**SWARM INTELLIGENCE**

1. **Introduction**

Swarm intelligence is a newly discovered branch of Artificial Intelligence that helps in modelling the cumulative behaviour of social swarms in nature like honey bees, flock of birds and ant colonies. A swarm is a large number of elementary, homogeneous individuals which interact locally among themselves and their environment without any central control resulting in a globally interesting behaviour. Swarm-based algorithms have recently emerged as a family of algorithms inspired by nature and are based on the behaviour of the members of the swarm population. These are contributing a lot in producing cheaper, fast and robust solution to numerous complex problems. Although these swarm individuals are not that sophisticated in themselves, they continuously interact with each other with certain behavioural patterns to achieve necessary tasks for their survival. The social interactions among these agents of the swarm can be either direct or indirect. Direct interactions generally happen through audio or visual contact, like the waggle dance of the honey bees. In indirect interaction, one individual agent responds to the new environment as a result of changes made in the environment by the other individual of the swarm, such as the pheromone trails that the ants deposit on their way while searching for food. This indirect interaction between the agents is referred to as stigmergy, which basically means communication through the environment. Some basic characters of stigmergy are:

* Indirect interaction of agent causing modification of the environment
* Modification of the environment serving as external memory
* Work can be continued by external memory
* Work does not depend on specific agents
* The same, simple behavioural rules can create different designs according to the environmental state.

Self-organization is also one of the important factors in the swarm behaviour. Self-organization is a set of dynamical mechanisms whereby structures appear at the global level of a system from interactions of its lower level components. Four basics of self-organization are given below:

* Activity amplification by positive feedback
* Negative feedback for activity balancing by stabilization and counterbalance
* Amplification of random fluctuation like random walks, errors, randomness
* Multiple interactions

Self-organization also gives rise to a structure emerging from a homogenous start-up state, multi stability that is a number of stable states coexisting and state transition with a significant change of the behaviour of the system. So, majorly there are two principles in the swarm intelligence, namely self-organization based on and stigmergy.

Some of the examples of swarm behaviour in the nature are groups of foraging insects, labour division, building of nests by the social insects, cooperative transportation, collective sorting and clustering. Swarm intelligence is an interesting field for IT. Some of the analogies in the social insects and IT are:

* Distributed system of interacting autonomous agents
* Goals: robustness, optimization and performance
* Decentralized self-organized control and cooperation
* Division of labour and distributed task allocation
* Stigmergy – interacting indirectly

Motivations and methods in biologically inspired Information Technology:

* The analogies in social between social insects and distributed computing.
* Biology provides the key to solution for a lot of computational problems.
* Analogy identification, computer modelled biological mechanisms, adaptation of biological mechanisms for IT applications.

**2. Models of the Swarm Behaviour:**

The swarm behaviour of the social insects like bees, ants and termites, has much more implications than the hives. Swarm Intelligence, is the collective behaviour of the agents of the swarm working individually, each of them responding to the local stimuli without any central supervision. It helps to understand and model phenomena as diverse as clotting of blood, traffic patterns at highways, immune response and gene expression.

Unlike a top-down organization which characterizes various human endeavours, a lot many social species employ the bottom-up approach to achieve their communal goals without any kind of structure for command and control. A swarm of termites is a good example of this, as they show a collective intelligence far exceeding the intelligence levels of any individual insect, which itself has limited capabilities for processing and communication of information.

The swarm exhibits the collective intelligence emerging from the actions of the individuals of the swarm and their responses to the local stimuli from the environment, and more importantly, from the fellow members of the swarm. There is no one in charge. No swarm individual has the bigger picture of the group. But yet in the aggregate, the actions of each insect which it performs locally according to the local stimuli it receives, can ultimately achieve a bigger goal collectively which serves the interests of the community as a whole. As now it turns out, the rules of the biological world can be applied in the computational world as well.

There are applications which demonstrate the fact that a group of small, simple agents with limited decision-making capabilities, intelligence and a communication path to the neighbour swarm-mates can outperform large processors that are centralized. Moreover, a decentralized system has numerous significant advantages over a centralized one, most notable ones being robustness and flexibility.

Principles of Swarm Intelligence have been employed in enormous variety of problem domains like to find optimal routes, optimization problems, image and data analysis, structural optimizations and schedulings. The swarms being computationally modelled provides the further scope of applying them to a wide ranging diverse set of domains that include dynamical systems, bioinformatics, operational research and machine learning, and also in finance and business.

