IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 6, Ver. II (Nov. - Dec. 2016), PP 59-71
www.iosrjournals.org

Primal-Dual Asynchronous Particle Swarm Optimization (*pdAPSO*) Algorithm For Self-Organized Flocking of Swarm Robots

Emmanuel Gbenga Dada
Department of Computer Engineering, Faculty of Engineering, University of Maiduguri, P.M.B 1069, Maiduguri, Borno State, Nigeria.

Abstract: *This paper proposed a hybrid PSO algorithm that combines the Primal-Dual method with APSO algorithm to address the problem of swarm robotics flocking motion. This algorithm combines the explorative ability of APSO with the exploitative capacity of the Primal Dual Interior Point Method. We hypothesize that the fusion of the two algorithms (APSO and Primal Dual) offers a robust prospect of preventing premature convergence of robots, and also make sure that the robots are not stuck in their local minimal. We did a comparison of the performances of the total iteration for pdAPSO, PSO, APSO and Primal Dual algorithms as the robots flock from the centre of the search space to the various zones (z1, z2, z3, and z4). In three (3) out of the four (4) cases, the pdAPSO proves to be more effective for the flocking of the robots than the PSO, APSO and Primal Dual algorithms for the fifty (50) simulations that was done. The results of our simulation gives a clear evidence of the efficacy of the pdAPSO algorithms. The hybrid algorithm contributed to the field of swarm robotics by providing novel algorithm that possess a flocking capability attained under suitable parameter values that is relatively robust and produces effective self-organized flocking in constrained environments.*
Keywords: *Flocking, Asynchronous Particle Swarm Optimization (A-PSO), Swarm Robots, Interior Point Method, Primal-Dual, gbest, and lbest.*

I. Introduction

The Swarm robotics was defined by Sahin [1] as the study of ways through which a vast number of simple inexpensive agents can be made to work together to bring about a preferred cooperative behaviour through communications at the local neighbourhoods among robots as well as between the robots and their surroundings. The attributes of a swarm robotic system includes: huge number of independent robots; ability to detect and transfer information from one robot to the other within the same local neighbourhood; they are decentralised and independent of global information; and cooperative behavior can be attained via spontaneous formation of spatiotemporal structure, and communications among the robots and between the robot and their surroundings. Typically, the problem synonym with the field of swarm robotics is managing and directing the movement of considerable number of robots to carry out a common mission. This type of task is normally impracticable, demanding and laborious for a particular set of robots to accomplish.

The inspiration of swarm robots is fundamentally drawn from the study of behaviour of animals like the flock of birds, herd of cattle, and school of fish. According to [1] and [2], the performance of the swarm at the global level will be largely influenced by the performance of the individual agent at the local level. Some of the traits of swarm robotics that have been extensively investigated are convergence, foraging [1], pattern formation [4], flocking, aggregation and segregation [3], box-pushing [2], cooperative mapping [5], soccer tournaments [6], site preparation [7], and sorting [8]. From the list of the various attributes of swarms above, flocking is the most attractive; where practical applications in areas such as search and rescue [37], system for monitoring behaviour or changing information [38], system for acquiring data which is used for measuring physical phenomenon [8], and networks of small low cost sensors [8] can be realized.

In swarm robotics, the problem of flocking entails directing a set of robots to move to a specific direction and converge to a target in an unfamiliar location. The robots that made up the swarm are required to accomplish this purpose as they are adjusting to their environments. Researchers have developed a number of control algorithms for flocking of swarm robots in the last few years. Reynolds, C.W., [3] was the one of the pioneers. He implemented a computer simulation to demonstrate the movement of a flock of birds, known as boids. This is based on the principal that the global behaviour of the boids as a whole is a direct outcome of the behaviour of every single participant (obeying certain instructions). There are three basic behavioural guidelines that every agent must have. They include: separation, cohesion, and alignment. While separation will prevent the agents from bumping into one another, cohesion will make the agents to be united, and alignment will cause them to move with a collective speed. In progression to Reynolds' work, e-boids was developed by Ward et al. [11] to model the flocking behaviour of shoal of fishes. A down-to-earth flocking model of independent agents was developed by Vicsek et al. [4], in which all the agents were assigned an unchanging fixed velocity. Jadbabie

et al. [5] went further to postulate hypothetical elucidations on the stated characteristics of the model presented by Vicsek. Some other researchers, Yang et al. [6], did a similar work and proposed some rules for flocking in an anonymous environment with barriers in which the closest point on the barrier is considered a virtual agent.

The flocking behavior (in a distributed environment) based on artificial potential field (APF) was then investigated by Kim et al [7]. They proposed a group of systematic procedures that can be used to create functions in the APF so that the agents in the swarm will not be trapped in the local minima. The focus of all the aforementioned research work is based on collision avoidance among the robots without considering the flocking. Genetic programming was implemented by Spector et al. [12] in a virtual environment to create collective behaviour for agents that hover in the swarm. Dorigo, Maniezzo, and Colorni [13] proposed a novel robotic system that is made up of a group of simple mobile robots that are capable of associating and disassociating from one other when subjected to changes (in what they are applied to and the environment where they operate). A robotic system that models the behaviour of physical ants that have the ability to locate the quickest route from one source of food to the subsequent one without any visible sign called pheromone was proposed by Payton et al. [14]. They also have the capability to quickly acclimatize to any environment variations that may require looking for a new quickest path if they encounter a new obstacle in the old one.

The challenge with many of the robotic systems mentioned above lies in the fact that they are not fully centralized. Moreover, there are there are only a few real-world applications from the flocking system described in literature though the theoretical work that have been done on swarm robotic flocking have attained substantial success in the past decade. Therefore, is there is the need to develop more swarm robotic flocking techniques that will serve as a suitable means to bridge the research gap between theory and practice.

This paper proposed a hybridized swarm intelligence based algorithm, and a numerical optimization algorithm i.e., a hybrid of Primal-Dual Interior-Point method and Asynchronous Particle Swarm Optimization (APSO), to attain the best collective performance for considerable volume of swarm robots. The advantage of the novel hybrid Primal-Dual-APSO organization architecture is based in the truth that the flocking can be done in real time, it is decentralized, and inherently scalable because global communication is irrelevant in this matter. Decisions can be made by any of the agent using the local information available to them. The design is simple, scalable, adjustable, and has the ability to recover from catastrophic failure without disrupting its operations.

II. Problem Statement

In this paper we assume that our swarm system consists of n entirely independent and identical (homogenous) robots. The robots in the swarm are individually designated by R_1, $R_2 \ldots R_n$ and they are represented as mobile points in two-dimensional space. The robots are given a neighborhood coordinate structure and they have limited capacity to perceive nearby robots. Moreover, the system is decentralized and there is no open interaction between the robots. The principal axis which defines the spatial locations of the environment is taken as the local axis of each of the robots. In our simulation, we used a point to represent each of the robot's two-dimensional space.

The velocity of the robot is updated at regular intervals and a maximum velocity is assigned to the robots. In addition, the robots flock in real time independently without any influence from other robots in the swarm. We are focusing on the convergence of the swarm to a specific point in the search space, and also the flocking patterns of the robots in the swarm. The Primal-Dual algorithm will calculate the positions, and afterwards, the APSO algorithm will gain control of the flocking movement. The APSO algorithm will guide the flocking of the swarm (that comprises of N agents) to converge towards a global minimum and then flock around in the coordinate system. The flocking is done in real time during our simulation.

III. Primal-Dual-APSO (pdAPSO) Algorithm for Flocking of Swarm Robots

The Interior-Point Methods (IPMs) are efficient algorithm for solving nonlinear optimization problems. The IPMs having constrains that are active at the current point, are regarded as the most robust algorithms for solving large-scale nonlinear optimization problems. Despite its efficiency, the algorithm suffers several problems such as how to handle of non-convexity, the procedure for making the barrier constraint up to date is cumbersome despite the existence of nonlinearities, and the need to ensure progress toward the solution.

