swarm optimizer for multimodal function optimization, In Proc. Genetic Evol. Comput. Conf., pp. 105–116, Jun. 2004.
[107] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Trans. Evol. Comput., Vol. 8, No. 3, pp. 240–255, 2004.
[108] F. Van den Bergh, A. P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Transactions on Evolutionary Computation, Vol. 3, pp.225–239, 2004.
[109] J. Liang, A. Qin, P. Suganthan, and S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput., Vol. 10, No. 3, pp. 281–295, Jun. 2006.
[110] F. T. Chan and M. K. Tiwari (eds.), Swarm Intelligence: Focus on Ant and Particle Swarm Optimization, I-Tech Education and Publishing, Vienna, Austria, 2007.
[111] T. M. Blackwell, Particle swarm optimization in dynamic environments. In S. Yand, Y. Ong, and Y. Jin (eds.), Evolutionary computation in dynamic environments, Springer, Berlin, pp. 29–49, 2007.
[112] P. Melin and W. Pedrycz (eds.), Soft Computing for Recognition based on Biometrics, Studies in Computational Intelligence, Springer, Vol. 312, 2010.
[113] C. Coello Coello and M. Lechuga, MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization, In Congress on Evolutionary Computation, Vol. 2, pp. 1051-1056, 2002.
[114] Y. Rahmat-Samii, Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) in Engineering Electromagnetics, The 17th International Conference on Applied Electromagnetics and Communications (ICECom'03), pp. 1-5, 2003.
[115] P. J. Angeline, Evolutionary Optimization Versus Particle Swarm Optimization: Philosophy and Performance Differences, Evolutionary Programming VII, Lecture Notes in Computer Science, Springer, Vol. 1447, pp. 601–610, 1998..
[116] R. Hassan, B. Cohanim, O. Weck, and G. Venter, A Comparison of Particle Swarm Optimization and the Genetic Algorithm, In the 46th

AIAA/ASME/ASCE/AHA/ASC Structures, Structural Dynamics and Materials, American Institute of Aeronautics and Astronautics, Reston, Vol. 2, pp. 1138-1150, 2005.
[117] R. Eberhart and Y. Shi, Comparison between Genetic Algorithms and Particle Swarm Optimization, In Proceedings of the 7th Annual Conference on Evolutionary Programming, Springer-Verlag, pp. 611-619, 1998.
[118] J. Kennedy and W. Spears, Matching Algorithms to Problems: An Experimental Test of the Particle Swarm and Some Genetic Algorithms on the Multimodal Problem Generator, In IEEE International Conference on Evolutionary Computation, Achorage, Alaska, USA, 1998.
[119] J. Holland. Adaptation in Natural and Artificial Systems. University of Michigan Press, 1975.
[120] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, pp. 1-25, 1989.
[121] D. E. Goldberg, Genetic Algorithms, New York, Addison-Wesley, Ch 1-4, 1989.
[122] J. H. Holland, Genetic algorithms, Scientific American, pp. 66-72, 1992.
[123] A. Lazinica, Particle Swarm Optimization, Ed. In-Tech. Vienna, Austria, 2009.
[124] K. Premalatha and A. M. Natarajan, Hybrid PSO and GA for Global Maximization, Int. J. Open Probl. Comput. Sci. Math, 2009.
[125] I. C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection Inf. Process. Lett., Vol. 85, pp. 317–25. 2003.
[126] M. Løvberg. Improving Particle Swarm Optimization by Hybridization of Stochastic Search Heuristics and Self Organized Critically, Master's Thesis. Department of Computer Science, University of Aarhus, Denmark, 2002.
[127] M. Løvberg and T. Krink, Extending Particle Swarm Optimizers with Self-Organized Criticality, In Proceedings of the Fourth Congress on Evolutionary Computation, Vol. 2, pp. 1588-1593, 2002.
[128] K. Veeramachaneni, T. Peram, C. Mohan and L. Osadciw, Optimization Using Particle Swarm with Near Neighbor Interactions, Lecture Notes Computer Science, Springer Verlag, Vol. 2723, 2003.
[129] M. Dorigo, In The Editorial of the First Issue of: Swarm Intelligence Journal, Springer Science + Business Media, LLC, Vol.1, No. 1, pp. 1–2, 2007.
[130] F. Buck, Cooperative Problem Solving With a Distributed Agent System - Swarm Intelligence, Master's Thesis, Dept. of Electrical and Computer Engineering, Utah State University, 2005.
[131] L. Chen, X.-H. Xu, and Y.-X. Chen, An adaptive ant colony clustering algorithm, In Proceedings of the 3rd Conference on Machine Learning and Cybernetics, pp. 1387–1392, 2004.