Swarm intelligence models are basically natural swarm systems inspired computational models. Up till now, lot of swarm intelligence models have been proposed and applied successfully in various fields helping find solutions in a new direction to several complex problems. A few examples of the swarm intelligence models are:

* Particle Swarm Optimization
* Bacterial Foraging
* Ant colony optimization
* Artificial Bee Colony
* Artificial Immune System
* Glow-worm Swarm Optimization
* Cat Swarm Optimization

**2.1 Designing Swarm Intelligence systems:** (A three step process)

1. Identifying analogies: in swarm biology and IT systems
2. Understanding development: computer modelling of realistic swarm biology
3. Engineering: model simplification tuning for IT applications

**3. Meta Heuristics**

These evolutionary algorithms, particle swarm optimization, ant colony optimisation and their variants dominate the field of nature inspired meta-heuristics. A few of them are explained below.

**3.1 Stochastic Diffusion Search(SDS):**

Stochastic Diffusion Search, a population-based multiagent algorithm for global search and optimization, is a distributed mode of computation which utilizes the interactions between the simple swarm agents. SDS introduces a new probabilistic approach for finding the solution to the problems of best-fit pattern recognition and matching problems.

SDS provides a solid mathematical framework, which describes the behaviour of the algorithm by investigating its convergence towards the global optimum, allocation of resources, robustness, linear time complexity and criteria for minimal convergence.

**3.1.1 The mining game**

This is metaphor which gives us a simplified high-level description of the behaviour exhibited by the agents in SDS. Here, the mountain ranges are divided into hills which are further divided into smaller regions. A group of miners start off with the only information that there is some gold that has to be found on the hills of a mountain range but no information about its distribution is known. The maximum number of miners should be digging at the hill (the information about which is not known beforehand) with the richest reserves of gold in order to maximize their collective wealth. Miners use a simple Stochastic Diffusion Search for solving this problem.

* In the beginning of the mining process, every miner gets allocated a hill for mining which is termed as his hill hypothesis h.
* Each of the miners everyday get assigned a randomly selected region on the hill for mining.

The probability of a miner to be happy is proportional to the amount of gold that is mined by him at the end of each day. Every evening, each miner who is not happy randomly chooses another miner for communication randomly. If this randomly selected miner is happy, then he shares the location of his hill and then both of them mine at this hill maintaining this as their hill hypothesis. And if the selected miner is not happy, then he selects another hill to mine at random.

This process being isomorphic to SDS, miners will gradually themselves get self-organized to congregate over the hills of the mountain having the highest concentration of gold.

**3.1.2 Algorithm 1:** The Mining Game

//Phase of initialization:

For each miner (agent) {

A hill is randomly allocated (hypothesis) to pick a random region for mining;

}

While (! all miners congregated over the mountain with highest gold concentration) {

//Testing phase:

Hypothesis evaluation: each miner evaluates the amount of gold mined by him;

Each miner is classified as either happy (active) or unhappy (inactive);

//Diffusion phase

For each unhappy miner take up a new hill by either communicating with another random miner. {

If (this selected miner is happy) {

Now both the miners will mine at his location;

}

Else {

No flow of information between the miners;

New hypothesis: the selecting miner chooses another hill randomly;

}

}

}

The further refinement of this metaphor (the mining game) can be done by adding at every location, either of the two assumptions mentioned below:

1. Resources being finite: every time a miner mining an area reduces the amount of gold reserved in the hill.
2. Resources being infinite: a conceptual or theoretical assumption that the amount of gold is potentially infinite.

**3.1.3 SDS Architecture:**

The SDS algorithm initiates an optimization or search by initialization of its population (for instance the miners in the above mining game analogy). In any Stochastic Diffusion Search, a hypothesis h is maintained by every agent, each of which define a possible solution just like in the agent hypothesis identifying a hill in the mining game metaphor. Following two phases are followed after initializing:

* Testing phase: for example, testing the gold availabilities at different locations
* Diffusion phase: for example, congregating and information exchanging

SDS, in the testing phase, performs the testing of the agent hypothesis to check whether it is successful or not by evaluating the partial hypothesis and returning a Boolean value independent of the domain. In the later iterations, contingent on the strategy employed, a successful hypothesis diffuses throughout the entire population and thus all the agents of the population get informed about a potential good solution.

All the agents, in the diffusion phase, interact with some other agent for communicating about a potentially good hypothesis. In the mining game analogy for instance, diffusion takes place in form of the communication of a hill hypothesis.