The primal-dual interior-point (PDIP) algorithm is an excellent example of an algorithm that uses the constraint-reduction methods. Mehrotra [18] in his research work developed the Mehrotra's Predictor-Corrector PDIP algorithm, which has been executed in almost all the interior-point software suite for solving both linear and convex-conic problems [19]. Wright [23] observed that the primal-dual methods are a new class of interior-point methods that have find practical application in solving large-scale nonlinear optimization problems. Primal-dual methods, contrary to what we have in the traditional primal method, evaluate both the primal variables x and dual Lagrange multipliers λ relating to the constraints concurrently. The disconcerted Karush-

Kuhn-Tucker (KKT) equations below can be solved using the precise primal-dual solution (x^*_μ, λ^*_μ) at a given parameter μ,

$$\begin{cases} \nabla F(x) - C^T\lambda = 0 \\ \lambda_i C_i(x) = \mu, i = 1, \dots, m \end{cases}$$

with the constraint $(C(x), \lambda) \geq 0$.

The Newton's algorithm and line search approach are employed to recursively solve any primal or primal-dual sub-problems for a given μ value according to Boyd and Vandenberghe [20]. Feasibility and convergence is enforced in the algorithm by carefully selecting the size of step in the iteration. One of the prevalent ways to do this is to appropriately reduce the merit function used in gauging degree of advance to the solution. The dual variables of the primal dual can be protected by using $F\mu$ as a function that can incorporate the primal and dual variables [24]; and simultaneously measures the harmony between data and the fitting model for a particular choice of the parameters [22]. The major setback of the barrier functions was explained by Emilie, Saïd & Jérôme [21] and some measures taken to overcome the challenges of the barrier functions.

Primal-dual algorithm can efficiently handle large linear programming problems (i.e., the bigger the problem size the more noticeable the efficiency of the primal-dual algorithm). The algorithm is not susceptible to degradation and the number of iterations does not depend on the number of vertices in the feasible search space. Primal-dual algorithm uses considerably less iteration compared to the simplex method. The algorithm is able to get the ideal solution for a linear programming problem in not more than 100 iterations irrespective of the huge number of variables involved in nearly all its implementations.

The algorithm however has some disadvantages that include its inability to detect the possibility of having unbounded status of the problem (to a certain extent the primal-dual algorithm can be tagged as incomplete). Some researchers have however been able to handle this problem through the use of model that is undiversified in nature [23] and [24]. In addition, the computational cost of each iteration in primal-dual algorithm is higher than that of the simplex algorithm. Finally, the primal steps have the inclination of producing inferior steps that defile the boundaries $s > 0$ and $z > 0$ extensively, causing the progress to dwindle.

Contrary to the way the standard PSO operates, the Asynchronous PSO (APSO), after evaluating the fitness of the swarm, the velocity and position of particles are updated as soon as it finishes calculating their fitness. From literature, it has been discovered that APSO performs better and converges faster than standard PSO and SPSO (Synchronous PSO) [33]. The SPSO algorithm have the advantage of quick convergence and good result. Carlisle and Dozier in [33] however observed that the synchronous update is costly as the first particle evaluated will be redundant for some time since it has to wait for other particles to be evaluated before it can progress to another position and continue exploring the search domain. Some other researchers concluded from their work that APSO provides the best accuracy at the expense of computational time. We took advantage of the explorative capacity of APSO, and the exploitative capability of the Primal-Dual Interior-Point algorithm [29]. In the one hand, the latter is key in ensuring that searching of every segment for a specific area pertaining the solution (to give a dependable approximate value of the global optimal). On the other, the former is essential in directing the search towards the best solutions [28]. According to Carl Damon Laird [17], the state-of-the-art Interior-Point algorithm have gained popularity as the chosen approach for providing solution to large scale linear programming problems. They are however limited due to their inability to solve problems that are unstable in nature. This is because contemporary Interior-Point algorithm might not be able to cope with the increasing need of the large number of constraints.

APSO which is a variant of PSO is a stochastic population based algorithm as explained by Del Valle et. al. [15] that works on the optimization of a candidate solution (or particle) centered on a given performance measure. The first PSO algorithm was proposed by Kennedy and Eberhart [16]. This algorithm was based on the social behaviour exemplified by a flock of bird, a school of fish, and herds of animals. The algorithm uses a set of candidates called particles that undergo gradual changes through collaboration and contest among the particles from one generation to the other. PSO is suitable for optimization problems because of its effectiveness, robustness, simplicity and extreme ease of implementation without the need of cumbersome derivative calculations. Kwang and Jong-Bae [26] observed that PSO algorithm is not derived from another algorithm.The ease of implementation makes it easy for it to be used for solving scientific and engineering problems. It makes use of few parameters which have little effect on the results when compared to other optimization algorithms. Ajith and Amit [27] opined that PSO is theoretically straightforward, and there is no sophisticated computation required in PSO algorithm. The time taken for the convergence of the PSO algorithm and getting the optimum value is relatively quick. When compared to other optimization methods, the initial start points of the PSO algorithm does not really influenced its final output. However, as discussed by Selvi and Umarani [25], the shortcoming of PSO algorithms have to do with premature convergence, inability to solve

dynamic optimization problems, and failure of some of the particles to escape been trapped in the local minimal. PSO algorithm and many of its variants suffers from the partial optimism, which degrades the regulation of its speed and direction.

In PSO, (X_i) represent the position of an agent, and (V_i) the velocity of the agent. The agent's number is i, where ($i = 1,\ldots,N$), and N is the number of agents in the swarm. The i^{th} agent is represented as $X_I = (X_{i1}, X_{i2},\ldots X_{iN})$. While the velocity is the rate at which the next position is changing with respect to the current position. $V_I = (V_{i1}, V_{i2},\ldots V_{iN})$ represents the velocity for the agent i. At the start of the algorithm, initial numerical values of the position and velocity of the agents are assigned randomly. Equations (1) and (2) will then update the position and velocity of the agents for subsequent iterations during the search process.

$$v_{i,m}^{(t+1)} = w * v_{i,m}^{(t)} + c_1 * rand1() * (pbest_{i,m} - x_{i,m}^{(t)} + c_2 * rand2() * (gbest_m - x_{i,m}^{(t)}) \quad (1)$$

$$x_{i,m}^{(t+1)} = x_{i,m}^{(t)} + v_{i,m}^{(t+1)} \quad (2)$$

According to Shi and Eberhart [9], to avert eruption, the value $v_{i,m}^{(t+1)}$ is fixed at $\pm v_{max}$. This is because the value of v_{max} will be too large if the search range is very wide. If v_{max} is too small, the scope of the search will be excessively limited, thereby forcing the agents to support local exploration. "w" is the inertia weight (constriction factor); this regulates the algorithm searching properties. It was suggested by Shi and Eberhart [9] that it is better to start with a large inertia value (a more global search) that will be dynamically reduced towards the end of the optimization (a more local search). The use of small inertia weight usually guarantees quick convergence as the time spent on the exploration of the global space is little [10]. The inclusion of w in the equation is to provide equilibrium between the global and local search capability of the agents. The positive constant c_1 and c_2 represent the cognitive scaling and social scaling factors which according to Kennedy, Eberhart and Shi [30] are set to the value of 2. Both the cognitive and social scaling factors assist the PSO to successfully build the local bests into the global best according to Pinar and Erkan [31]. The stochastic variable rand1() and rand2() has the Uniform distribution. These random variables are independent functions that provide energy to the agents. The best position found so far by the agent is represented as $pbest_{i,m}$. The best position attained by the neighboring agents is represented as $gbest_m$.

