Special Issue - 2017
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
ICCCS - 2017 Conference Proceedings

Swarm Intelligence

Aditya Alok Jha
CSE (3rd Year)
HMR Institute of Technology and Management
Hamidpur, New Delhi, India

Romi Chaudhary
CSE (3rd Year)
HMR Institute of Technology and Management
Hamidpur, New Delhi, India

***Abstract*— Swarms are systems that consist of many individuals that are organized and coordinated by principles of decentralized control, indirect communication, and self-organization. In this paper some of the real life problems that can be (or are being) solved with swarm intelligence have been discussed. Types and applications of Swarm Intelligence have also been discussed briefly. Swarm Robotics, which may be simply defined as an approach to the coordination of multi-robots systems which consist of large numbers of mostly simple physical robots to coordinate in order to perform some task that requires accuracy, precision, efficiency, and demands a higher rate of success with lesser effort, has been described and a minimalist flocking algorithm has been discussed for the implementation of the same as well. To conclude, the future aspects of Swarm Robotics has also been brought into the limelight.**

Keywords—Swarm; ACO; PCO; TSP; 0-1Knapsack; Crowd Control; PSSD; Bee ColonyOptimization; Bat Algorithm; Cuckoo search; Swarm Robotics; Flocking

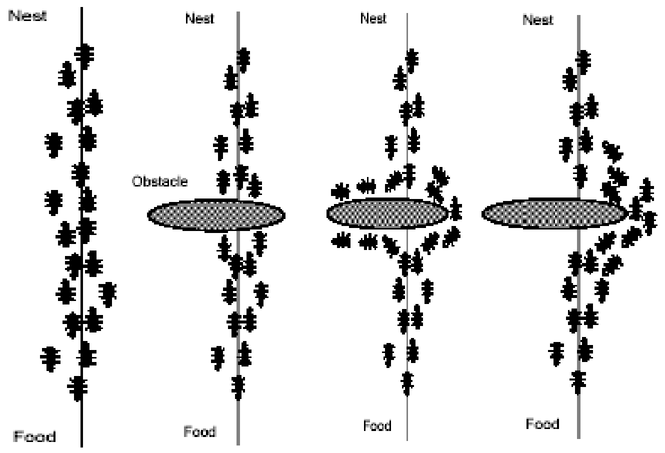
I. INTRODUCTION TO SWARM INTELLIGENCE

Swarm Intelligence is the collective behavior of decentralized, self-organized systems, natural or artificial. The concept is employed in work on artificial intelligence. The expression was introduced by Gerardo Beni and Jing Wang in 1989, in the context of cellular robotic systems.

II. TYPES OF SWARM INTELLIGENCE

A. Ant Colony Optimization(ACO):

Ant Colony Optimization was initially proposed by Marco Doringo in 1992 in his PhD thesis, and has been originally used to solve discrete optimization problems in the late 1980s. ACO draws inspiration from the social behavior of ant colonies. It is natural observation that ants seek a shortest path between their colony and a source of food. The following section presents some details about ants in nature, and shows how these relatively unsophisticated insects can cooperatively interact together to perform complex tasks necessary for their survival. Most ants live on the ground and make use of the soil surface to leave pheromone trails, which can be followed by other ants on their way to search for food sources. Ants that happened to pick the shortest route to food will be the fastest to return to the nest, and will reinforce this shortest route by depositing food trail pheromone on their way back to the nest. This route will gradually attract other ants to follow, and as more ants follow the route, it becomes more attractive to other ants. The overall result is that when one ant finds a good (i.e., short) path from the colony to a food Source, other ants are more likely to follow that path, and positive feedback eventually leads to all the ants following a single path. The idea of the ant colony algorithm is to mimic this behavior with "simulated ants" walking around the graph representing the problem to solve.