**3.1.4 Algorithm 2:** SDS Algorithm

Initialization phase: every agent generates an initial hypothesis in the beginning;

While (termination criteria not fulfilled) {

Test phase: evaluation of hypothesis by each agent;

Diffusion phase: each agent deploys a strategy for communication;

Relate phase: random deactivation of active agents having the same hypothesis;

Halt phase: termination criteria is evaluated;

}

**3.1.5 Terminology:**

In the original SDS approach, a group of agents together search for the best possible solution for some optimization problem under consideration. All the possible feasible solutions are grouped together in a set referred to as the solution space S. All the points in S are associated with an objective value. These objective values are taken over from the entire solution space and fed to an objective function f. In simple words, an assumption is made which states that the objective is to minimize the sum of n {0,1}-valued component functions fi:

min ∀s∈S f (s) = min ∀s∈S Ʃ fi (s)     fi: S → {0,1} .

During the operation, a hypothesis is maintained by every agent regarding the best solution to the given problem. No theoretical assumptions are made beforehand about the representation of the hypothesis.

* **Initialize**: In the beginning, initialization of the hypothesis parameters of the agents takes place. There exist various methods for initialization, but their specifications are not required to understand the basic concepts behind the algorithm.
* **Test**: every agent randomly chooses a single component function fi where i∈ {1, 2,…, n}, and carries out the evaluation of its particular hypothesis sh∈ S. then all the agents are categorized into two groups on the basis of the results from the evaluations, namely, active and inactive. An active agent is defined as the one with fi (sh)=1. If fi is allowed to be probabilistic, then there are possibilities of various different evaluations of fi(sh) to result in different outcomes. The following pseudocode describes the testing phase:

For each agent {

Comp-func = select-component-function ();

If (comp-func (agent\_hypothesis) = 0)

Agent\_activity = true;

Else

Agent\_activity = false;

}

* **Diffuse**: in the process of diffusion, each of the active agents picks one agent randomly from the population for communication. If this selected another agent is classified active, then the selecting agent takes up his hypothesis to updates his own, and this is termed as diffusion of information. On the other hand, if this another agent turns out to also inactive, then the selecting agent is allocated a new hypothesis randomly, thus resulting in no flow of information. No communication is started by the active agents in the standard form of the SDS algorithm. Summary of the diffusion is given below in form of a pseudocode:

For each agent {

If (agent\_activity = false) {

Another-agent = select-agent-randomly (all-agents);

If (another-agent\_activity = true)

Agent\_hypothesis = another-agent\_hypothesis;

else

Agent\_hypothesis = select-hypothesis-randomly (all-hypotheses);

}

}

**3.2 Ant Colony Optimization (ACO):**

Every ant algorithm represents a multi-agent system, their agents being referred to as artificial ants. Real ants are the actual inspiration behind the behavioural patterns of their artificial ants. These ant algorithms are one of the most successful examples of the systems harnessing swarm intelligence and they have been employed in a wide range of problems, varying from the classic travelling salesman problem to routing in the telecommunications network.

**3.2.1 The double bridge experiment:**

A colony of Argentine ants cultured in a laboratory is provided with access to a source of food, with the only possible paths from their colony to the food reserve being through two bridges having different lengths. From the experiment, it is observed that after a phase of transition, majority of the ant population follows the shorter path. Autocatalysis (positive feedback) and differential lengths of paths helps explain this emerging of the shortest path selection behaviour. This behaviour is possibly exhibited because of an indirect form of interaction called stigmergy, mediated by making some modifications in the environment locally. If simplified, the basic reason behind this behaviour is that the rate of pheromone deposition is higher on the shorter path as compared to the longer one, which makes the ants select the shorter path more and more until the entire population ends up using this path.

**3.2.2 Ant Colony Optimization Metaheuristics:**

Ant Colony Optimization is an algorithm based on the pheromone lying/ pheromone following behaviour which helps the real ants to find the shortest path between the food source and their nest. In numerous ways, the behaviour of the real ants is simulated by the behaviour of the artificial ants:

* For the reinforcement of the most potential solution components of the construction graphs, pheromone trails are left on the shortest path.
* For the solution construction, the artificial ants move through 10 construction graphs choosing their paths with respect to probabilities, which are determined by the pheromone trails deposited previously.
* These trails of artificial pheromone keep decreasing quickly enough at each iteration, to suffice to stimulate the phenomena of slowly-evaporative pheromone trails

The fitness function of ACO is often formulated implicitly like the cost minimization of components of the solution, that is, the goal of artificial ants is to walk on the construction graph selecting the nodes that result in minimizing the overall cost of the solution path.