Unlike what we have in the standard PSO, the Asynchronous PSO (APSO) works by updating the velocity and position of particles immediately it finish updating the fitness of the swarm [31]. It computes the fitness using partial or limited information about the neighbourhood. This results into varieties in the swarm since some of the information is from the previous iteration while some is from the current iteration. The asynchronous PSO will be very beneficial in solving swarm robotic problems as the robots will have the capacity to navigate the search space uninterruptedly using the current information it is having without any need to wait for the entire swarm [10]. According to Basterrechea, and Perez [32], it have been successfully proved by some researchers from their experimental findings that the performance of the APSO is better than that of the standard PSO and the SPSO [33 - 36], [31] and [10]. This claim was countered by [34] in their research findings when they concluded that there is no significant difference between the performance of APSO and SPSO [34]. The algorithm for the Asynchronous PSO (APSO) is in the Fig. 1 below. The importance of the Primal-Dual algorithm becomes more pronounced as the size of the agents in the swarm rises. The fusion of Primal-Dual into APSO will make the hybrid architecture to become feasible and efficient in a large scale multiple interacting agents' environment.

Step 1: Initialize iteration count, particles with randomly chosen positions, and velocities within the limits of the search space.
Step 2: Initialize the number of primal variables (n), the number of constraints (m), and the total number of primal-dual optimization variables (nv). Initialize the Lagrange multipliers. Initialize the second-order information.
Step 3: Primal Dual phase activated
Step 4: Reset the random number generator
Step 5:Initialize some of the algorithm parameters (such as The maximum centering parameter, The maximum forcing number, Minimum barrier parameter, Maximum step size, Minimum step size, Granularity of backtracking search, Amount of actual decrease we will accept in line search).
Step 6: Compute the responses of the unperturbed Karush-Kuhn-Tucker optimality conditions
Step 7: Check for convergence
Step 8: Update the BFGS approximation to the Hessian of the objective.
Step 9: Find Solution to perturbed KKT system.
Step 10: Do Backtracking line search.
Step 11: Compute the response of the merit function and the directional gradient at the current point and search direction.
Step 12: Compute the candidate point, the constraints, and the response of the objective function and merit function at the candidate point.
Step 13: Stop backtracking search if we've found a candidate point that sufficiently decreases the merit function and satisfies all the constraints.
Step 14: Decrease the step size if candidate point does not meet our criteria
Step 15: Compute the response of the merit function at (x, z).
Step 16: PSO phase activated
Step 17: Get primal dual points boundary values
Step 18: Get primal dual values for particles position
Step 18.1: Particles best positions = particles positions
Step 18.2: Particles velocity = particles dimension
Step 18.3: Get particles best value
Step 18.4: Best val hist = zeros
Step 18.5: particles positions = random particles dimension
Step 19: Main loop starts here
Step 19.1: For iteration = 1 to iteration number
Step 19.2: Loop through every single particle
Step 19.3: for temp = 1: particles
Step 20: Update the position of an individual particle
Step 20.1: Particles positions = particles positions + particles velocity
Step 21: Evaluate the objective function for each particle
Step 21.1: If objective function < particles best value
Step 22: Update position, (Best position).
Step 22.1: Particles best positions = particles positions
Step 22.2: Update best value so far (Particles best value = objective function)
Step 22.3: End
Step 23: Best function value calculation
Step 23.1: Compute Gbest val = min (particles best value);
Step 23.1: Compute Best val hist = gbest;
Step 24: Velocity component update for an individual particle
Step 24.1: Compute particles velocity = c1 * particles velocity + c2 * rand (particles dim) *particles best positions – particles positions + c2*rand (particles dim) * (particles best positions) – particles positions
Step 24.1: End
Step 25: End

Figure 1: The algorithm for *pdAPSO*

IV. Primal-Dual-APSO (pdAPSO) Algorithm Implementation

To In one of our previous work, we proposed a new algorithm called Primal-Dual PSO (*pdPSO*) in [39]. In the bid to improve the performance of our algorithm, we developed the Primal-Dual APSO (*pdAPSO*) and used the algorithm to solve flocking problem in swarm robotics [40]. Having discovered from our findings in [41] that APSO have a superior performance compared to the conventional PSO, our intention was to present a hybridized APSO and Primal Dual algorithm which will perform better other earlier variants of PSO that have been developed.

At the beginning of simulation, there is a haphazard allocation of agents A^i ($0<i\leq N$) in the environment to be explored. The new algorithm works by first starting the agents' positions randomly. Then, the agents are directed to the Primal-Dual method, which gives us its initial optimization result after some number of iterations. The result of the Primal-Dual optimization is then feed into APSO, which creates a perturbation in the population and also maintain diversity in the population until there is either convergence to the global optimal or the termination criteria is reached. During the iterations that are performed in the optimization, the agents carry out shift in the search space from one point to the other (which have connection to behaviour evolutions in space). The cognitive and social scaling factors of the APSO are very important parameters for determining the optimal global behaviors of the agents in the swarm.

4.1 Experimental Setup

In our experiments, we applied the Primal-Dual-APSO algorithm to solving the flocking problem of swarm robotics. The Primal Dual algorithm will run until its tolerance is achieved on the objective function, and then the APSO is runned until its tolerance is achieved. From our experimental design, the search space is assumed to have a centre (c), then four (4) zones e.g. z1, z2, z3, and z4. For each experiment carried out, we want to know the number of iteration it takes the particles to converge to the centre (c), and the number of iteration it takes the particles to flock from c to z1, z2, z3, and z4. The different coordinates for different points that we used are as follows: Z1 (-30, 30), Z2 (-30, -30), Z3 (30, 30), Z4 (30, -30). The zones Z1, Z2, Z3 and Z4 are all points in the Cartesian coordinate plane. Our aim is that for each experiment carried out, we want to know the number of iteration it takes the particles to converge to any given point (p), and the number of iteration it takes the particles to flock from p to z1, z2, z3, and z4.

We used our Primal-Dual-APSO to make the particles to converge to the center point C(0,0) first, i.e., the particles will all be concentrated to a single point. Once they are there, we changed the target point to any zone that we want (Z1, Z2, Z3 or Z4) automatically in the script, and all the points that were converged to C(0,0), will start moving towards the zone in the form of a flock but will eventually converge again to Z1 in the form of a single point. We set the termination and convergence criteria for the particles. When the objective function value is less than the tolerance, the algorithm will stop iterating. The threshold is not on gbest value, but on the value of the objective function. The Primal Dual brings down the objective value to 1 and then gives control to the APSO algorithm. Once, APSO brings down the objective function down to 1e-8, the algorithm stops iterating. The Fig. 2 below illustrates the search space and how the swarm flocks after converging at the centre from the centre to z1, z2, z3, and z4 respectively.

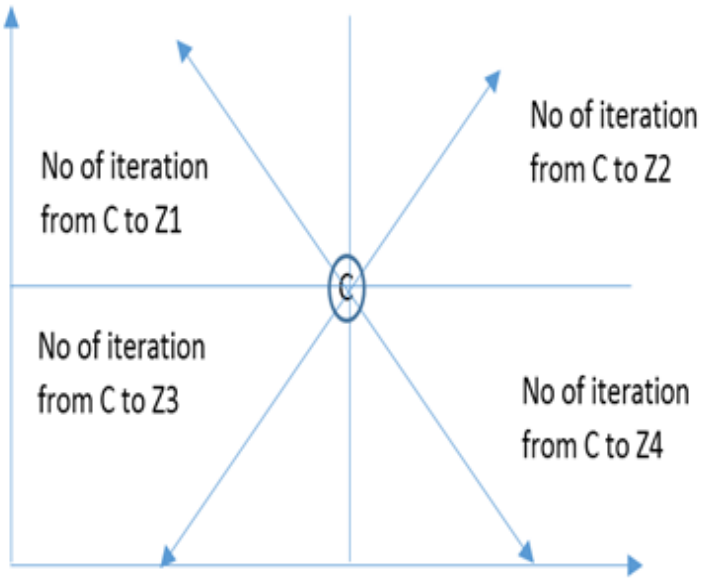

Figure. 2. Convergence and Flocking strategy. The Fig. 2 shows the zone 1, zone 2, zone 3, and zone 4 where the robots will flock to after converging at C.