[132] Y. Shi and R. C. Eberhart, Fuzzy adaptive particle swarm optimization, In Proceedings of 2001 Congress on Evolutionary Computation, pp. 101–106. 2001.
[133] N. I. Ghali, N. El-Dessouki, A. N. Mervat, and L. Bakrawi, Exponential Particle Swarm Optimization Approach for Improving Data Clustering, International Journal of Electrical, Computer, and Systems Engineering, pp. 208-212, 2009.

[134] J. Tautz, K. Rohrseitz, and D. C. Sandeman, One-strided waggle dance in bees. Nature, Vol. 382, p. 32, 1996.
[135] T. D. Seeley, The wisdom of the hive, Harvard University Press, Cambridge, MA, 1995.
[136] E. Bonabeau, G. Theraulaz, J.-L. Deneubourg, S. Aron, and S. Camazine, Self-organization in social insects, Trends in Ecol. Evol., Vol. 12, pp. 188-193, 1997.
[137] K. Frisch von, The dance language and orientation of bees, Harvard University Press, Cambridge, MA, 1967.
[138] J. Chen, E. Antipov, B. Lemieux, W. Cedeno, and D. H. Wood, DNA computing implementing genetic algorithms, Evolution as Computation, Springer Verlag, New York, pp. 39–49, 1999.
[139] M. Perretto and H. S. Lopes, Reconstruction of phylogenetic trees using the ant colony optimization paradigm, Genetic and Molecular Research, Vol. 4, No. 3, pp. 581–589, 2005.
[140] M. Beekman, G. A. Sword, and S. J. Simpson, Biological foundations of swarm intelligence. In C. Blum and D. Merkle (eds.) Swarm Intelligence. Introduction and Applications, Springer, Berlin, Germany, pp. 3-41. 2008.
[141] C. Blum and X. Li, Swarm Intelligence in Optimization. In C. Blum and D. Merkle (eds.) Swarm Intelligence. Introduction and Applications, pp. 43-85. Springer, Berlin, Germany, 2008.
[142] J. D. Moss and C. G. Johnson, An ant colony algorithm for multiple sequence alignment in bioinformatics, In D. W. Pearson, N. C. Steele, and R. F. Albrecht, (eds.), Artificial Neural Networks and Genetic Algorithms, pp. 182–186. Springer, Berlin, Germany, 2003.
[143] P. Meksangsouy and N. Chaiyaratana, DNA fragment assembly using an ant colony system algorithm, in Proc. of Congress on Evolutionary Computation, IEEE Press, USA, 2003.
[144] O. Karpenko, J. Shi, and Y. Dai, Prediction of MHC class II binders using the ant colony search strategy, Artificial Intelligence in Medicine, Vol. 35, No. 1-2, pp. 147–156, 2005.
[145] R. Poli, An analysis of publications on particle swarm optimization applications. Technical Report CSM-469, Dept. Computer Science, University of Essex, UK, 2007.
[146] C.-H. Chen and C.-T. Chu, Low complexity iris recognition based on wavelet probabilistic neural networks, In Proceedings of IEEE International Joint Conference on In Neural Networks (IJCNN ’05), Vol. 3, pp. 1930 – 1935, 2005.
[147] J. Haiyan and Y. Jinli, The application study of apple color grading by particle swarm optimization neural networks, The 6th World Congress on Intelligent Control and Automation (WCICA’06), pp. 2651 – 2654, 2006.
[148] M. Sugisaka and X. Fan, An effective search method for NN-based face detection using PSO. In SICE Annual Conference, Vol. 3, pp. 2742 – 2745, 2004.
[149] S. Das, A. Abraham, and A. Konar, Spatial information based image segmentation using a modified particle swarm optimization algorithm. In the 6th International Conference on Intelligent Systems Design and Applications (ISDA ’06), pp. 438 – 444, 2006.

[150] W. Brinkman and T. Thayaparan, Focusing ISAR images using the AJTF optimized with the GA and the PSO algorithm – comparison and results, In IEEE Conference on Radar, pp. 24-27, April 2006.
[151] V. Ciesielski, G. Wijesinghe, A. Innes, and S. John, Analysis of the superiority of parameter optimization over genetic programming for a difficult object detection problem, In IEEE Congress on Evolutionary Computation (CEC'06), pp. 1264 – 1271, 2006.
[152] K. Chandramouli and E. Izquierdo, Image classification using chaotic particle swarm optimization, In IEEE International Conference on Image Processing, pp. 3001 – 3004, 2006.
[153] W. H. Slade, H. W. Ressom, M.T. Musavi, and R.L. Miller, Inversion of ocean color observations using particle swarm optimization, IEEE Transactions on Geoscience and Remote Sensing, Vol. 42, pp. 1915 – 1923, 2004.