**3.2.3 Algorithm 1:** Basic flow of Ant Colony Optimization

1. Representing the solution space using a constructive graph.
2. Setting parameters of ACO and initializing pheromone trails.
3. Generating ant solutions from the walk of each ant on the construction graph guided by the pheromone trails.
4. Updating intensity of the pheromones.
5. Going to step 3 and repeating till the termination criteria is fulfilled or convergence.

The artificial neural network (ANN) is a relatively close paradigm to the Ant Colony Optimization as both of them can be described as a kind of connectionist system having the individual units (like the artificial neurons of ANN or artificial ants of ACO) that follow some particular pattern in their connections.

**The ACO meta-heuristics:**

Setting the parameters, initializing pheromone trails.

Scheduling activities:

1. Construction of ant solution
2. (optional) Daemon actions
3. Updation of pheromones

Accumulation of virtual trails on the path segment.

* Construction of ant solutions:

Probability of an ant moving from a node i to node j is:

pi,j = ((τ α i,j ) (η β i,j ))/∑ (τ α i,j )(η β i,j )

Where,

τi,j = quantity of pheromone on the edge (i,j )

α = parameter for controlling the influence of τi,j

    ηi,j = desirability of edge (i, j), especially 1/di,j

β = parameter for controlling the influence of ηi,j

* Updation of pheromones:

The following equation is used for deciding the quantity of pheromones for updating:

τi,j = (1 – ρ) τi,j+ Δ τi,j

where,

    τi,j = quantity of pheromone on the edge (i,j )

ρ = rate of evaporation of pheromone

    Δ τi,j  = quantity of pheromone deposited, given by the following formula:

Δ τi,jk = {0 otherwise1/L(k) if ant k travels on edge i,j

Where L(k) = cost of the tour (basically length) of the kth ant.

Among the various Ant Colony Optimization metaheuristics that have been proposed, the following are the most successful three: Ant System (AS), Ant Colony System (ACS), and Max-Min Ant System (MMAS).

**3.2.4 Standard ACO Algorithms:**

Some of the important standard ACO algorithms and their variations are briefly explained below.

1. **Ant Systems (AS):**

Ant system is an ant colony algorithm (ACO) which is mainly for the travelling salesman problem. The Ant system was modelled as multi-agent system for the optimization of combinatorics. Its agents are referred to as ants and they use a probabilistic decision rule, while the term pheromone was used to indicate the qualities of the decision variables that are learned. In Ant Systems, ants use the two components, namely problem-dependent heuristic information and pheromone trails, to construct the solutions. These algorithms are frequently modified to improve their efficiency. A lot of improvements have happened on the Ant Systems (AS). The Elitist Strategy for Ant System (EAS) was the first improvement of AS in which every ant which finds a better solution bears a chance of depositing more pheromone.

1. **Ant Colony System (ACS):**

A modification of the AS is Ant Colony System. ACS and AS majorly differ in the form of the decision rule that the ants use during the construction process. It involves the use of local and global update rules for pheromones. The local update of pheromones is done when the solution is being constructed. Global updation of pheromones is done at the end of the construction process. Ants use the pseudo random rule of proportionality: the probability for all ants to move from i to j depends on q, a uniformly distributed random variable (distributed over [0,1]), and on a parameter q0; if q < q0, then the component which maximizes the product is chosen among all the feasible ones.

1. **Max-Min Ant System (MMAS):**

Max-Min Ant System proposed by Stuzle and Hoos is an enhanced version of the Ant System. It provides effective and exploit good solution. MMAS is different from AS in that:

* 1. Pheromone trails are added by only the best ants
  2. The maximum and minimum values of the pheromones (τmax, τmin) are explicitly limited.

All edges are initially set to τmax and when about to stagnate (idle state) then are re-initialised to τmax. It uses pheromone trails to find out the potential solutions in the search space and creates new promising initial solution for the local search. The advantages that the application of MMAS provides inspires us to start investigation further for using it to solve CO problems.

1. **Rank-Based Ant System (ASrank):**

Another modified version of Ant System as proposed by Bullnheimer at al, is ASrank or Rank-based Ant System. In the ASrank, fitness is used to rank all the possible solutions. The amounts of pheromones deposited for all solutions is weighted and then used as a criterion as more pheromone is deposited in the more optimal solutions than the less optimal ones.

1. **Recursive Ant Colony Optimization (RACO):**

RACO is a recursive version of the AS that divides the whole search space into several subspaces, then solving the objective on these subspaces. A few of the results are finally chosen from the best ones obtained from all the subdomains, and then these are prompted for the following levels. The subspaces or subdomain which correspond to the selected results are then subdivided further and this process is repeated until a result of the required precision is obtained. This algorithm is already tested and known to work well on the ill-posed geological inversion problems.

