**CHAPTER ONE**

**INTRODUCTION**

**1.1 BACKGROUND TO THE STUDY**

Nature conceals many mysteries. In the past, some of these behaviours like ants foraging and flight flocks were considered as magical secrets of nature. Nature always plays a vital role to solve complex human problems. In the past few years biology based techniques get the attentions of researchers. These and other phenomena inspired researchers to study and understand their secrets. The unraveling of many of these mysteries and secrets led to the foundation of new artificial intelligence science known as Swarm Intelligence (SI)

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 intelligence refers to systems which accomplish complex global tasks through the simple local interactions of autonomous agents. The control is completely distributed among the individual agents with no leader coordinating any of the activities

Nature always plays a vital role to solve complex human problems. In the past few years biology based techniques get the attentions of researchers. Many natural biological inspired techniques have been proposed for the network security, one of them is swarm intelligence.

Swarm intelligence is the emergent collective intelligence of groups of simple agents. It is a computational intelligence approach to solve real world complex problems.

It was first introduced in cellular robotics system by Beni and Wang in 1989. Swarm intelligence systems buildup of a population of simple agents interactive with each other individually or with their environment. The inspiration of swarm intelligence comes from the biological or natural system. Example of SI includes ant colonies, bees, fish schooling, bird flocking and animal herding bacterial growth.

The complex problem that seems almost impossible at individual level is solved by insects, bees and birds in the form of swarms.

Researchers have done so many work in this field and create many swarm intelligence based algorithms models and applications.

Few important Swarm Intelligence (SI) algorithms are;

Algorithms:

* Ant colony optimization algorithm
* Artificial Bee colony algorithm
* Particle swarm optimization
* Firefly Algorithm
* Multi-swarm optimization
* River Formation Dynamics
* Bacterial Foraging algorithm ‎
* Cat Swarm Optimization algorithm
* Artificial Immune System algorithm
* Glowworm Swarm Optimization algorithm

**Why is Swarm Intelligence interesting for IT?**

Swarm intelligence is so interesting because of the following analogies.

* distributed system of interacting autonomous agents.
* goals: performance optimization and robustness.
* self-organized control and cooperation (decentralized)
* division of labour and distributed task allocation.
* indirect interactions

Sharvani G S (2012). It is found that the Swarm Intelligence (SI) inspired algorithms such as Ant Colony Optimization (ACO) are better suited for highly adaptive networks like Mobile Ad hoc Networks (MANETs).

Biological ants at the time of food foraging, navigate their chosen path and deposit a chemical called pheromone on the ground, there by establishing the trail. Thickness of the trail attracts other ants to follow the path to reach the food source. The principles of ACO are used for each packet flow in MANETs, resulting in emergent routing behavior. Other additional benefits achieved from ACO are reduced control overhead and efficient route maintenance.

A set of wireless communication nodes performing self-configuration in a dynamic mode for formation of network excluding fixed infrastructure or centralized supervision is termed as mobile ad hoc network (MANET). Often, there may be random changes in the network topology as nodes are mobile. In addition to the role of router, the nodes also play the role of end host. The routing protocol in such a network is an authority to determine the routes and offering communication among end points via intermediate nodes. The MANET is well-liked and attractive since they offer good communication in the changing infrastructure for the applications such as rescue operations, tactical operations, environmental monitoring, conferences, and the like.

Distributed systems like peer-to-peer networks, social networks, and mobile ad hoc networks require cooperation among the participating entities to guarantee the formation and sustained existence of network services. The reliability of interactions among anonymous entities is a significant issue in such environments. The distributed entities establish connections to interact with others, which may include selfish and misbehaving entities and result in bad experiences.

Therefore, trustworthiness evaluation using trust management techniques has become a significant issue in securing these environments to allow entities decide on the reliability and trustworthiness of other entities, besides it helps coping with defection problems and stimulating entities to cooperate. Recent models on evaluating trust and security in distributed systems have heavily focused on assessing trustworthiness of entities and isolate misbehaviors based on trust metrics.

**1.2 RESEARCH MOTIVATION**

According to Iftikhar and Fraz (2013) research work is motivated by natural systems because they have many characteristics that might be used for the network security purposes. Like, a swarm of insect with very confined capabilities still finishes very complex tasks. Swarm based security systems or intrusion detection systems are light in weight and simple to put into practice. They are quite robust, vastly adaptive to different conditions and more importantly self-configurable.

(Shabut A.R.M. 2015). Due to the recent applicability and performance of mobile ad hoc networks in future paradigms, including vehicular and mesh networks, as well as many civilian and military services ranging from emergency rescue services to exchanging critical information on the battlefield or even home and personal area networking, MANET’s security has been investigated in the literatures using different techniques. The formation and sustained existence of MANET services is mainly based on an individual node’s cooperation in packet forwarding. It is indeed a challenge to safeguard MANETs with a lack of infrastructure (i.e. pre-existing communication backbone) and central authority (such as base stations or mobile switching centres) to establish and facilitate communication in the network against a wide range of attacks. Due to these unique characteristics and demands for use, MANETs are vulnerable to attacks launched by misbehaving nodes (Wu et al 2007). Trust management techniques are put forward as one of the approved mechanisms to improve security in MANETs to deal with misbehaving nodes and stimulate them to cooperate (Li et al 2012).