[154] H. Zhang and D. Luo, A PSO-based method for traffic stop-sign detection, The 6th World Congress on Intelligent Control and Automation (WCICA'06), pp. 8625 – 8629, 2006.
[155] D.-M. Tsai, Y.-H. Tseng, S.-M. Chao, and C.-H. Yen, Independent component analysis based filter design for defect detection in low-contrast textured images, The 18th International Conference on Pattern Recognition (ICPR'06), pp. 231 – 234, 2006.
[156] K. Kameyama, N. Oka, and K. Toraichi, Optimal parameter selection in image similarity evaluation algorithms using particle swarm optimization, In IEEE Congress on Evolutionary Computation (CEC'06), pp. 1079 – 1086, 2006.
[157] Y. Owechko, S. Medasani, and N. Srinivasa, Classifier swarms for human detection in infrared imagery, In the IEEE Computer Vision and Pattern Recognition (CVPR'04), pp. 121 – 121, 2004.
[158] S. Das, A. Abraham, and S. K. Sarkar, A hybrid rough set–particle swarm algorithm for image pixel classification, The 6th International Conference on Hybrid Intelligent Systems (HIS '06), p. 26, 2006.
[159] Y. Owechko and S. Medasani, Cognitive swarms for rapid detection of objects and associations in visual imagery, In Proceedings of IEEE Swarm Intelligence Symposium (SIS'05), pp. 420 – 423, 2005.
[160] P. Saisan, S. Medasani, and Y. Owechko, Multi-view classifier swarms for pedestrian detection and tracking, In the IEEE Computer Vision and Pattern Recognition (CVPR'05), p. 18, 2005.
[161] Y. Zhang, Y. Meng, W.-H. Li, and Y.-J. Pang, A fast algorithm for image analogy using particle swarm optimization, In Proceedings of the International Conference on Machine Learning and Cybernetics, Vol.7, pp. 4043 – 4048 , 2004.
[162] T. Huang and A. S. Mohan, Significance of neighborhood topologies for the reconstruction of microwave images using particle swarm optimization, In Proc. of Asia-Pacific Microwave Conference (APMC'05), pp. 237–240, 2005.
[163] O. Sjahputera and J. M. Keller, Particle swarm over scene matching, In Proc. of IEEE Swarm Intelligence Symposium (SIS'05), pp. 108 – 115, 2005.
[164] B. Fumin, L. Aiguo, and Q. Zheng, Photo time-stamp recognition based on particle swarm optimization, In Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI'04), pp. 529 – 532, 2004.

[165] N. M. Kwok, Q. P. Ha, D. K. Liu, and G. Fang, Intensity-preserving contrast enhancement for gray-level images using multi-objective particle swarm optimization. In the IEEE International Conference on Automation Science and Engineering (CASE'06), pp. 21 – 26, 2006.
[166] Q. Sun, Y. Shi, R. C. Eberhart, and W. A. Bauson, Utilizing particle swarm optimization to label a structured beam matrix. In Proceedings of the IEEE Swarm Intelligence Symposium (SIS '03), pp. 118 – 123, 2003.
[167] M. Zhang, L. Zhang, Y. Sun, L. Feng, and W. Ma, Auto cropping for digital photographs. In the IEEE International Conference on Multimedia and Expo (ICME 2005), 2005.
[168] Y. Li, J. Li, and L. Meng, Character recognition based on hierarchical RBF neural networks, In the 6th International Conference on Intelligent Systems Design and Applications (ISDA'06), pp. 127 – 132, 2006.
[169] J.-X. Du, D.-S.Huang, J. Zhang, and X.-F. Wang, Shape matching using fuzzy discrete particle swarm optimization. In Proceedings of the IEEE Swarm Intelligence Symposium (SIS'05), pp. 405 – 408, 2005.
[170] Y.-C. Chen, H.-C. Wang, and T.-J. Su, Particle swarm optimization for image noise cancellation. In the First International Conference on Innovative Computing, Information and Control (ICICIC'06), pp. 587 – 590, 2006.
[171] H. K. Arachchi and W. A. Fernando, PSO based bit rate optimization for MPEG-1/2 video coding. In the IEEE International Conference on Image Processing (ICIP'05), pp. II– 329–332, 2005.
[172] G.-Y. Du, T.-S. Huang, L.-X. Song, and B.-J. Zhao, A novel fast motion estimation method based on particle swarm optimization, The International Conference on Machine Learning and Cybernetics, Vol. 8, pp. 5038 – 5042, 2005.
[173] L. Anton-Canalis, M. Hernandez-Tejera, and E. Sanchez-Nielsen, Particle swarms as video sequence inhabitants for object tracking in computer vision, In the 6th International Conference on Intelligent Systems Design and Applications (ISDA'06), pp. 604 – 609, 2006.