We demonstrated the performance of our proposed hybrid Primal-Dual-APSO algorithm in a decentralized environment by simulating the movement of robots in the swarm in a virtual environment written in Matlab. There are some constraint that were set for the primal-dual algorithm based on the upper bound and the lower bound. The parameters for the APSO that were also set include the exploration environment which is a 200 x 200 dimension coordinate system. Each of the robots is symbolized with the black dots, the system is designed to operate in real time. The number of robot and the dimension of the search is specified in the GUI. This simulation was run on a MATLAB R2013a on an Intel ® Core ™ i3-2328M machine with 4GB memory running Windows 7. We designed a GUI shown in Fig. 3 to simulate how the robots converge to a point and then start flocking by simply clicking the mouse at the point where we want them to converge. The GUI is user-friendly and it easy to change the parameters. The user can halt, exit, or restart the simulation any time he feels like.

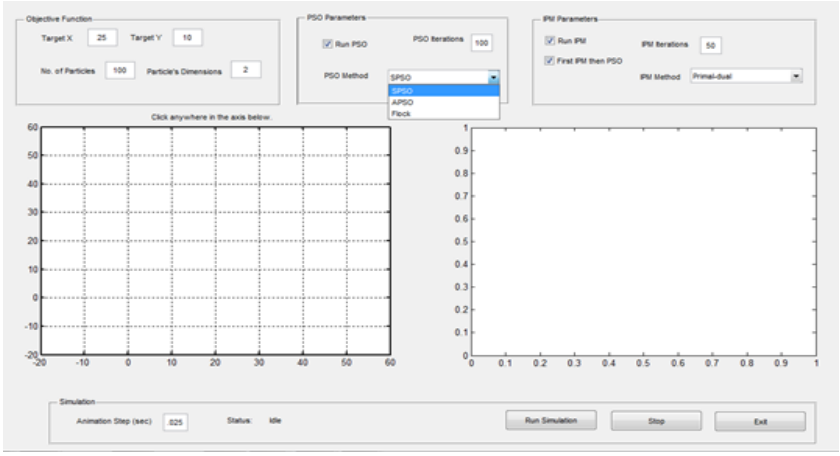

Figure 3. Screenshot of our GUI.

In Fig. 3, the GUI enables us to specify the values for some parameters such as the number of particles, the dimension of the particles, the coordinate on the x and y axis, the algorithms to use for the flocking, the number of iterations, and the animation steps (sec). Once the values are set the simulation can start by clicking Run Simulation button, we can stop the simulation by clicking on the stop button, and exit button when clicked will close the GUI. Figs. 4 – 9 shows the screen shots of our GUI, how the robots converged to the centre and how the flocking is done from the centre to the different zones.

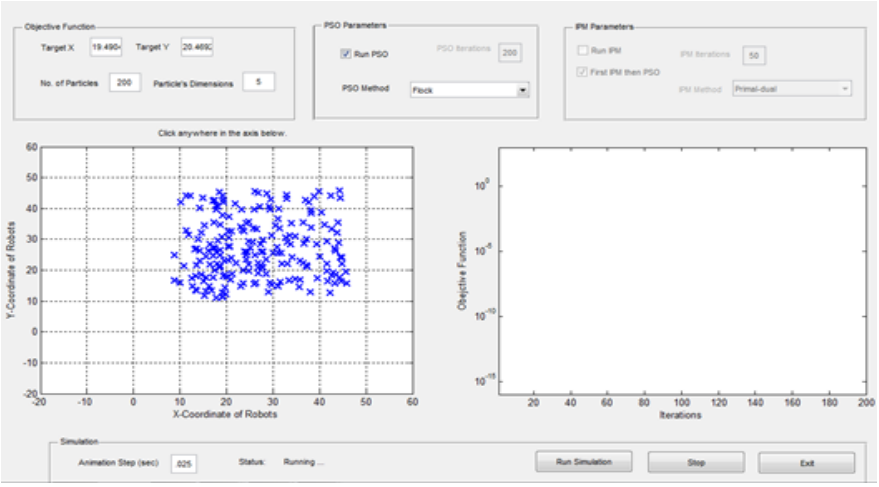

Figure 4. Screenshot of robots moving towards convergence on the GUI.

In figure 4, the 200 robots moving towards converging at the center as shown in the GUI.

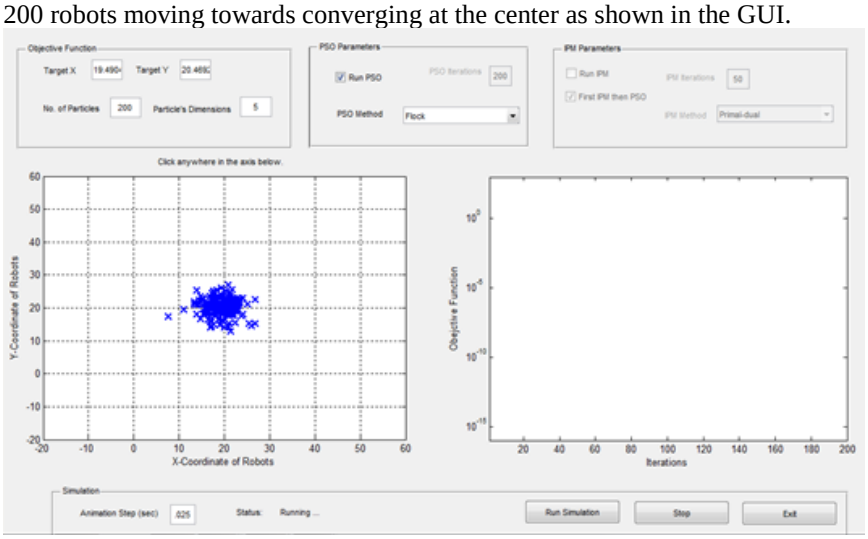

Figure 5. Screenshot of robots converging at the centre.

The Fig. 5 shows the robots converging at the centre.

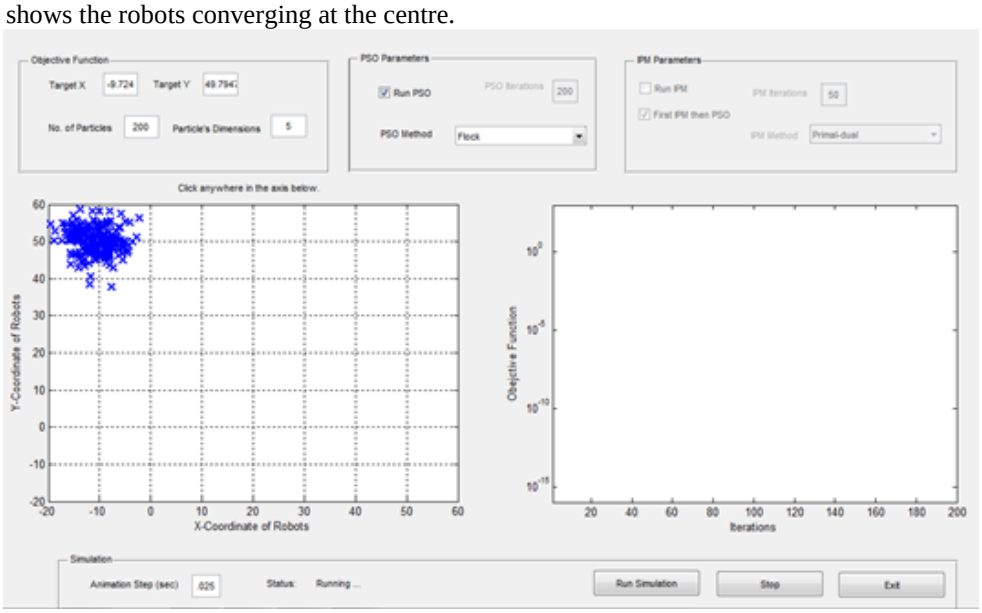

Figure 6. Screenshot of robots flocking towards zone 1 on the GUI.