In recent years, different trust management models have been proposed to enhance security in MANETs to enable nodes to evaluate their neighbours directly or through recommendations from other nodes in the network. CORE (COllaborative REputation) by Michiardi and Molva (2002), Context-Aware Detection by Paul and Westhoff (2002), Cooperation Of Nodes-Fairness In Dynamic Ad-hoc NeTworks (CONFIDANT) by Buchegger and Le Boudec (2004), to name a few, are mechanisms which support cooperation in ad hoc networks by detecting and isolating malicious nodes. Although the proposed models have paid attention to the problem of misbehaving nodes, multi-dimensional trust metrics including social properties of trust to deal with situations like dishonest recommendations are still in their early stages. This is considered a research gap in which the focus of such models is directed toward a single parameter only in computing security. Many of the existing models seem to be filtering untrustworthy nodes by only considering a packet forwarding metric for example in improving the overall performance in the network and are not efficient enough in handling other misbehaving nodes related to dishonest recommendations.

According to Shabut, A., et al, (2013). Some models omit some important evaluation metrics such as quality of service and social properties in evaluating nodes’ trustworthiness. By considering this problem, a security evaluation mechanism should be effective in encouraging cooperation between nodes, enhancing the false negatives and false positives in judging the behaviour of nodes by utilising multi- dimensional trust attributes.

**1.3 RESEARCH SCOPE AND LIMITATION**

Due to the numerous algorithms of swarm intelligence and multiple areas of applications, this research work will only explore the concept of swarm intelligence system and how it can be deployed in network security operational scenarios using Mobile Ad-hoc Network (MANET)

The research will only focus on two Algorithms, which are

* Ant colony optimization algorithm (ACO)
* Particle swarm optimization (PSO)

**1.4 RESEARCH OBJECTIVES**

The specific objectives of this research are to:

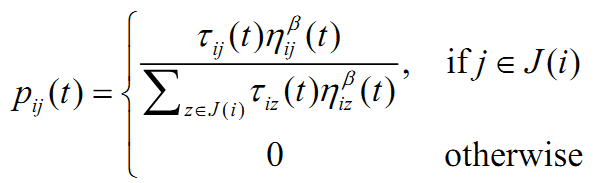
1. simulate a Mobile Ad-hoc Network (MANET) model based on Swarm Intelligence algorithm.
2. design a trust and reputation management model using probability theory technique to monitor the nodes activities and evaluate the security of ‘a’ above.
3. implement a, and b above
4. evaluate the performance of the proposed system

**1.5 RESEARCH METHODOLOGY**

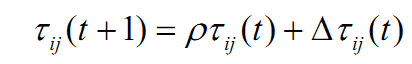
A thorough research on related literature will be carried out to get a better understanding of the previous work in the field of Swarm Intelligence. It was also be used to gain an increasing knowledge within the area of Swarm Intelligence (SI) algorithm models.

* A Mobile Ad-hoc Network will be simulated using hybridized algorithms of Ant Colony Optimization (ACO) and Particle swarm optimization (PSO) to survey the network nodes and to detect threats based on the set threat characteristics at any of the network nodes

ACO model

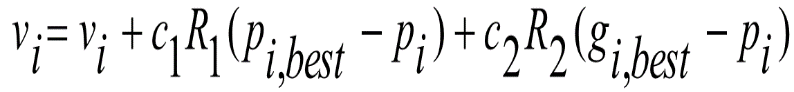
 1

where J(i) i is the set of vertices that remain to be visited by the ant, β is a parameter that determines the relative influence of the pheromone trail and the heuristic information. After all ants have completed their tours, the pheromone level is updated by



2

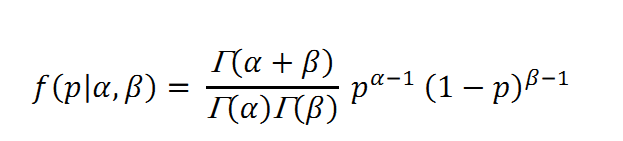
and PSO model

 3

where pi and vi are the position and velocity of particle i, respectively; pi,best and gi,best is the position with the ‘best’ objective value found so far by particle i and the entire population respectively; w is a parameter controlling the dynamics of flying; R1 and R2 are random variables in the range [0,1]; c1 and c2 are factors controlling the related weighting of corresponding terms

This research will make use of a Bayesian statistical approach for computing trust values based on the assumption that they follow a beta probability distribution.