* 1. **Particle Swarm Optimization (PSO):**

PSO has successfully been applied and used to track dynamic systems, evolve weights and structure of neural networks, registering 3D-to-3D biomedical image, analysing human tremor, controlling reactive power and voltage, also learn playing games and composition of music. Particle Swarm Optimization is based on the sociological behaviour associated with the bird flocking. This phenomenon of bird flocking is a natural observation, that birds can fly in very large groups without colliding for extended long distances, by making an effort to use their ability to maintain an optimum distance between each other and their neighbours. This kind of interactions are based on the “nearest neighbour principle” in which birds of the swarm follow certain flocking rules to adjust their motion (position and velocity) according to only their own nearest neighbours, without any central coordination.

Reynolds proposed three simple flocking rules to implement a simulated flocking behaviour of birds:

1. Cohesion: flock centring (flock members fly in a direction which keeps them closer to the centroid of the neighbouring birds of the flock in an effort to stay close to the nearby flock mates)
2. Separation: collision avoidance (each member of the flock avoids collision with nearby flock mates based on its relative position), and
3. Alignment: velocity matching (flock members continuously try to match their velocity with their neighbours).

Particle swarm optimization algorithm is a population-based search strategy which uses a set of flying particles having velocities which are adjusted according to their performance up till now, and also the nearby particles in the search space. PSO is used to solve problems whose solutions can be represented as a set of points in a n-dimensional solution space. The term “particle” means the member of the population which is fundamentally described as the swarm position in the n-dimensional solution space.

All the particles are given a velocity vector, which represents the speed of respective particles in every dimension, and then are set into motion. Every particle has a memory for storing the information about his historically best position (typically, the best position that it ever attains in the search space up till now, that is also referred to as its experience).

* + 1. **PSO Basic Algorithm:**

At each step, the experience of each agent is communicated to a part or the whole swarm, thus promoting the progress of the swarm overall in the most promising area which are known so far in the search space. Therefore, at each iteration, the moving particles continuously evaluate their current position according to the fitness function of the problem that has to be optimized and compare their current fitness to best position up till now of themselves as well as to the other swarm members, either locally within the neighbourhood (like the local version of PSO) or globally across the whole swarm (like the global version of the PSO). For each particle, if it’s current position is better than its historically best position, the particle keeps updating its experience and adjusting its velocity to imitate the global best particle of the swarm or a neighbour particle whose current position is a better solution (basically, the local best particle in the neighbourhood), by moving towards it. At the end in each iteration, the index of the global best particle of the swarm or the local superior neighbour, is updated by changing it to the most recently updated position of any of the particle in the entire swarm (or, within a predetermined neighbourhood topology) if it happens to have a better position than the current position of the global best particle (or, the local best particle in the neighbourhood).

**3.3.2 Algorithm 1:** PSO Algorithm

**procedure** Particle Swarm Optimization {

For each particle {

initialize xi, vi and xbesti;

}

**While** (not termination condition) {

**For** each particle i {

Evaluate objective function;

Update xbesti

}

**For** each i {

g = neighbour with best xbesti;

Calculate vi using g;

Update xi = xi + vi;

Evaluate objective function;

Update xbesti;

}

}

}

According to the basic algorithm that Kennedy and Eberhart proposed:

           xik – particle position

           vik – particle velocity

           pik – best “remembered” individual particle position

           pgk – best “remembered” swarm position

c1, c2 – cognitive and social parameters

           r1, r2 – random numbers between 0 and 1

The position update rule for the individual particles is as follows:

           xik +1 = xik + vik +1

With the velocity being calculated using the following formula:

           vik +1 = vik + c1r1 (pik – xik) + c2r2 (pgk – xik).

Searching in the Particle Swarm Optimization is carried out by a collection of particles which are updated from iteration to iteration. Each particle moves towards the direction of its previously best position and the global best position in the swarm, which in turn leads them to the optimal solution.

**Fig. PSO algorithm flowchart**

Initialization of Particles

Evaluate fitness value for every particle

Is current fitness value better than pBest?

Current fitness assigned as new pBest

Keep previous pBest

pBest value of best particle is assigned to gBest

Calculate velocities for all particles

Updating data value of all particles by using their velocity

Target or maximum epochs reached?