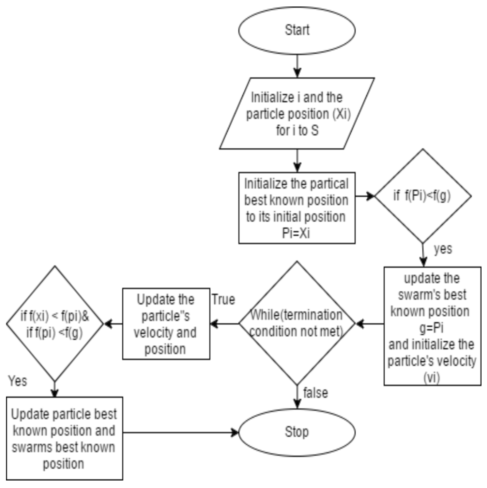


B. Particle Swarm Optimization(PCO):

Particle swarm optimization or PSO is a global optimization algorithm for dealing with problems in which a best solution can be represented as a point or surface in an n-dimensional space. Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy inspired by the emergent motion of a flock of birds that is flying around to search for food. The motion of each single bird in the flock is influenced by potential for aging places the bird itself has seen so far and also by the behavior of other birds in the flock. Similarly, in a PSO algorithm, a swarm of individuals, that are referred to as particles, tries to find the global minimum (or maximum) of a given function. It solves a problem by having a population of candidate solutions, here dubbed particle, and moving these particles around in the search space according to simple mathematical formula over the particle's position and velocity.

Each particle's movement is influenced by its local best known position, but is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.
The Algorithm for the PSO can be defined as:

```
for each particle i = 1, ..., S do
    Initialize the particle's position with a uniformly distributed random vector: $X_i \sim U(B_{lo}, B_{up})$
    Initialize the particle's best known position to its initial position: $P_i \leftarrow X_i$
    if $f(p_i) < f(g)$ then
        update the swarm's best known position: $g \leftarrow P_i$
    Initialize the particle's velocity: $v_i \sim U(-|B_{up}-B_{lo}|, |B_{up}-B_{lo}|)$
while a termination criterion is not met do:
    for each particle i = 1, ..., S do
        for each dimension d = 1, ..., n do
            Pick random numbers: $A_p, A_g \sim U(0,1)$
            Update the particle's velocity: $v_{i,d} \leftarrow \omega\, v_{i,d} + \varphi_p A_p (p_{i,d}-x_{i,d}) + \varphi_g A_g (g_d-X_{i,d})$
        Update the particle's position: $x_i \leftarrow x_i + v_i$
        if $f(x_i) < f(p_i)$ then
            Update the particle's best known position: $p_i \leftarrow x_i$
            if $f(p_i) < f(g)$ then
                Update the swarm's best known position: $g \leftarrow p_i$.
```


Start
Initialize i and the particle position (Xi) for i to S
Initialize the partical best known position to its initial position Pi=Xi
if f(Pi)<f(g)
yes
update the swarm's best known position g=Pi and initialize the particle's velocity (vi)
While(termination condition not met)
True
Update the particle"s velocity and position
if f(xi) < f(pi)& if f(pi) <f(g)
Yes
Update particle best known position and swarms best known position
false
Stop