Beta distribution is estimated by using two parameters (α, β). These can be calculated by accumulating observations of forwarding and dropping packets where α represents the accumulation of positive observations (forwarded packets) and β represents the accumulation of negative observations (dropped packets).

 4 Where 0 ≤ Ƥ≤ 1, α, β, > 0 with a condition that Ƥ ≠ 0 if α< 1 and Ƥ ≠1 if β < 1.

Nodes in the network observe each other’s behaviour in order to construct a trust relationship representing the degree of trustworthiness one node (known as an evaluating node) can place on another (known as the evaluated node) by evidence collected by the node itself or by other nodes (known as recommending nodes). The trust is measured as a real number in the range of 0-1, in which 0 denotes that the node is completely untrustworthy or unsecured and 1 denotes that the node is completely trustworthy and secure.

Therefore, to conduct this research, NS2 simulator is to be used. The underlying trust components are added to the simulator to build the proposed models based on the following assumptions:

**Assumption 1**: All nodes have a unique ID and they are not able to change identity during the simulation time.

**Assumption 2**: Nodes are operating in promiscuous mode and they can listen to transmitting packets within their transmission range.

**Assumption 3**: Links between every two nodes are symmetric and omni-directional antennas.

**Assumption 4**: Correct forwarding of packets is the main behaviour to evaluate nodes’ security.

**Assumption 5**: Trustworthiness of nodes is based on a predefined threshold.

**Assumption 6**: Nodes can only monitor the behaviours of their one hop neighbours; consequently, they are able to evaluate their trustworthiness.

**Assumption 7**: Nodes can evaluate trustworthiness of two or more than one hop neighbours by using the monitoring information of other nodes which have interacted with them.

**Assumption 8**: Nodes can evaluate the future behaviour of each other based on past experiences of direct observations and recommendations.

**Assumption 9**: Trust among nodes is asymmetric and is not completely transitive

**Assumption 10**: Nodes’ behaviours are independent from each other except for some attacks in which attackers collude together to achieve a specific attack.

Therefore, to conduct this research, NS3 simulator is to be used. The underlying trust components are added to the simulator to build the proposed models based certain assumptions which will be discussed later in chapter 3.

This research work will be implemented using at least Network Simulator 3. A table will be constructed to provide a summary of the chosen simulation parameter values.

A quantitative assessment will be performed Bayesian statistical technique based on the proposed swarm Intelligence algorithm modes. This helped evaluate the strengths, security, privacy and trust of the algorithm.

**1.6. CONTRIBUTIONS TO KNOWLEDGE**

This work is expected to establish a security and trust model in a network environment using Swarm Intelligence.

**CHAPTER TWO**

**LITERATURE REVIEW**

**2.1 HISTORICAL DEVELOPMENT OF SWARM INTELLIGENCE SYSTEM**

In various areas like computer science applications, robotics etc there always arise a need to design an intelligent system capable of performing complex tasks with the help of multiple self-organizing autonomous nodes which are distributed in nature and have no central control. Many researchers were inspired by collective behaviour of groups of animals like school of fishes, flocks of birds and social insects like ants, bees etc while designing such systems. Various algorithms were designed for distributed problem solving devices based on the intelligent behaviour of swarms. This way of problem solving which was inspired by the intelligent collective behaviour of swarms was defined as swarm intelligence. (Jangra et al, 2013)

In nature swarms (ants, bees, termites etc) consists of simple creatures as individuals which have limited intellectual capabilities still the swarm as a whole presents an intelligent and efficient solution for complex problems such as shortest path finding , predator evasion. If we take the example of ants and bees they both search for their food but using somewhat different strategies. Ants are capable of finding the shortest path for their food and they do it by communicating through their environment using a chemical substance called pheromones as they can’t communicate directly. This indirect communication through the environment is known as stigmergy. Honey bee swarms are capable of finding good quality food sources. They do it by sending their scout bees to search for food sources and then after searching good quality food source scout bees perform a kind of dance which encodes some information by which conveys the direction and quality of the food source. The location for which enough number of scouts vote is chosen.

Thus based on such intelligent behaviour of swarms various algorithms have been designed.

Swarm intelligence is a modern artificial intelligence discipline that is concerned with the design of multiagent systems with applications, e.g., in optimization and in robotics. The design paradigm for these systems is fundamentally different from more traditional approaches. Instead of a sophisticated controller that governs the global behavior of the system, the swarm intelligence principle is based on many unsophisticated entities that cooperate in order to exhibit a desired behavior. Inspiration for the design of these systems is taken from the collective behavior of social insects such as ants, termites, bees, and wasps, as well as from the behavior of other animal societies such as flocks of birds or schools of fish.

Even though the single members of these societies are unsophisticated individuals, they are able to achieve complex tasks in cooperation. Coordinated behavior emerges from relatively simple actions or interactions between the individuals. Moreover, engineers are increasingly interested in this kind of swarm behavior since the resulting “swarm intelligence” can be applied in optimization for ex. in telecommunicate systems, robotics, traffic patterns in transportation systems and military applications.