The Fig. 6 shows the robots flocking from the centre to zone 1.

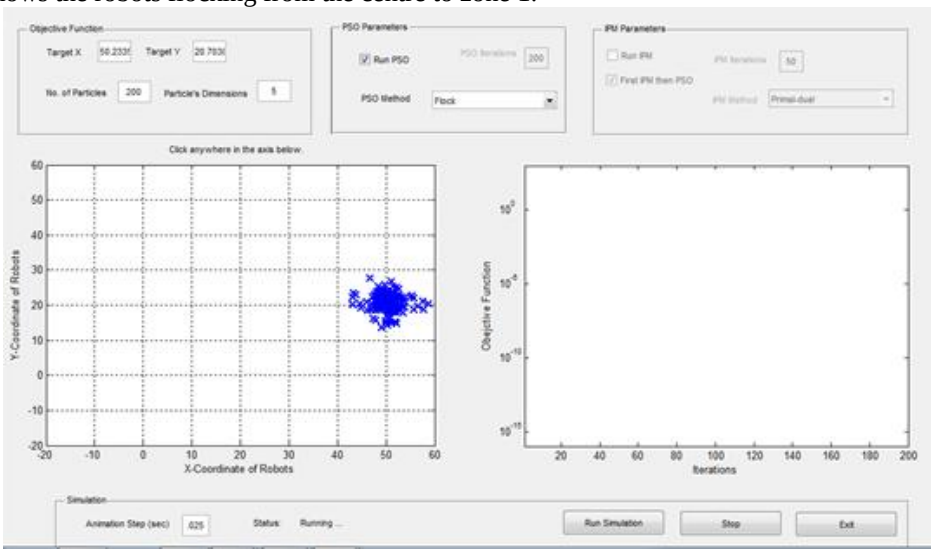

Figure 7. Screenshot of robots flocking towards zone 2 on the GUI.

The Fig. 7 shows the robots flocking from the centre to zone 2.

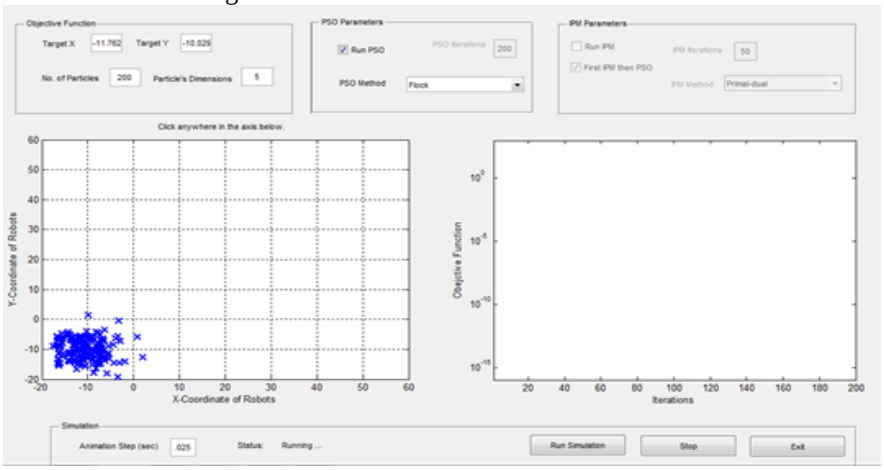

Figure 8. Screenshot of robots flocking towards zone 3 on the GUI.

The Fig. 9 shows the robots flocking from the centre to zone 3.

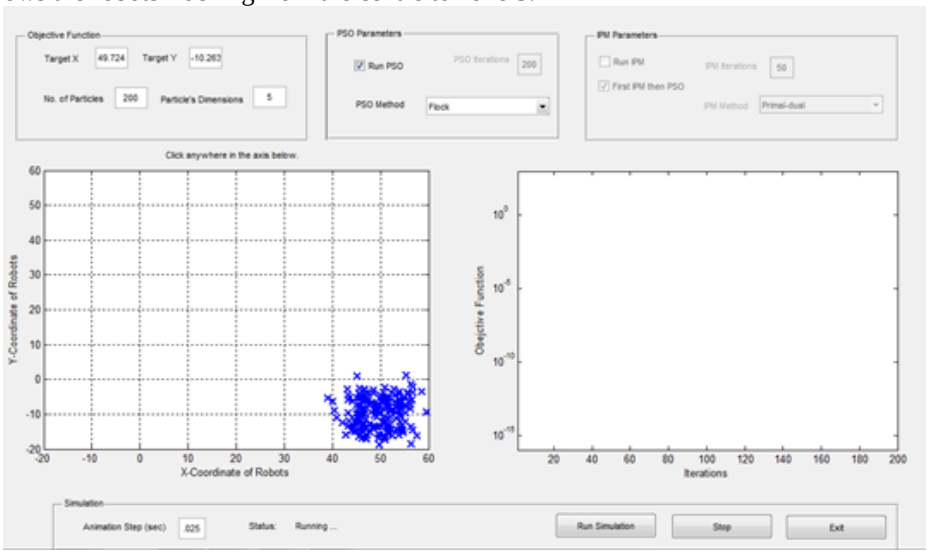

Figure 9. Screenshot of robots flocking towards zone 4 on the GUI.

The Fig. 8 shows the robots flocking from the centre to zone 4.

V. Result and Discussions

At the beginning of simulation, there is a haphazard allocation of agents A^i ($0<i\leq N$) in the environment to be explored. The new algorithm works by first starting the agents' positions randomly. Then, the agents are directed to the Primal-Dual method, which gives us its initial optimization result after some number of iterations. The result of the Primal-Dual optimization is then feed into APSO, which creates a perturbation in the population and also maintain diversity in the population until there is either convergence to the global optimal or the termination criteria is reached. During the iterations that are performed in the optimization, the agents carry out shift in the search space from one point to the other (which have connection to behaviour evolutions in space). The cognitive and social scaling factors of the APSO are very important parameters for determining the optimal global behaviors of the agents in the swarm. To evaluate the effectiveness of flocking capability of our *pdAPSO* algorithm we compared its performance with other existing algorithms such as the PSO, APSO and Primal Dual. We observed from our simulations that for Primal Dual algorithm, the robots in the swarm flock very tightly as a single point. When the agents in the swarm move from random starting position to converge at the centre (0, 0), they become almost one point, which means they are tightly flocked. When they move from the centre position (0, 0) to different zones, they are so tightly flocked that they move almost as a single point. The result is that no matter the zone, it always takes equal number of iterations to reach different zones. Therefore, we can say that flocking in Primal Dual is very tight compared to what is obtainable in *pdAPSO*, APSO and PSO in which particles still move around a center point randomly. The issue of tightness of the robots in the swarm when using Primal Dual for flocking is impracticable in the real world sense as a safe distance must be maintained between the robots to avoid collision.