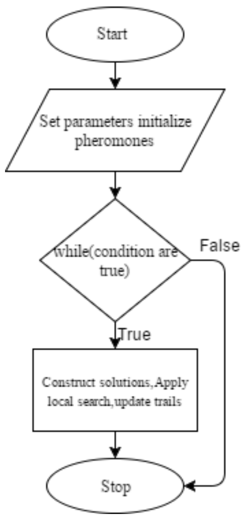
III. APPLICATIONS OF SWARM INTELLIGENCE

A. *ACO Applications:*

1) *Traveling Salesman Problem:*

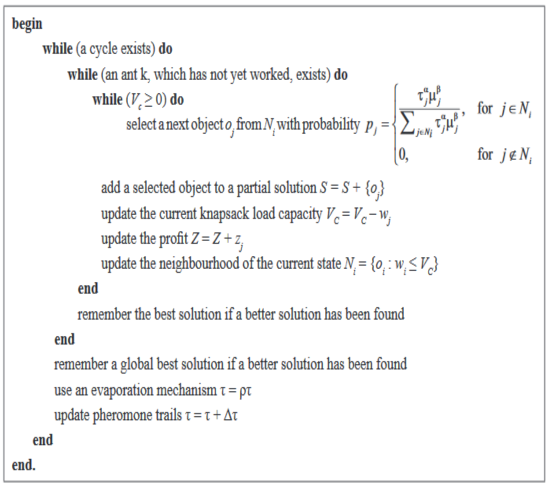
The traveling salesman problem (TSP) can be stated very simply: a salesman spends his time visiting *n* cities (or nodes) cyclically. In one tour he visits each city just once, and finishes up where he started. The question is: in what order should he visit the cities to minimize the distance traveled?
The idea was published in the early 90s for the first time. The base of this simulation was two artificial connections between the ant hill and a food source. For practical use of ACO, it was necessary to project virtual ants. It was important to set their properties. These properties help virtual ants to scan the graph and find the shortest tour. Virtual ants do not move continuously; they move in jumps, which means that, after a time unit, they will always be in another graph node. The absolved path is saved in ant memory. The created cycles are detected in ant memory. In the next tour, the ant decides on the base of pheromones power. Just because the property of pheromone evaporation, pheromones on shortest edges are stronger, because of the fact that the ant goes across these edges faster. Based on these facts we can mathematically describe the behavior of the virtual ants by, the algorithm for the ACO can be defined as:

```
Start
Set parameters, initialize pheromone trails
while (termination condition not met) do
    Construct_Solutions
    Apply_Local_Search  (% optional)
    Update_Trails
end while
end
```


Start
Set parameters initialize pheromones
while(condition are true)
False
True
Construct solutions,Apply local search,update trails
Stop

2) *0-1 Knapsack problem:*

Many optimization problems in decision-making can be presented as the 0-1 Knapsack Problem (KP). The 0-1 Knapsack Problem consists of loading objects in to a knapsack in such a way that the obtained total profit of all objects included in the knapsack is maximum and the sum of the weights of all packed objects does not exceed the total knapsack load capacity. Each object can be loaded or not loaded into the knapsack; this is the 0-1 decision concerning object loading. There are also other versions of this problem such as the Multi-dimensional 0-1 Knapsack Problem or the Multiple 0-1 Knapsack Problem. In ant algorithms a colony of artificial ants is looking for a good solution to the investigated problem. The pseudo-code of the ACO algorithm is presented as procedure 1. Each artificial ant constructs an entire solution to the problem in a certain number of steps; at each step there is an intermediate solution, a partial solution or a state. In each step, each ant k goes from one state i to another state j and thus constructs a new intermediate solution. At the end, the entire solution will have been obtained in a certain number of steps. At each step, each ant k takes into consideration a set of feasible expansions to its current state and moves to one of these in probability. This set of feasible expansions is called a neighborhood. At each state i there is a partial solution Si of the Knapsack Problem; each ant selects the next object Oi from the set Ni of available objects, goes to the next state j and adds this selected object to a partial solution Sj in order to construct, at the end of the algorithm operation, the set of objects S constitutes a solution to the 0-1 Knapsack Problem.



```
begin
    while (a cycle exists) do
        while (an ant k, which has not yet worked, exists) do
            while ($V_c \geq 0$) do
                select a next object $o_j$ from $N_i$ with probability $p_j = \begin{cases} \frac{\tau_j^{\alpha}\mu_j^{\beta}}{\sum_{j \in N_i} \tau_j^{\alpha}\mu_j^{\beta}}, & \text{for } j \in N_i \\ 0, & \text{for } j \notin N_i \end{cases}$
                add a selected object to a partial solution $S = S + \{o_j\}$
                update the current knapsack load capacity $V_C = V_C - w_j$
                update the profit $Z = Z + z_j$
                update the neighbourhood of the current state $N_i = \{o_i : w_i \leq V_C\}$
            end
            remember the best solution if a better solution has been found
        end
        remember a global best solution if a better solution has been found
        use an evaporation mechanism $\tau = \rho\tau$
        update pheromone trails $\tau = \tau + \Delta\tau$
    end
end.
```

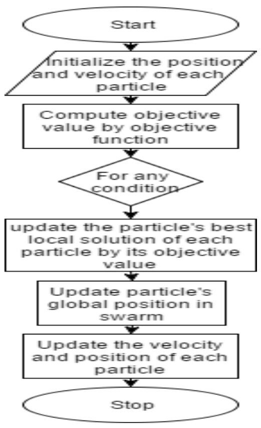
B. PCO Applications:

1) Crowd Control:

Crowd control focuses on creating a realistic smooth and flexible motion for virtual human beings by utilizing the computational facilities provided in Particle swarm optimization (PSO). A person can be considered as a particle, which would like to find a way to reach the best solution. Although PSO does possess some characteristics of the crowd behavior, it is still incompatible with the use for crowd control. Firstly, the particle in PSO is absolutely free to fly through everywhere in the given multidimensional space. However, the environment for a crowd may have obstacles, and the pedestrians in the crowd must avoid collisions, including the collision with the given obstacles and the collision with the fellow pedestrians, where other pedestrians can be considered as dynamic obstacles. These dynamic obstacles are not predictable and may appear and disappear in the environment at any moment.

Steps followed by Particle swarm optimization:

- Initialize the position and the velocity of each particle.
- Compute the objective value by the objective function.
- Update the particle's best local solution of each particle by its objective value.
- Update the particle's global solution in the swarm
- Update the velocity and the position of each particle


Start
Initialize the position and velocity of each particle
Compute objective value by objective function
For any condition
update the particle's best local solution of each particle by its objective value
Update particle's global position in swarm
Update the velocity and position of each particle
Stop

Particle swarm optimization (PSO) is an optimization paradigm proposed in the field of evolutionary computation for finding the global optimum in the search space. The concept of PSO is easy to comprehend, and the mechanism is easy to implement. The ability of PSO to reach the position of the optimum creates the possibility to automatically generate non-deterministic paths of virtual human beings from one specified position to another. On the other hand, if the target is the best position, the movement of a person is a process to find a walk-able path to the destination. For these essential reasons, we propose the model to work with the original PSO for path creation.

2) PSSD (Particle Swarm Inspired Underwater Sensor Deployment) :

PSO is a centralized intelligent searching method. Inspired by the operation mechanism of particle swarm systems, There is a distributively realizable underwater sensor deployment algorithm. Sensors correspond to the particles of PSO. Sensors moving and covering events is similar to particles searching for solutions .

```
S ← Randomly deploy sensors in the monitoring space;
 for step ←1 to I do
 N event(sj) ← detect1(sj)
/* detect the number of the events covered by sj itself */;
 N neighbor(sj) ← detect2(sj)
/* detect the number of the adjacent sensors */;
N near(sj) ← detect3(sj)
/* detect the number of the near sensors */;
if N neighbor(sj) > 0 then
find the best adjacent sensor s*;
 if N event(s*) > N event(sj) and N near(s*) < δ (s*) then
/* follow the gbest */;
 move towards s*;
end
else find the best position of sj during its moving ( xˆ j );
 if xˆ j ! = x j(t) then
/*follow the pbest */;
move towards xˆ j;
 else
 move randomly;
 end
 end
end
```

3) Bee colony optimization:

A honeybee colony typically consists of about 20,000 to 80,000 bees with one queen and a few hundred male bees or drones, with the rest being female worker bees. The workers can be scouts, onlookers, guards, or nectar collectors. A scout communicates to other workers through a waggle dance that it has found a new nectar source. An algorithm that mimics the main characteristics of the bees' foraging behavior—the artificial bee colony algorithm—was developed in 2005.The algorithm divides bees into employed, scout, and onlooker bees; codes the nectar source's location as a solution vector; and links the nectar amount with the landscape of objectives. Although this description is somewhat simplistic, it captures the main characteristics of foraging behavior. It has been used to solve unconstrained numerical optimization problems and constrained optimization problems as well as to train neural networks.

4) Bat algorithm:

Most of the 800+ micro bat species use echolocation for navigation, which involves emitting about 10 to 20 ultra sonic bursts per second. Each burst lasts only a few thousandths of a second, has a frequency of 20 to 200 kHz (humans can hear at most a 20-kHz burst), and can be as loud as 120 dB (the noise level of a jumbo jet taking off). When a bat finds an insect and is homing in on its prey, the pulse emission rate can accelerate to 200 pulses per second with a higher frequency. Echolocation allows the bats to more accurately gauge a flying insect's size, position, range, speed, and direction. The bat algorithm, developed in 2010, uses characteristics of pulse emission and frequency tuning16 and considers the bat's location as a solution vector in the search space. The frequency tuning lets the bat explore the search space on a larger scale, while the speedup of the pulse emission focuses on the neighborhood of local promising solutions. Among the whole bat group, there is a global best solution, and other bats tend to swarm toward it. Consequently, convergence is relatively rapid, controlled by frequency tuning, pulse emission, and loudness. The bat algorithm has been applied in many real-world applications such as engineering optimization, training neural networks, image processing, and solving the TSP.