Swarm intelligence is the emergent collective intelligence of groups of simple autonomous agents. Here, an autonomous agent is a subsystem that interacts with its environment, which probably consists of other agents, but acts relatively independently from all other agents. (Sumit and Nirmaljit 2014).

**2.2 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, Particle Swarm Optimization, Artificial Bee Colony, Bacterial Foraging, Cat Swarm Optimization, Artificial Immune System, and Glowworm Swarm Optimization.

This section will primarily focus on two of the most popular swarm intelligences models, namely, Ant Colony Optimization and Particle Swarm Optimization.

**2.2.1 Ant Colony Optimization (ACO) Model**

The first example of a successful swarm intelligence model is Ant Colony Optimization (ACO), which was introduced by Dorigo et al, and has been originally used to solve discrete optimization problems in the late 1980s. (Hazem and Janice 2012**)**

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.

Ant Colony Optimization principles are based on the natural behaviour of ants. In their daily life, one of the tasks ants have to perform is to search for food, in the vicinity of their nest.

While walking in such a quest, the ants deposit a chemical substance called pheromone in the ground. This is done with two objectives. On the one hand, it allows ants to ﬁnd their way back to the nest, such as Hansel and Gretel in the fairytale. And on the other hand, it allows other ants to know the way they have taken, so that the others can follow them. The curiosity is that, because hundreds or even thousands of ants have this behaviour, if one could see the pheromone laid in the ground as a kind of light, the ground would be a large network with some of the arcs brighter than the others. Within the paths created by those arcs would surely be the shortest path between the nest and the food source. This behaviour can be seen as a kind of communication between the ants. If the path has a large concentration of pheromone, this is probably due to its shorter length that allowed ants to travel faster, resulting in a larger number of travels through the path therefore with much more ants depositing pheromone on it. Furthermore, over time the pheromone evaporates and thus its concentration reduces. The more time it takes for the ant to travel from the nest to the food source and back to the nest, the more time the pheromones have to evaporate. This system is thus based both on the positive feedback, i.e. depositing of pheromone attracts other ants to use the same path which will increase the pheromone quantity, and on negative feedback, i.e. dissipating of the pheromone through evaporation leads to lower levels of pheromone thus discouraging other ants. Deneubourg et al (1990) and Goss et al (1989) performed some experiences with real ants and they were able to show that foraging ants can ﬁnd the shortest path between their nest and some food source, by the use of a chemical substance called pheromone that they deposit while walking. After these experiments the authors proposed a stochastic model to describe what they had observed. This was the ﬁrst step leading to an optimization algorithm based on the foraging behaviour of ants.

Ant Colony Optimisation is a probabilistic technique which may be used for solving computational problems in Mobile ad hoc networks (MANETs - which will be discussed later). ACO was introduced by Marco Dorigo in the year 1992 and was originally called the Ant System. ACO studies artificial systems which are inspired by the behavior of real ant colonies and the ACO algorithms can be used to solve discrete optimization problems. The foraging behavior of real ant colonies is modeled in ACO algorithms and the optimal path is determined by generating artificial ants. In nature the real ants search for food in their environment, the artificial ants in the system modeled too search the solution space for optimal routes. The nature of ants is to move away from their nest to wander randomly in search of food. They lay along a trail of chemical substance known as pheromone as they move which the other ants have the ability to detect and follow. The strength of the pheromone deposit directs the artificial ants toward the best paths and has the ability to evaporate with time and avoiding the convergence to a locally optimal solution.

The system which is probabilistically determined allows the movement of ants such that the ants explore new paths and can also re-explore the paths that have been visited earlier. The basic idea of the Ant Colony Algorithm is to simulate the behavior of ants within the artificial ants and use them to find an optimal solution in routing in MANETs. There have been classes of algorithms which have been inspired on the behavior of foraging ants.

The ACO algorithm’s basic principle is to obtain information about the routes through repeated sampling, ants which are nothing but some control packets are sent for this purpose (Baras and Mehta 2003). Ants are created and released by the source node to test check and acquire information for computing parameters like end-to end delay, no. of hops etc. to the destination that it wants to send data to. Ants upon acquiring information about a route update the intermediary nodes whilst going back to the source node.

Ants constantly sample complete paths from one source node to the destination node.