No

**3.4 Biogeography Based Optimization (BBO):**

BBO is an evolutionary algorithm based on the migratory behaviour of species between habitats. BBO is known to be a powerful search technique because it makes use of the migration exploration and exploiting strategies. The BBO being one of the fastest-growing and nature-inspired algorithm, is successful for cracking a solution for the practical optimization problems. BBO is advantageous in a lot of ways as it is simple, flexible and computationally efficient, as well as has a stochastic nature, thus having no need for the derivatives of the objective function.

HSI (Habitat Sustainability Index) is a means used to differentiate between the islands which are more suitable for habitation than the others. Some of the features that define habitability are temperature, rainfall, topography and diversity of vegetation, etc. Biogeography is the nature’s way of distributing the species and interestingly, analogous to the solution of general problem. Refining this analogy, an island having a high HSI corresponds to a good solution, and a poor solution is represented by a low HSI island. High HSI solutions resist change more than low HSI solutions. Poor solutions take up a lot of good features from good solutions. As the sustainability of the habitat improves or the HSI increases:

* The count of species increases
* Emigration increases (number of species exiting the habitat increases)
* Immigration decreases (lesser species enter the habitat)

We begin by taking an optimization problem and a population of the candidate solutions. Here, we will refer to candidate solutions as “solutions” for simplicity. All the solutions consist of features, also called independent variables. A good solution represents a biological habitat which is well suited for life, while a poor solution corresponds to a habitat having a low HSI. A solution with high fitness value tend to share features with the other solutions since there are high chances for features emigrating from high-fitness solutions and immigrating to the ones having low finesses. Solutions having low fitness values have a tendency to accept shared features from the better solutions. As with any other EA, BBO has two steps, namely, information sharing and mutation. Information sharing of BBO is implemented with migration.

The basic methodology for Biogeography based optimization:

1. The problem is initialized with a set of solutions
2. HSI (fitness) is calculated for all solution
3. For each solution, S, λ and μ are calculated
4. Habitats (migration) are modified based on
5. Mutate on the basis of probability
6. Typically, elitism is implemented
7. If required then for next iteration go to step 2

**3.4.1 Algorithm 1:** A generation of the standard BBO algorithm, where N is the size of the population, y is the whole population of candidate solutions, yk is the kth candidate solution and yk (s) is the sth feature of yk.

For each solution yk

set emigration rate equal to μk proportional to fitness with μk ∈ [0,1];

}

For each solution yk

set emigration rate equal to λk = 1-μk;

}

z←y;

For each solution zk (k = 1 to N) {

           For each solution feature index s

                          Use λk to decide whether to immigrate to zk. or not;

                          If immigration decided {

Use {μk} to select the emigrating solution, i.e., yj;

zk (s)←yk (s);

                          }

                          Next solution feature

                          Decide whether to mutate.

           Next solution

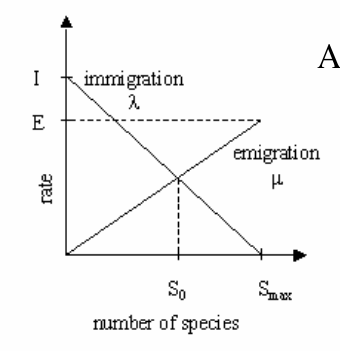
           y←z

}

Migration in BBO is probabilistic. The features are shared stochastically using the migration rate of each solution. For each solution yk, it is decided one at a time that whether each of its features have to be migrated or not, using its immigration rate λk. If the stochastic decision made favours the immigration, then one more decision is made randomly; the solution to be emigrated yj is stochastically selected on the basis of rate of emigration μj. Migration is denoted as:

                                         yk (s) ← yj(s)

where, s is a solution feature index. The probability of migration depends on deterministic curves.



Mutation being a probabilistic function can make modifications in the features of the solutions and also can be implemented in any other EA. the diversification of the population is the major purpose of migration. The above algorithm demonstrates only one generation of BBO. Before any solution is replaced, whole population undergoes mutation and migration in which the use of temporary population z is required.

**4. Hybrid Computational Intelligence**

The major characters defining a computationally intelligent system are dealing with numerical or low-level data only, having components for pattern recognition, not using knowledge in AI sense, and in addition to all this, when it exhibits the following properties:

* Computationally adaptive
* Tolerance to computational fault
* Speed that approaches human-like turnaround
* Gives error rates approximating human performance

The hybrid intelligent systems have shown an increased rate of popularity. This was due to these systems extensively succeeding in a wide range of complex problem of the real world. Another factor that contributed to this was the enhanced capabilities of the advancing computational technology. One of the reasons for this success has to do with the synergy derived by the computationally intelligent components like neural networks, fuzzy logic, genetic algorithm, machine learning, or other intelligent techniques and algorithms. Every partial methodology provides hybrid systems with methods of searching and complementary reasoning, allowing to solve complex problem using the empirical data and the domain knowledge.  Intelligent systems are the ones which exhibit the properties of adaptation, self-maintenance, preservation of information and enhancement in complexity, as well as use other means to achieve these objectives.