5) Cuckoo search:

Some cuckoo species, such as Old world cuckoos, engage in brooding parasitism, in which the cuckoo lays eggs in the nest of a host bird such as a warbler. The eggs then hatch and the host bird raises the cuckoo chicks. Because cuckoos are adept at mimicry, the texture, color, and size of the cuckoo's eggs look very similar to those of the host birds' eggs. Even so, some host birds can recognize cuckoo eggs and then get rid of them or abandon the nest, creating a kind of evolutionary arms race between the two species. The cuckoo search algorithm, developed in 2009, considers a cuckoo's egg as a solution vector and the nesting field as the search space. There is some evidence that both the cuckoo's and host bird's flight paths can obey Lévy flights—flights with occasional long jumps followed by many local random steps—which makes the search more effective over a large region. The similarity of eggs can be converted into the similarity of solutions, which helps the iterative search process reach convergence, and the discovery probability aids in global exploration. The cuckoo search algorithm has been successfully applied to engineering optimization and image processing problems.

IV. SWARM INTELLIGENCE AND ROBOTICS

In nature, vast groups of individuals cooperate and assemble to create highly complex global behavior through local interactions -- from multi-cellular organisms to complex animal structures such as army ants, bivouacs and flocks of birds. Swarm robotics is an approach to the coordination of multi-robots systems which consist of large numbers of mostly simple physical robots. It is supposed that a desired collective behavior emerges from the interactions between the robots and interactions of robots with the environment. This approach emerged on the grounds of artificial swarm intelligence as well as the biological studies of insects, ants and other fields in nature, where swarm behavior occurs. Researchers have designed tiny robots, inspired by ants, bees, and cells, envisioned to work together in large swarms or as programmable materials. Nevertheless, there still exists a substantial gap between the conceptual designs and the realized systems. Creating engineered systems with similar abilities poses challenges in the design of both algorithm and physical systems that can co-operate at such scale. There is a vast body of work on algorithms meant to control collectives of hundreds or even thousands of robots, however, for reasons such as cost, time, or complexity, they are validated in simulation only, or on a group of a few 10s of robots. The research of swarm robotics is to study the design of robots, their physical body and their controlling behavior. It is inspired but not limited by the emergent behavior observed in social insects called swarm intelligence. Relatively simple individual rules can produce a large set of complex swarm behavior. A key-component is the communication between the members of the group that build a system of constant feedback. The swarm behavior involves constant change of individuals in cooperation with others, as well as the behavior of the whole group. The two other similar fields of study which more or less have the same team structure and almost the same goals are multi-robot exploration and multi-robot coverage.