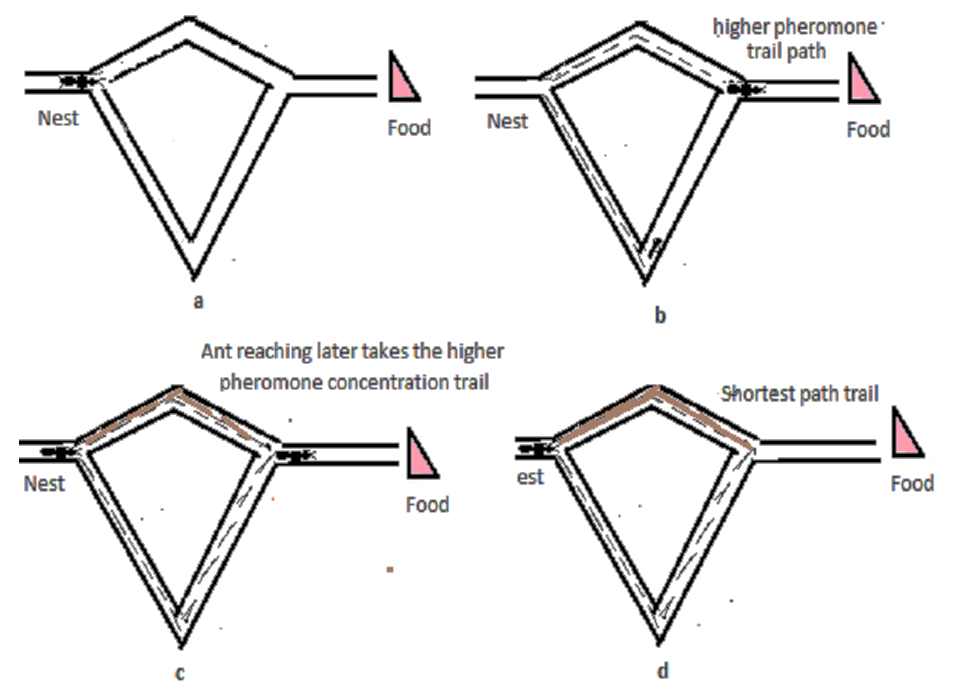
Individual tables for each destination contain vector entries called the pheromone variables, one for each known neighbor node. These pheromone entries at each node indicate the goodness of the path, and hence specify how good it is to take this route via the node to the destination. The pheromone variables are frequently updated according to path quality values as calculated by the ants. Every node has a list of frequently used, available paths along with its estimated measure of quality. This is a result of the repeated and concurrent generation of path-sampling ants. Random hops are taken in a probabilistic manner, based on higher pheromone values. This collective learning behavior is achieved through the individual ants. Every artificial ant is a complex entity, which is autonomous, and updates the pheromone entries thereby participating in stigmergic communication in the network.

**2.1.1.2 Background of ACO**

Ants are creatures of limited intelligence, with limited eyesight and aren’t capable of achieving complex tasks on their own. However in nature they manage amazing feats such as building nests, have a strong defense mechanism, capable of forming bridges, cooperatively carry large items and most importantly they collaborate in an organized manner to find food. A French entomologist named as Pierre-Paul Grasse investigated into the social behavior of insects and found that the ants are capable of reacting to what he referred to as “significant stimuli”, which has signals that can activate reaction in similar genetic species. He observed that the effects of these reactions can act as new significant stimuli for the other insects in the colony. Grasse coined the term Stigmergy (Nagalakshmi and Rakesh 2016).

Stigmergy is a form of self-organization which creates complex explicit structures that have an associated behavior without any particular type of direct communication (Wikipedia). The ants can smell pheromone and whilst choosing their way, in general they choose the paths marked by stronger pheromone concentrations. As more and more ants follow the trail, the trail becomes increasingly more attractive. The binary bridge experiment was done to investigate pheromone trail laying and the behavior of ants done by Goss et.al. (1989). Here in, the nest of a colony of ants and a food source were separated by a diamond-shaped double bridge. Ants were then left free to move between the nest and the food source. The strength of ants which choose one or the other of the two branches was observed over time. As a result of this experiment it was the shortest path between the two available paths that was chosen and there by choosing the natural optimization path in observed that after few minutes’ ants tend to converge on a same path. The branches of the bridges with different length had the majority of ants follow the shortest path due to pheromone concentration in the food-path experiment shown in figure 1 below (Nagalakshmi and Rakesh 2016).

The goal for the artificial ants is to solve more complicated problems than the simple bridge. The artificial ants not only model real ant behavior but also exploit elements of ant stigmergic activities to create a heuristic for difficult problems. The artificial ants need to be flexible towards the changing environment. The real ants lay pheromone on the trip to and from the food source.

 Figure 1: Binary Bridge experiment with different path length (Nagalakshmi and Rakesh 2016).

There is an evaporation of pheromone to prevent one locally optimal solution. However there are problems of the real ant systems like, when deciding complicated paths and ants can become trapped in loops, sometimes shortest path may no longer be the favored path. These have to be addressed by the artificial ants in the algorithms which are inspired by the real ants.

Network routing issues may be addressed using ACO in Mobile ad hoc networks (MANETs). Network characteristics like traffic load and network topology may vary stochastically and in a time varying nature also the distributed nature of network routing in MANETs is well matched to the multiple agent collaborative nature of ACO algorithms (Deepalakshmi and Radhakrishnan 2009). The network of nodes can be represented as a graph in which the vertices correspond to set of nodes which act as routers and the links represent the connections amongst the nodes. To identify efficient routes minimum cost path routes between nodes in a MANET can be done by taking into considering ant algorithms.