Though Primal Dual have the lowest number of iterations to flock from one zone to another zone followed by *pdAPSO,* APSO and PSO respectively. We however observed from our results that the value for number of iterations to flock from one zone to another for Primal Dual is constant (value 10). This is understandable because in Primal Dual-based methods, there is no random number involved. Therefore, whether the robots move from the centre (0, 0) to z1, z2, z3 or z4, the algorithm moves in a deterministic way. As long as the distance between the centre (0, 0) and z1, z2, z3 and z4 points are equal, Primal Dual takes equal number of iterations. The fact that the number of iterations is constant further confirms the inability of the particles in Primal Dual algorithm to escape being trapped in the local minimal. We posit that Primal Dual having a lower number of iterations is not an index that it is better than any of the other three (3) algorithms we compared in this paper. It is possible that one iteration of Primal Dual is more complex and has more computations than one iteration of *pdAPSO,* APSO and PSO. However, there is a lot of variation in *pdAPSO,* APSO and PSO iterations when the robots in the swarm move from the centre (0, 0) to different zones, whereas in Primal Dual the variance is very low. The lower variance shows that Primal Dual keeps the particles tightly knit together compared to *pdAPSO,* APSO and PSO. If you look at the formula for PSO-based algorithms (*pdAPSO,* APSO and PSO), it has random numbers involved. Therefore, when we run the *pdAPSO,* APSO and PSO methods, every time the algorithm has slightly different trajectories for the robots. The different trajectory results in different number of iterations every time.

Figs. 10 – 13 shows the graph of total iteration to converge to the centre and flock from there to the different zones (zone 1, zone 2, zone 3, and zone 4) using *pdAPSO*.

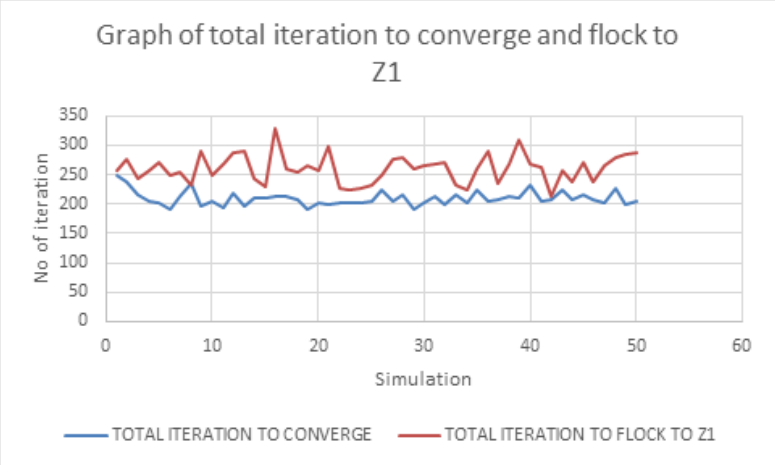

Figure 10. Graph of total iteration to converge and flock to Zone 1. This is a graphical illustration of the number of iterations it took the robots to converge to the center (0, 0) and flock to zone one (1) of the search space using the *pdAPSO* algorithm.

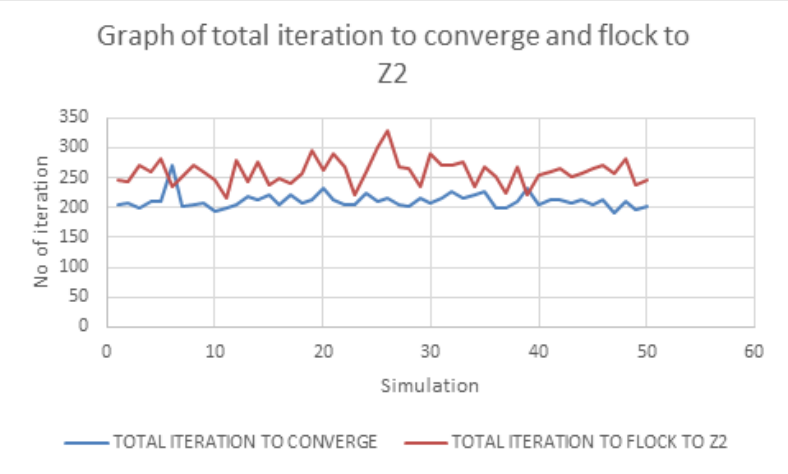

Figure 11. Graph of total iteration to converge and flock to Zone 2. The above figure depicts the graphical representation of the number of iterations it took the robots to converge to the center (0, 0) and flock to zone two (2) of the search space using the *pdAPSO* algorithm.

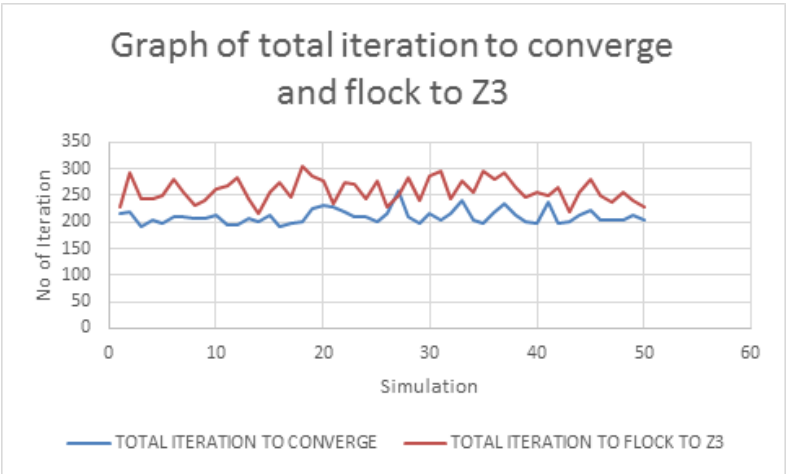

Figure 12. Graph of total iteration to converge and flock to Zone 3. Above is a graph of the number of iterations it took the robots to converge to the center (0, 0) and flock to zone three (3) of the search space using the *pdAPSO* algorithm.

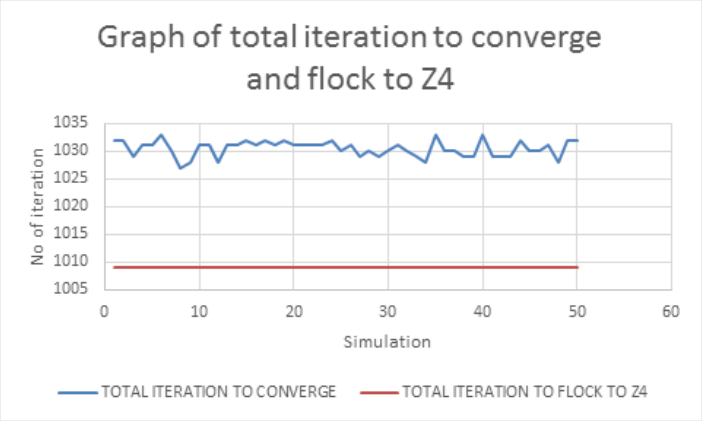

Figure 13. Graph of total iteration to converge and flock to Zone 4. The Fig. above shows the graphical depiction of the number of iterations it took the robots to converge to the center (0, 0) and flock to zone four (4) of the search space using the *pdAPSO* algorithm.

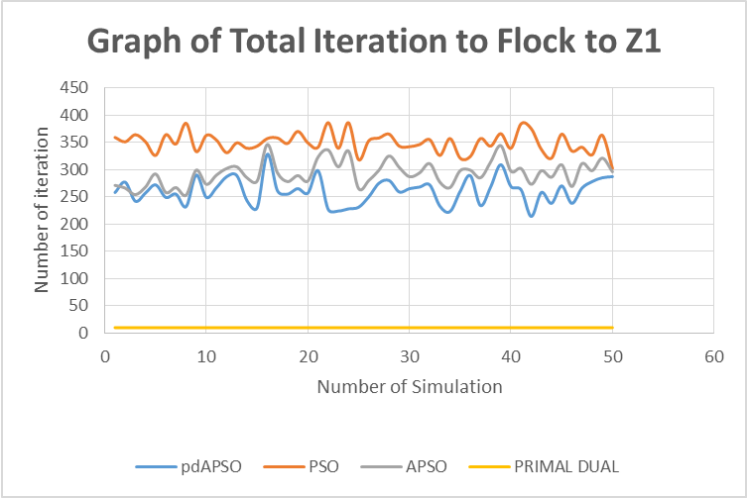

Figure 14. Graph of performance comparison of total iteration for *pdAPSO*, PSO, APSO and Primal Dual algorithms as the robots flock from the centre to Zone 1. The *pdAPSO* performs better than the other three (3) algorithms by having the lowest number of iterations in the fifty (50) simulation that was done (except for Primal Dual that have a constant number of iterations for each of the simulations because of its deterministic nature).