During the last two decades, the entire domain of AI is dominated by four major streams as follows:

1. Machine learning and data mining
2. Neural networks
3. Fuzzy sets and soft computing
4. Genetic algorithms and evolutionary computations (also include nature inspired algorithm).

**4.1 Hybridization of the Artificial Bee Colony (ABC) and Particle Swarm Optimization (PSO) algorithms:**

The aim behind the hybrid algorithm is to effectively mix the components from both Artificial Bee Colony and Particle Swarm Optimization algorithms which will provide us with the algorithm that easily solves separable problems as ABC while at the same time having a rotationally invariant behaviour as SPSO. ABC works great on the separable functions because its update equation updates only a single problem variable at a time and after that, a new problem is re-evaluated.

**4.1.1 Mixing Approach:**

Addition of the ABC component to SPSO is done after the main loop. Every time a particle i is selected and the ABC update equation is used to produce a new candidate solution v, and this process is repeated for m trials. This is done after selection of some particle k as a neighbour on a random basis and a random problem variable j, hence producing v as follows:

           vl = pbestil,  l not equal to j,

           vl = pbestil + Φil x ( pbestil -  pbestkl ),      l = j.

If the new candidate solution v has a better fitness then it replaces pbesti. The step by step procedure in this hybrid approach is shown below:

**4.1.2 Algorithm:** The ABC-PSO algorithm

Evaluate the Max\_Function\_Evaluations, c1, c2, n, m, w;

Initialization of the swarm;

Evaluation of the swarm;

Max\_iterations = num\_particles / max\_function\_evaluations;

Iter\_number = 1;

While (iter\_number <= max\_iterations) {

For each particle i {

update vi;

update xi;

if (f(pbesti) <= f(xi)) {

pbesti = xi;

                         }

           }

           update gbest;

                for m trials {

           select a particle i to be improved;

                          select a different random particle k;

                          select a random problem variable j;

                          apply ABC update rule to pbesti as in equation above;

                          update pbesti and gbest;

}

Iter\_number = iter\_number + 1;

}

return gbest;

**4.2 Hybridization of Particle Swarm Optimization (PSO) with adaptive Genetic Algorithm (GA) operators**

Two particles are selected using a probabilistic approach in proportion to their fitnesses, and the algorithm performs a crossover between them. Then depending on the adaptive crossover probability Cp, every position between the first and the maximum is swapped. The original inspiration behind this crossover operator was the pbest crossover introduced by Chen, which makes an attempt in increasing the explorative search in PSO. the movement of the particle stops and the convergence sets in when the distance between the particles and their pbest approaches zero. Hence, the pbest crossover has a tendency to reduce the convergence of PSO by moving the particle’s position away from its pbest. In contrast to Chen’s original pbest crossover which employs periodic crossover, its currently implemented version applies crossover depending on an adaptive crossover probability.

**4.2.1 Algorithm1:** Hybrid PSO and GA algorithm

Each particle is initialized with a random position and velocity vector;

For each particle {

Calculate fitness for the position of the particle (x);

If (fitness, i.e., x > pbest) {

Pbest = x;

}

}

While (! max\_iteration\_completed) {

Set gbest;

Selection;

Adaptive crossover;

Adaptive mutation;

Update particles;

}

**5. Case Studies:**

**5.1 Swarm intelligence for permutation optimization: A case study of n-queens problem**

* **Hypothesis**: Metaheuristics approaches (concretely PSO) can find successful schedules of traffic lights for heterogeneous urban scenarios in reasonable time.
* **Proposal**: A Swarm Intelligence approach (Particle Swarm Optimization) coupled with SUMO (Microscopic Simulator of Urban Mobility), to automatically search quasi-optimal solutions (traffic lights schedules).
* A common program controls each of the traffic light situated on the same intersection. E.g., the combination of colour states during a cycle period is kept valid (it has to follow specific traffic rules of that particular intersection).
* **Main objective:** finding an optimized cycle program (CP) for all the TLs located in a given area. CPs are referred to as the time span (or phase duration) of a set of TLs, in a given intersection, keeping their colour states.
* As in real schedules, CPs are designed for established time periods with certain vehicle densities and speeds (rush hours, nocturne periods, etc.)
* **Solution encoding**: vector of integers in which every element is representing a time span or phase duration of one state of TLs in the intersection (SUMO structure of CPs). Adjacent intersections have to be also coordinated for the improvement of the global flow of vehicles.
* **Optimization algorithm (PSO) with Simulation procedure (SUMO):**