The most important part of any network is control routing that strongly affects the overall network performance. Routing deals with issues like maximizing network performance by defining optimal paths and controls the flow of data traffic. In a routing system decision making can be seen to be distributed. Every node forwards the data packets in accordance to the contents of the routing table. A variety of different classes of specific routing can be defined according to the different characteristics of processing, transmission components, traffic pattern and type of performance (Nagalakshmi and Rakesh 2016).

The following is the set of ACO characteristics as used in routing (Nagalakshmi and Rakesh 2016).

• It provides traffic-adaptive and multipath routing.

• Uses both proactive and reactive forms for information computation.

• Make use of stochastic components

• Does not allow local estimates to have effect on larger scale.

• The paths set up are load balanced rather than purely shortest paths.

These characteristics are identified as a result from the application of ACO’s design guidelines along with the use of controlled random experiments. The ants are generated in a controlled manner to actively gather information about the characteristics of a set of paths connecting Source-destination nodes. (Nagalakshmi and Rakesh 2016).

**2.1.1.4 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 below summarizes the main differences between artificial ants and real ants.

|  |  |  |
| --- | --- | --- |
| **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. | 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 (Hazem and Janice 2012)

**Ant Colony Optimization Algorithm**

Meanwhile, some improvements were inserted into the Ant system (AS) such as the introduction of elitist ants into the colony, the ranking of ants, and the bounding of the allowed accumulated pheromone in each path (Marta et al 2012),

The ACO, which is described in Algorithm below, is made of general guidelines for the development of algorithms based on foraging ants to solve combinatorial optimization problems.

Algorithm Pseudo-code for Ant Colony Optimization.

1: *Initialize parameters*

2: *Initialize pheromone trails*

3: *Create ants*

4: ***while*** *Stopping criteria is not reached* ***do***

5: *Let all ants construct their solution*

6: *Update pheromone trails*

7: *Allow Daemon Actions*

8: ***end while***

The main difference from the basic structure of the AS algorithm is the introduction of a Daemon. The daemon can perform problem speciﬁc operations or centralized operations, which use global knowledge of the solutions, thus having a very active and important role in the algorithm.

Note that in contrast to the AS no global knowledge is used since each ant deposits pheromone in its solution despite what the other solutions are like. This is a task that has no equivalence in the nature. The daemon can, for example, control the feasibility of each solution or give an extra pheromone quantity to the best solution found from the beginning of the algorithm or to the best solution in the current iteration. These last operations were already mentioned in previous algorithms but never attributing its responsibility to a main entity in the colony.

Another important feature, frequently used by authors on ant based algorithms is the introduction of Local Search procedures following the construction of the solutions. This is an optional feature that has been proved to be very important in the exploitation of the search space near to good solutions, leading almost always to better performances of the ACO.

**2.1 Ant System**

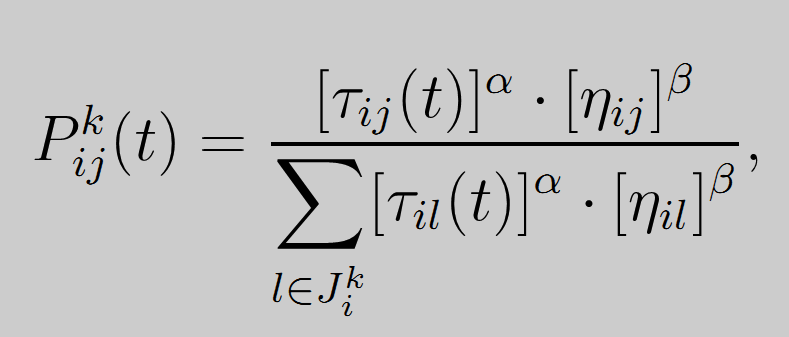
The objective of the travelling salesman problem is to ﬁnd the shortest route between a set of cities, starting and ﬁnishing in the same city, going through all cities without visiting each city more than once. This problem is very easily adapted to the idea of the Ant System due to their similarity in concepts: ﬁnd the shortest path between two points in a graph.

An Ant System (AS) algorithm considers a single ant colony with m artiﬁcial ants cooperating with each other. Before the algorithm starts to run each arc linking two different cities is given a certain quantity of pheromone τ0. This is usually a very small value just enough to ensure that the probability of each arc to be chosen is different from zero. Also, the ants are created.

The algorithm has two main phases, the construction of the tour/solution and the pheromone update. Other important decisions have to be made before the ants can start ﬁnding a solution, such as deﬁning the structure (representation) of the solution, or the initial pheromone quantity to be given to each arc. These questions will be discussed further ahead.