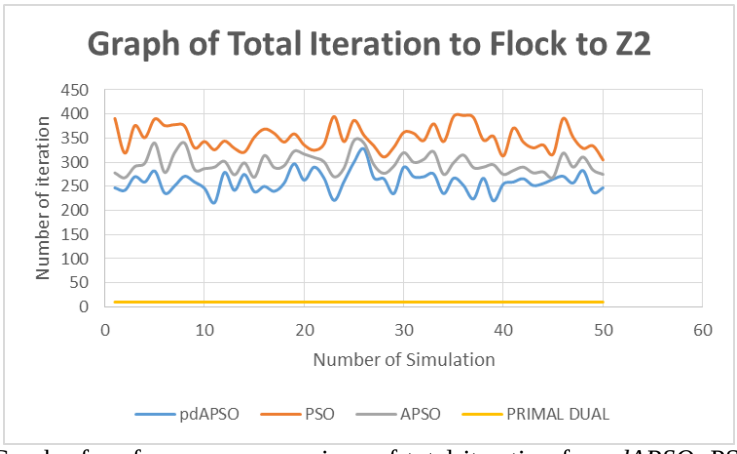

Figure 15. Graph of performance comparison of total iteration for *pdAPSO*, PSO, APSO and Primal Dual algorithms as the robots flock from the centre to Zone 2. The performance of *pdAPSO* is better than the one of the other three (3) algorithms by its ability to flock to zone 2 in using the minimum number of iterations in the fifty (50) simulation that was done (except for Primal Dual that have a constant number of iterations for each of the simulations because of its not a heuristic algorithm).

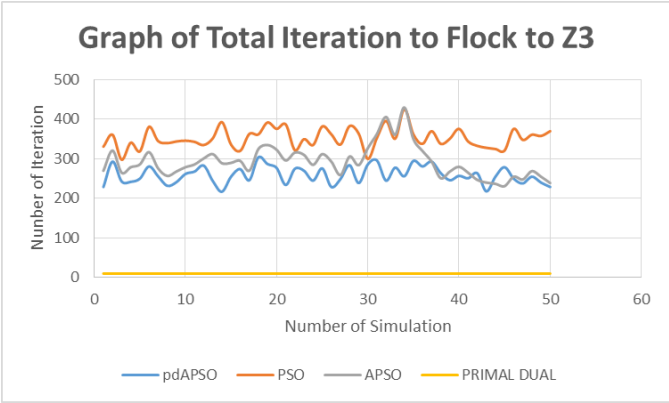

Figure 16. Graph showing the performance comparison of total iteration for *pdAPSO*, PSO, APSO and Primal Dual algorithms as the robots flock from the centre to Zone 3. Our algorithm *pdAPSO* shows a better flocking capability than that of PSO, APSO and Primal Dual algorithms by allowing the robots to flock to zone 3 by having the minimum number of iterations in the fifty (50) simulations that was done (except for Primal Dual that have a constant number of iterations value 10 for each of the simulations since it is a heuristic algorithm).

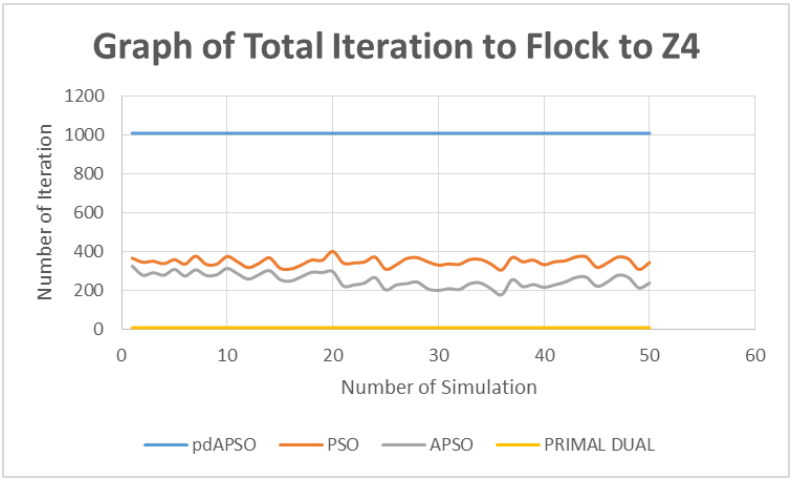

Figure 17. Graph showing the performance comparison of total iteration for *pdAPSO*, PSO, APSO and Primal Dual algorithms as the robots flock from the centre to Zone 4. The APSO algorithm performs better than *pdAPSO*, PSO and Primal Dual algorithms in this scenario by having the minimum number of iterations in the fifty (50) simulations that was done. The *pdAPSO* and Primal Dual a constant number of iterations value 1009 and 10 respectively for each of the simulations.

VI. Conclusion

This paper proposes a hybrid algorithm strategy (*pdPSO*) for flocking of swarm robots. This algorithm combines the explorative ability of APSO with the exploitative capacity of the Primal Dual Interior Point Method thereby possessing a strong capacity of avoiding premature convergence there making the robots to converge to a point and flock in real time. From the result of our simulations, the mean of the iteration for the agents to converge at the centre, and also flock from one zone to the other is almost the same. We decided to measure the number of iteration and not the time because the former is platform independent. The performance comparison of total iteration for *pdAPSO*, PSO, APSO and Primal Dual algorithms as the robots flock from the centre to the different zones (z1, z2, z3, and z4) in the search space was done. The *pdAPSO* performs better than the PSO, APSO and Primal Dual algorithms in the fifty (50) simulations that was done as the robots flock to z1, z2 and z3. The only exception was z4 where APSO have the best performance among the other algorithms. The results of our flocking simulations indicated that the performance of our algorithm is good in terms of precision, convergence rate, equilibrium, robustness and ability to flock using homogenous set of swarm robots. In our future work, we seek to extend the use *pdAPSO* to solving flocking problems of swarm robotics with obstacle avoidance.

Acknowledgements

I want to thank University of Maiduguri, Nigeria for financial support towards the successful completion of his PhD at University of Malaya, Malaysia.