Unlike distributed robotic system in general, swarm robotics emphasizes on the large number of robots, and promotes scalability for instance by using only local communication. That local communication for example can be achieved by wireless transmission systems, like radio frequency or infrared. In the recent study of Harvard university the swarm robots called Kobots have been designed in which 1000's of robots can mimic the behavior of natural swarms. Each robot has the basic capabilities required for an autonomous swarm robot (programmable controller, basic locomotion, and local communication), but is made with low-cost parts and is mostly assembled by an automated process. These swarm robots can be investigate further to do special tasks which involves artificial intelligence. They are now only involves collective behavior although some censors for the movements and decentralized mechanism is also implemented on those robots. Study started with the making of termites like robots that could build structures much larger than themselves like the spinifex termite Nasutitermes triodiae in northern Australia, which is 10m high, except they will build the structures according to us. In the starting of the project there were two main problems faced by the researchers first is making even one individual that is capable of doing what a single termite can do like build structures and move , second problem arises is how to make those limited robots to collaborate with each other so that they can achieve the intelligence that a single individual can't do, the only thing could be done as to make the robots capable of reacting to the environmental happenings as the termites do not talk to each other, they co-operate with each other without talking. The task was to make the robots which might see the building pattern and react to that and decide what to build, moreover not only that what they build must also influence others. By doing so the robots can communicate with each others in the environment through a channel and they can work together to design and construct objects and even repair the same by looking the building patterns. The robots are made such that they move and climb over the stairs and build stairs further on and then climb over those stairs again and so on hence creating a structure. Now in order to do this, different sensors are incorporated in the robots like distance sensors, pattern sensors, sensors of balance, touch sensors. The robots are also made to carry objects to make structures. To do the collective work they are programmed to make a particular pattern with no knowledge of how many robots are working on the structure and how much the structure is completed so that if something is destroyed or removed the robots will automatically build the structure again. The next step is to make the smaller sized robots which will look like ants and can do big things like moving the beams, creating complex structures etc. The robots now have their own computer systems which can be programmed to do tasks like blinking flash light like fireflies. Moreover, the robots can synchronize with each other to create a self-organizing structure. Other tasks like pattern formation can be done by recreating the programs in the robots. The plus point of these robots is that they are programmed identically but they don't behave identical, there are variations in their behavior like in the natural systems. They can form the patterns with the existing codes while at the same time, none of them knowing what the role they are going to play, they know the goal ,they can communicate with 10 neighboring robots ,they have no leaders, no overhead cameras but together they can achieve the goal. The research is going with the goal of making the robots small making them work as a single entity implementing them in the computer systems that can behave like swarms. The main idea behind the development of swarm robotics is to solve the problems which cannot be solved by individual talent to ensure greater possibility of solving the tasks. Apart from that some others application of robot swarms include tasks that demand for miniaturization (nanorobotics, microbotics) like distributed sensing tasks in micro-machinery or the human body. One of the most promising uses of swarm robotics is in disaster rescue missions. Swarms of robots of different sizes could be sent to places rescue workers can't reach safely to detect the presence of life via infra-red sensors. On the other hand, swarm robotics can be suited to tasks that demand cheap designs, for instance mining tasks or agricultural foraging tasks. Also some artists use swarm robotic techniques to realize new forms of interactive art. More controversially, swarms can be used in military to form an autonomous army. Recently, the U.S. Naval forces have tested a swarm of autonomous boats that can steer and take offensive actions by themselves. The boats are unmanned and can be fitted with any kind of kit to deter and destroy enemy vessels. Most efforts have focused on relatively small groups of machines. However, a swarm consisting of 1,024 individual robots was demonstrated by Harvard in 2014, the largest to date. Another large set of applications may be solved using swarms of micro aerial vehicles, which are also broadly investigated nowadays. In comparison with the pioneering studies of swarms of flying robots using precise motion capture systems in laboratory conditions, current systems enable to control teams of micro aerial vehicles in outdoor environment using GNSS systems (such as GPS) or even stabilize them using on-board localization systems in GPS denied environment. Swarms of micro aerial vehicles have been already tested in tasks of autonomous surveillance, plume tracking, and reconnaissance in a compact phalanx. Besides, numerous works on cooperative swarms of unmanned ground and aerial vehicles have been conducted with target applications of cooperative environment monitoring, convoy protection, and moving target localization and tracking.