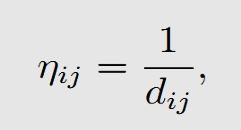
At each iteration, each ant is randomly placed in a city, from the set of n cities. That city will be the starting point of the tour that is to be constructed by the ant. A solution to the TSP can be represented by a set of n consecutive cities. Therefore, at each step of the construction the ant has to choose, with a given probability, the next city to travel to.

This choice is made by using a transition rule, the short expression for random proportional transition rule, that uses a combination of attractiveness of the city, which is given by the heuristic information ηij of the problem, and of the ﬁtness of the move, i.e. past usage, which is given by the pheromone quantity τij . The transition rule quantiﬁes the probability of ant k, positioned at city i, travelling to city j and it is given by:



(1)

where ηij , the heuristic information or visibility of arc (i, j), is the inverse of the distance between city i and city j, i.e.

 (2)

is the set of cities not yet visited by ant k while at city i, and α and β are parameters weighting the relative importance of the pheromone and of the heuristic information, respectively.

Therefore, the closest cities, that is, the ones that the ant can see from where it is standing, will have a higher visibility value, whereas the others will have a lower one.

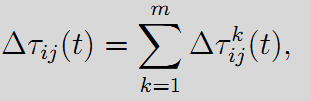
The values α and β are two tunable parameters that weight the pheromone information and the heuristic information on the transition rule.

After building the solutions the pheromone values in the arcs are updated. The update is done in two phases. Just before the ants can deposit pheromone in the arcs of their solution, the algorithm applies an evaporation rate ρ, with ρ ∈ ( 0, 1), to the pheromone present at each arc, see equation (3).

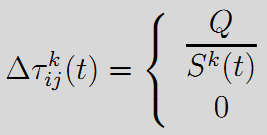
τij(t) = (1 − ρ) · τij(t). (3)

This operation simulates the natural process of evaporation preventing the algorithm from converging too quickly (all ants constructing the same tour) and getting trapped into a local optimum. The value of the evaporation rate indicates the relative importance of the pheromone values from one iteration to the following one. If ρ takes a value near 1, then the pheromone trail will not have a lasting effect, potentiating the exploration of the solutions space, whereas a small value will increase the importance of the pheromone, potentiating the exploitation of the search space near the current solution.

The length Sk of each tour is then calculated and the ants will be allowed to deposit pheromone in every arc of their tour. The pheromone quantity to be deposited in each arc is proportional to the quality of the solution of each ant and to the number of ants to incorporate that arc in its solution, as can be seen in equations (4) and (5).



(4)



if (i, j) belongs to the solution of ant k, (5).

Otherwise,

where Q is a positive proportionality parameter and Sk(t) is the length of the tour constructed by ant k at iteration t. For small problem instances, this update leads to a reduction of the search space thus converging to one where the optimal solution components will have the highest values in the matrix. However, for large instance problems it is known that stagnation is likely to happen, driving the solution to a suboptimal solution rather than to an optimal one. This is why pheromone evaporation is so important.

The previous steps are performed until some stopping criterion is reached, which can be a ﬁxed number of iterations, as was the case, but it can also be the setting of a bound on running time or even the number of solutions evaluated.

The best values for the parameters used in ant algorithms depend both on problem characteristics and on the strategy chosen for searching the solution space. Therefore, before setting values for the parameter, decisions on the search strategy have to be made. Then, the algorithm must be run several times in order to establish the values of the parameters which tend to perform better.

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 ‎ (Kennedy and Eberhart 1995). 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, evolve weights and structure of neural networks, 3D-to-3D biomedical image, control reactive power and voltage, even learning to play games ‎and music composition.

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.

**Birds in Nature**

Vision is considered as the most important sense for flock organization ‎. 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. 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. For example, Pigeons can see 300 degrees without turning their head, and American Woodcocks have, amazingly, the full 360-degree field of view. 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 (Kennedy and Eberhart 1995).

**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 (Kennedy and Eberhart 1995). 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 ‎ (Reynolds, C.W.1987). The pioneering work of Reynolds proposed three simple flocking rules to implement a simulated flocking behaviour of birds:

1. flock centering (flock members attempt to stay close to nearby flock mates by flying in a direction that keeps them closer to the centroid of the nearby flock mates),
2. collision avoidance (flock members avoid collisions with nearby flock mates based on their relative position), and
3. velocity matching (flock members attempt to match velocity with nearby flock mates) ‎

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. 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 (Reynolds, C.W.1987). . 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 ‎ (Reynolds, C.W.1987).

Reynolds rules are also comparable to the evaluation, comparison, and imitation principles of the Adaptive Culture Model in the Social Cognitive Theory (Hazem and Janice 2012).



Figure 4: The flocking behaviour of a group of birds (Xiong, et al 2010).