References

[1]. L. Bayindir, E. Şahin, A Review of Studies in Swarm Robotics, *Turk. J. Elec. Engin. 15(2)*, 2007, 115–147.
[2]. Şahin E. Swarm Robotics. From Sources of Inspiration to Domains of Application. In: Şahin, E., Spears, W. M. (eds.) Swarm Robotics (2004) *Springer, Heidelberg, LNCS, 3342*, 2005,10–20.
[3]. C.W. Reynolds, Flocks, Herds, and Schools: A Distributed Behavioural Model, *Computer Graphics*, *21*(4), 1987, 25–34.
[4]. T. Vicsek, A. Czirok, E. B. Jacob, I. Cohen, O. Schochet, Novel Type of Phase Transitions in a System of Self-Driven Particles, *Physical Review Letters*, *75*, 1995, 1226–1229.
[5]. A. Jadbadaie, J. Lin, A. S. Morse, Coordination of Groups of Mobile Autonomous Agents Using Nearest Neighbor Rules, *IEEE Transactions on Automatic Control*, *48(6)*, 2003, 988–1001.
[6]. Y. Yang, N. Xiong, N. Y. Chong, X. Défago. A Decentralized and Adaptive Flocking Algorithm for Autonomous Mobile Robots *In: the 3rd International Conference on Grid and Pervasive Computing Workshops*, IEEE Press, 2008, 262–268.
[7]. D. H. Kim, H. Wang, S. Shin, Decentralized Control of Autonomous Swarm Systems Using Artificial Potential Function-Analytical Design Guidelines, *J. Int. Robot Systems*, 2006, 45:36–394.
[8]. S. Jeschke, H. Liu, D. Schilberg, Swarm Robot Flocking: An Empirical Study, *(Eds.): ICIRA, Part II, LNAI 7102*, 2011; 495–504.
[9]. Y. Shi, R. C. Eberhart, Parameter Selection in particle swarm optimization, *In Proceedings of the 7th International Conference on Evolutionary Programming*, 1998, 591 – 600.
[10]. N. A. A. Aziz, Z. Ibrahim, Asynchronous Particle Swarm Optimization for Swarm Robotics, *International Symposium on Robotics and Intelligent Sensors IRIS 2012, Procedia Engineering, 41*, 2012, 951 – 957.
[11]. C. R. Ward, F. Gobet, G. Kendall. Evolving collective behavior in an artificial ecology, *Artificial Life, 7(2)*, 2001,191-209.
[12]. L. Spector, J. Klein, C. Perry, M. D. Feinstein, Emergence of collective behavior in evolving populations of flying agents, *In E. Cantu-Paz et. al. editor. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, Berlin, Germany, 2003, 61-73.
[13]. M. Dorigo, V. Maniezzo, A. Colorni, Ant System: Optimization by a Colony of Cooperating Agents (1996) *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics* 26 (1): 29 - 41.
[14]. D. Payton, M. Daily, R. Estowski, M. Howard, C. Lee, Pheromone Robotics (2001) *Autonomous Robots*, 11 (3):319 - 324.
[15]. V. Y. Del, G. K. Venayagomoorthy, S. Mohagheghi, J. C. Hernandez, R. G. Harley, Particle swarm optimization: basic concepts, variants and applications in power systems (2008) *IEEE Trans Evol Comput*, 12 (2):171 – 195.
[16]. J. Kennedy, R. C. Eberhart, Particle swarm optimization (1995) in: *Proceedings of the International Conference on Neural Networks*, vol. 4, IEEE Press, Piscataway, NJ, pp. 1942-1948.
[17]. D L. Carl, *Structured Large-Scale Nonlinear Optimization Using Interior-Point Method: Applications in Water Distribution Systems*, PhD Thesis Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, 2006.
[18]. S. Mehrotra, On the implementation of a primal-dual interior point method, *SIAM Journal on Optimization*, *2*, 2005, 575-601.
[19]. K R. Frisch, *The logarithmic potential method of convex programming*, Technical Report, University Institute of Economics, Oslo, Norway, 1955.
[20]. S. Boyd, L. Vandenberghe, *Convex Optimization*, Cambridge University Press, New York, 1st edition, 2004.
[21]. E. Chouzenoux, S. Moussaoui, J. Idier, Efficiency of line search strategies in interior point methods for linearly constrained signal restoration, *Statistical Signal Processing Workshop (SSP), IEEE*, 2005, 101- 104. DOI: 10.1109/SSP.2011.5967631,
[22]. C. A. Johnson, J. Seidel, A. Sofer, Interior-point methodology for 3-D PET reconstruction, *IEEE Trans. Medical Imaging*, *19(4)*, 2000, 271 - 284.
[23]. S. J. Wright, Primal-dual interior-point methods, *SIAM, Philadelphia*, PA, 1st edition, 1997.
[24]. V. H. Quintana, G. L. Torres, Introduction to Interior-Point Methods, *IEEE PES task Force on Interior-Point Methods Applications to Power Systems*, 1997.
[25]. V. Selvi, R. Umarani, Comparative Analysis of Ant Colony and Particle Swarm Optimization techniques, *International Journal of Computer Applications*, 5(4), 2010,1-6.
[26]. J. B. Park, J. R. Shin, K. Y. Lee, Y. W. Jeong, An Improved Particle Swarm Optimization for Non-Convex Economic Dispatch Problems, *IEEE Transactions on Power Systems, 25(1)*, 2010, 156- 166
[27]. S. Das, A. Abraham, A. Konar. Metaheuristic Clustering, Studies in Computational Intelligence, *Springer Verlag, Germany, 178*, 2009,1-62.
[28]. A. Z. Torn, Global Optimization, *Lecture Notes in Computer Science, Springer-Verlag*, 350 pages, 1989.
[29]. A. Abraham, M. Pant, P. Bouvry, and R. Thangaraj, Particle swarm optimization: Hybridization perspectives and experimental illustrations, *Appl. Math. Comput., vol. 217*, 2011, 5208 – 5226, doi:10.1016/j.amc.2010.12.053
[30]. J. Kennedy, R. Eberhart, Y. Shi. Swarm Intelligence, *Morgan Kaufmann Publishers*, 2001, US.
[31]. C. Pinar, B. Erkan, A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms, *Artif Intell Rev*, *39*, 2013, 315-346, DOI 10.1007/s10462-011-9276-0.
[32]. A. George, B. I. Koh, B. Fregly, R. Haftka, Parallel asynchronous particle swarm optimization. *International Journal for Numerical Methods in Engineering*, *67*, 2006, 578–595.
[33]. J. Luo, Z. Zhang, Research on the Parallel Simulation of Asynchronous Pattern of Particle Swarm Optimization, *Computer Simulation 22(6)*, 2006, 78–70.
[34]. J. Schutte, *Particle Swarms in Sizing and Global Optimization*. Master's thesis, University of Pretoria, South Africa, 2001.
[35]. A. Carlisle, G. Dozier, An off-the-shelf PSO, *In: Workshop on Particle Swarm Optimization*, 2001, 1-6.
[36]. J. Basterrechea, J. R. Perez, Particle swarm optimization and its application to antenna far field-pattern prediction from planar scanning, *Microwave and optical technology letters 44(5)*, 2005, 398–400.
[37]. Bjorn Gernert, Sebastian Schildt, Lars Wolf, Bjorn Zeise, Paul Fritsche, Bernardo Wagner, Maksims Fiosins, Ramin Safar Manesh, Jorg P. Muller, An interdisciplinary approach to autonomous team-based exploration in disaster scenarios, *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2014, 1 - 8.
[38]. Sifat Momen, P. Bala. Amavasai, H. Nazmul Siddique, Mixed Species Flocking for Heterogeneous Robotic Swarms, *EUROCON 2007 The International Conference on "Computer as a Tool", IEEE*, 2007, 2329 - 2336.
[39]. E. G. Dada, and E. I. Ramlan, Primal-Dual Interior Point Method Particle Swarm Optimization (pdipmPSO) Algorithm. In: 3rd *Int'l Conference on Advances in Engineering Sciences and Applied Mathematics (ICAESAM'2015)*, London (UK), 2015a, 117-124.
[40]. E. G. Dada, and E. I. Ramlan, A Hybrid Primal-Dual-PSO (*pdipmPSO*) Algorithm for Swarm Robotics Flocking Strategy. The *Second International Conference on Computing Technology and Information Management (ICCTIM2015).* Malaysia, IEEE, 2015b, 93-98, ISBN: 978-1-4799-6211-2.
[41]. E. G. Dada, and E. I. Ramlan, Understanding the limitations of particle swarm algorithm for dynamic optimization tasks. *ACM Computing Survey, Vol. 49, No. 1, Article 8*, 2016, pp 8:1-8:25, DOI: http://dx.doi.org/10.1145/2906150.