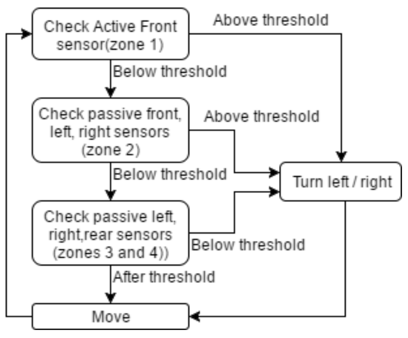
V. MATERIAL & METHODS

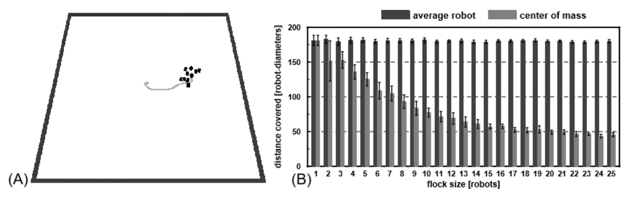
A. *Flocking Algorithm*

Each robot in the swarm periodically emits IR-pulses. The robots then react (move straight, turn left or turn right) depending on information from their active and passive IR-sensors. These sensors are polled periodically and the returned values are then checked against predefined thresholds in a simple subsumption architecture (Fig. 1A). First, the active IR-value for the front sensor is polled to find out whether there is an obstacle in front. If the value for the reflected IR-light is above a certain threshold, the robot turns away in a random direction. This is the basic collision avoidance of our robots. If there are no objects in its way, the robot checks the passive IR-values of all sensors. If the front, left or right sensor is above a certain threshold, the robot turns away from what is presumably another robot which is too close. This rule is usually referred to as the separation rule in flocking algorithms. If there is no other robot too close, the robot checks the passive IR-values of its left, right and rear sensors. For every sensor that returns a value that is above the environmental IR-light threshold but below the threshold which defines the maximally desired distance to another robot in that sector, the robot performs a basic vector addidtion and adds up all turns. It then decides to turn in a direction depending on whether there were more left or more right turns. Robots in the rear zone trigger a random turn reaction. This rule is usually referred to as the cohesion rule in flocking algorithms. The third rule in flocking algorithms is usually the alignment rule which generates the common direction of movement in a flock. Since we wanted our

Fig. 1. A: Simple subsumption architecture depicting the flocking algorithm. The first decision results in collision avoidance, the second decision results in robot separation and the third decision results in flock cohesion and emergent alignment. Simplified depiction of the perceived IR-values of other objects (reflected active IR) or flock mates (passive IR) dependent on their distance to the robot. Thresholds and the resulting zones for a robot with 4 IR-sensors.

algorithm to be as simple as possible we wanted to exclude complex communication or image recognition procedures and implemented a method which generates emergent alignment. To achieve this we adjusted the thresholds for the cohesion rule so that robots tend to follow other robots. This is done by simply shifting the threshold for the rear sensor more outwards in comparison to the thresholds for the left and right sensors. Depending on the position and heading of two approaching robots, one robot will be behind the other robot. When both robots move, the robot behind will turn towards the robot in front before the robot in front reacts and turns around. This creates a leader robot and a follower robot, purely by chance. These two robots will then move around in the arena without separating. If the path of these two robots is blocked by an obstacle or another robot joins the flock, the arrangement can change instantly. If two robots approach frontally, they will avoid each other, only to turn back to each other shortly after, which can create a deadlock situation. To prevent such situations, we implemented a random-turn reaction which means that robots will randomly turn either left or right when avoiding other robots in front.


Check Active Front sensor(zone 1)
Above threshold
Below threshold
Check passive front, left, right sensors (zone 2)
Above threshold
Below threshold
Turn left / right
Check passive left, right,rear sensors (zones 3 and 4))
Below threshold
After threshold
Move



VI. CONCLUSION AND FUTURE APPLICATIONS

Swarm Robotics has a very vast field of application in the present day world and as per its adaptability, versatility, efficiency in solving day to day problems, and its developing nature is taken into considerations, it would be safe to say, in near future Swarm Robotics would take over most of the tasks that require accuracy, precision, efficiency, and demand a higher rate of success with lesser effort. Some of the obvious implementations would be in the military, rescue operations, location and monitoring of remote areas, development of Modular space vehicles, Disaster planning, Risk planning and analysis, Medication and Critical operations, and much more.

ACKNOWLEDGMENT

This work was supported partly by the mentor(s) assigned to us by the institution itself.

REFERENCES

[1] https://en.m.wikipedia.org/wiki/Swarm_intelligence
[2] https://suw.biblos.pk.edu.pl/downloadResource&mId=1139440
[3] http://www.ef.uns.ac.rs/mis/archive-pdf/2011%20-%20No4/MIS2011_4_2.pdf
[4] https://en.wikipedia.org/wiki/Particle_swarm_optimization
[5] http://ijcsmc.com/docs/papers/May2013/V2I5201397.pdf
[6] Xin-She Yang, Suash Deb and Simon Fong, “From Swarm Intelligence to Metaheuristics: Nature-Inspired Optimization Algorithms”
[7] Christoph Moeslinger, Thomas Schmickl, and Karl Crailsheim, “A Minimalist Flocking Algorithm for Swarm Robots”