Particle Swarm Optimization (PSO) is a heuristic optimization technique introduced by Kennedy and Eberhart in 1995 ‎ (Kennedy and Eberhart 1995). It is inspired by the intelligent, experience-sharing, social flocking behaviour of birds that was first simulated on a computer by Craig Reynolds ‎ (Reynolds, C.W.1987), and further studied by the biologist Frank Heppner ‎ (Frank and Grenander, 1990). 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. While ACO solves problems whose search space can be represented as a weighted construction graph. 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. 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).

**The PSO Algorithm**

The original PSO was designed as a global version of the algorithm, 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 ‎ (Hazem and Janice 2012).

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 (Hazem and Janice 2012). The original version of the PSO algorithm is essentially described by the following two simple velocity‖ and position‖ update equations, shown in 6 and 7 respectively.

vid(t+1)= vid(t) + c1 R1(pid(t) – xid(t)) + c2 R2 (pgd(t) – xid(t)) (6)

xid(t+1) = xid(t) + vid(t+1) (7)

Where:

* vid represents the rate of the position change (velocity) of the ith particle in the dth dimension, and t denotes the iteration counter.
* xid represents the position of the ith  particle in the dth  dimension. It is worth noting here that xi is referred to as the ith  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.
* pid represents the historically best position of the ith  particle in the dth dimension (or, the position giving the best ever fitness value attained by xi).
* pgd represents the position of the swarm‘s global best particle (xg) in the dth dimension (or, the position giving the global best fitness value attained by any particle among the entire swarm).
* R1 and R2 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.
* c1 and c2 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.

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 c1 = c2 = 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 c1 + c2 = 4 in some typical applications ‎ . The values of c1 and c2 can remarkably affect the search ability of PSO by biasing the new position of xi toward its historically best position (its own private experiences, Pi), or the globally best position (the swarm‘s overall social experience, Pg):

* High values of c1 and c2 can provide new positions in relatively distant regions of the search space, which often leads to a better global exploration, but it may cause the particles to diverge.
* Small values of c1 and c2 limit the movement of the particles, which generally leads to a more refined local search around the best positions achieved.
* When c1 > c2, the search behaviour will be biased towards particles‘historically best experiences.
* When c1 < c2, the search behaviour will be biased towards the swarm‘s globally best experience.

The velocity update equation in (6) has three main terms:

(i) The first term, vid(t), is sometimes referred to as inertia‖, momentum‖ or habit. It ensures that the velocity of each particle is not changed abruptly, but rather the previous velocity of the particle is taken into consideration. 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, xg, This is because if a particle, xi, 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, pi, will coincide with both the swarm‘s global best position, pg, and its own position vector, xi, in the next iteration (i.e., pi = xi = pg) ‎. Therefore, the effect of last two terms in equation (7) will be no longer there, since in this special case pid(t) – xid(t) = pig(t) – xid(t) = 0, 0, ∀ 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 ‎

(ii) The second term, ( pid(t) – xid(t) ), is the ―cognitive‖ part of the equation that implements a linear attraction towards the historically best position found so far by each particle. This term represents the private-thinking or the self-learning component from each particle‘s flying experience, and is often referred to as “local memory”, “self-knowledge”, “nostalgia” or “remembrance” ‎.

(iii) The third term, ( pgd(t) – xid(t) ), is the ―social‖ part of the equation that implements a linear attraction towards the globally best position ever found by any particle. This term represents the experience-sharing or the group-learning component from the overall swarm‘s flying experience], and is often referred to as “cooperation‖”, “social knowledge‖”, “group knowledge” or “shared information” ‎.

According to the aforementioned equations (6) and (7), the basic flow of the original PSO algorithm can be described as shown below.

**Algorithm: Basic flow of PSO (adapted from ‎ Shi, Y 2004)**

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, Pi, 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 g and its position at Pg.

5) Update the velocities of all the particles.

6) Move each particle to its new position.

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).

**Comparison Between Genetic Algorithm And PSO**

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 ‎ (Goldberg, 1989) drawing inspiration from the evolution metaphor of the Darwinian Theory, and following basic genetics principles. GA employs three operators to propagate its population from one generation to another: Selection, Crossover and Mutation.

1. 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.
2. 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.
3. 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.

**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.
* 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 (c1 and c2) need to be cleverly decided upon in PSO ‎

**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 chooses the path of competition‖, i.e., the convergence is driven through learning from competitive individuals following the survival of the fittest principle. (Hazem and Janice 2012).

**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 (Hazem and Janice 2012).‎.

**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

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.

**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) ‎ (Hazem and Janice 2012)..

**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, plus the population in each iteration of Gas replaces 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 one-way information sharing mechanism (Hazem and Janice 2012).

**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 (Hazem and Janice 2012).

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 (Hazem and Janice 2012).

**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. 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 (Hazem and Janice 2012).

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. For example. It has been shown that the decrease of inertia weight dramatically increases the swarm‘s convergence ‎(Hazem and Janice 2012)..

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 .

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 population, 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.

**Applications of Swarm Intelligence**

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 ‎ (Hazem and Janice 2012).

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 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 ‎ (Hazem and Janice 2012).