[174] S. Ivekovic and E. Trucco, Human body pose estimation with PSO. In the IEEE Congress on Evolutionary Computation (CEC'06), pp. 1256 – 1263, 2006.
[175] D. Srinivasan, W. H. Loo, and R. L. Cheu, Traffic incident detection using particle swarm optimization, In Proceedings of the IEEE Swarm Intelligence Symposium (SIS'03), pp. 144 – 151, 2003.
[176] H. W. Ressom, Y. Zhang, J. Xuan, Y. Wang, and R. Clarke, Inference of gene regulatory networks from time course gene expression data using neural networks and swarm intelligence, In the IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology (CIBCB'06), pp. 1 – 8, 2006.
[177] N. Khemka, C. Jacob, and G. Cole, Making soccer kicks better: a study in particle swarm optimization and evolution strategies, In the IEEE Congress on Evolutionary Computation, Vol.1, pp. 735 – 742, 2005.
[178] H.-Y. Lv, W.-G. Zhou, and C.-G. Zhou, A discrete particle swarm optimization algorithm for phylogenetic tree reconstruction, In Proceedings of the International Conference on Machine Learning and Cybernetics, Vol. 4, pp. 2650 – 2654, 2004.
[179] R. Xu, G. C. Anagnostopoulos, and D. C. Wunsch, Multiclass cancer classification using semi-supervised ellipsoid art-map and particle swarm

optimization with gene expression data. In the IEEE/ACM Transactions on Computational Biology and Bioinformatics, pp. 65 – 77, 2007.
[180] R. Xu, X. Cai, and D. C. Wunsch, Gene expression data for DLBCL cancer survival prediction with a combination of machine learning technologies, In the 27th Annual International Conference of the Engineering in Medicine and Biology Society (IEEE-EMBS'05), pp. 894 – 897, 2005.
[181] C. T. Hardin and E. C. Rouchka, DNA motif detection using particle swarm optimization and expectation-maximization, In IEEE Proceedings of Swarm Intelligence Symposium (SIS 2005), pp. 181 – 184, 2005.
[182] X. Xiao, E. R. Dow, R. C. Eberhart, Z. B. Miled, and R. J. Oppelt, Gene clustering using self-organizing maps and particle swarm optimization, In Proceedings of the 2nd IEEE international workshop on high performance computational biology, Nice, France, 2003.
[183] X.-Y. Chang, C.-G. Zhou, Y.-W. Li, and P. Hu, Identification of transcription factor binding sites using GA and PSO, In the 6th International Conference on Intelligent Systems Design and Applications (ISDA'06), pp. 473 – 480, 2006.
[184] H. W. Ressom, R. S. Varghese, E. Orvisky, S. K. Drake, G. L. Hortin, M. Abdel-Hamid, C. A. Loffredo, and R. Goldman, Analysis of MALDI-TOF serum profiles for biomarker selection and sample classification, In Proceedings of IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB '05), pp. 1 – 7, 2005.
[185] S.-H. Doong, Protein homology modeling with heuristic search for sequence alignment, In the 40th Annual Hawaii International Conference on System Sciences (HICSS'07), pp. 128 – 128, 2007.
[186] B.-F. Liu, H.-M. Chen, H.-L. Huang, S.-F. Hwang, and S.-Y. Ho, Flexible protein-ligand docking using particle swarm optimization. In IEEE Congress on Evolutionary Computation, Vol.1, pp. 251 – 258, 2005.
[187] W. Cedefto and D. Agraflotis, Particle swarms for drug design. In IEEE Congress on Evolutionary Computation, Vol. 2, pp. 1218 – 1225, 2005.
[188] Y. Li, D. Yao, and W. Chen, Adaptive particle swarm optimizer for beam angle selection in radiotherapy planning. In IEEE International Conference Mechatronics and Automation, Vol. 1, pp. 421 – 425, 2005.
[189] L. Xie and L. Jiang, Global Optimal ICA and its Application in Brain MEG Data Analysis, In the International Conference on Neural Networks and Brain (ICNN&B'05), pp. 353 – 357, 2005.
[190] M. Neethling and A. P. Engelbrecht, Determining RNA secondary structure using set-based particle swarm optimization. In IEEE Congress on Evolutionary Computation (CEC'06), pp. 1670 – 1677, 2006.
[191] L. Qiu, Y. Li, and D. Yao, A feasibility study of EEG dipole source localization using particle swarm optimization, In IEEE Congress on Evolutionary Computation, Vol.1, pp. 720 – 726, 2005.
[192] K. Veeramachaneni, L. A. Osadciw, and P.K. Varshney, An adaptive multimodal biometric management algorithm, IEEE Transactions on Systems, Man and Cybernetics, Part C, Vol. 35, pp. 344 – 356, 2005.