**5.1.1 Algorithm 1:** Pseudo code for Standard PSO 2007 for OCP

swarm\_initialization ()

While (g < maximum\_iterations) {

For each particle xig {

                      bng= bestNeighbourSelection (xig, n)

                      vig+1=updateVelocity (w, vig, xg, φ1, pg, φ2, bng)

                      xig+1=Q (updatePosition (xig, vig+1))

        evaluate (xig+1)

                      pig+1=update(pig)

}

}

* **Performance Comparison:** in general, PSO yields the best results, followed by DE, SCPG and Random Search. Statistically, each distribution pair (Wilcoxon) obtained significant differences (α=0.05), excepting for DE and RAND with (500 vehicles). The higher the traffic density higher the traffic density, the greater the benefits of using PSO.  The global trip time goes on decreasing as the PSO starts approaching the termination condition (improvement of 17.45% with respect to SCPG solutions). The number of vehicles reaching their destination increases along with the search progress. A low traffic density can be observed in PSOs’ solutions, but traffic jams appeared in SCPG ones.
* **Conclusion**: A Swarm Intelligent approach was proposed that could be coupled with the SUMO traffic simulator and can be used to successfully find the cycle programs of the traffic lights. Focus mainly was on a metropolitan area of Córdoba city.

**5.2 Enhancing the urban road traffic with Swarm Intelligence: A case study of Córdoba city downtown**

* **Problem**: The n-queens problem is the kind of problem which consist of placing n queens on a square (n cross n) chessboard, such that they don’t attack each other. Here, this problem is handled by using PSO.
* This problem is used as a benchmark generation of permutations, backtracking algorithms, the problem of constraint satisfaction, the divide and conquer paradigm, genetic algorithms (GA) and the neural networks (NN). The problem has found its practical application in a lot of areas such as data compression, air traffic control, data and message routing, VLSI testing, modern communication systems, optical parallel processing and computer task scheduling.
* There are 3 variations of the n-queens problem:
  + Find all the solutions
  + Find a family of solutions
  + Find just one solution
* **Dealing with Permutations:**

All the particles represent a corresponding solution in the parameter space in the traditional Particle Swarm Optimization. Each particle is encoded as a string of positions that all represent a multi-dimensional space. A great benefit of PSO is that the velocity and the particle are performed are updated independently in each dimension because all the dimensions are independent of each other. However, this particular property is not applicable to the problems of permutation as the elements in considerations are not independent of each other. There is a possibility that more than one position gets assigned the same value, thus breaking the permutation rule. So, a new strategy is proposed for the updation of the particles, to eliminates or minimize the conflicts. In the traditional Particle Swarm Optimization algorithm, the particle updation is a distance measure as the velocity is added to the particle on each dimension. The larger the velocity is, the more distant areas the particle will explore. Similarly, in the permutation scenario, the new velocity is a measure of the possibility that the particle changes. The likelihood of the particle to change to a new permutation sequence increases as the velocity increases. The update formula for velocity remains the same. However, the velocities are limited to the absolute values as they represent only the difference between the particles. The procedure of updation the particles is modified as follows:

* + Normalizing the velocity to a range of 0 to 1 by dividing it by the maximum range of the particle.
  + Then all positions randomly determine that if there is a swap with a probability determined by the velocity.
  + In case a swap is required, the position will be set to the value of the same position in nBest by swapping values.

**5.2.1 Algorithm:** Procedure for PSO

Initialization of population

while (termination criterion not fulfilled) {

For each particle {

Calculate fitness\_value;

if (fitness\_value > best fitness value, i.e., pBest in history) {

pBest = fitness\_value;

}

}

nBest = the particle with the best fitness\_value of all topological particles

For each particle {

Calculate new velocity, i.e., Vnew:

Vnew = w \* Vold + c1 \* rand () \* (PpBest - X) + c2 \* rand () \* (PpBest – X);

Update the position of the particle:

Xnew = Xold + Vnew;

}

}

* **Conclusion:** the purpose of the study was to determine the success of application of PSO in handling the parameter sets of permutation. The n-queens problem was used to validate the performance and feasibility of the new technique. The performance of PSO was well comparable to the Genetic algorithms and it demonstrated the effectiveness of the Particle Swarm Optimization algorithm to handle the n-queens problem.