Mobile Robot Path Planning in an Obstacle-free Static Environment using Multiple Optimization Algorithms

C. O. Yinka-Banjo*, U. Agwogie

Department of Computer Sciences, University of Lagos, Akoka, Nigeria.

ABSTRACT: This article presents the implementation and comparison of fruit fly optimization (FOA), ant colony optimization (ACO) and particle swarm optimization (PSO) algorithms in solving the mobile robot path planning problem. FOA is one of the newest nature-inspired algorithms while PSO and ACO has been in existence for a long time. PSO has been shown by other studies to have long search time while ACO have fast convergence speed. Therefore there is need to benchmark FOA performance with these older nature-inspired algorithms. The objective is to find an optimal path in an obstacle free static environment from a start point to the goal point using the aforementioned techniques. The performance of these algorithms was measured using three criteria: average path length, average computational time and average convergence speed. The results show that the fruit fly algorithm produced shorter path length (19.5128 m) with faster convergence speed (3149.217 m/secs) than the older swarm intelligence algorithms. The computational time of the algorithms were in close range, with ant colony optimization having the minimum (0.000576 secs).

KEYWORDS: Swarm intelligence, Fruit Fly algorithm, Ant Colony Optimization, Particle Swarm Optimization, optimal path, mobile robot.

[Received January 22, 2020, Revised June 16, 2020, Accepted August 22, 2020] Print ISSN: 0189-9546 | Online ISSN: 2437-2110

I. INTRODUCTION

In an environment, there are many paths for a robot to reach a specified goal, but the best path is selected according to some criteria. These criteria are the shortest distance, the shortest time, the least energy consumed. The most adopted criterion is the shortest distance. Path planning is an optimization problem since its purpose is to search for a path with the shortest distance under certain constraints such as the given environment with collision- free motion (Mansi et al, 2013).

Path planning algorithms can be classified into heuristics and non-heuristics. Some of the well-known non-heuristics are cell decomposition, Voronoi diagrams, and B-Spline curve. In the cell decomposition technique, two methods are used. They are the exact and the approximate cell decomposition methods. The exact method is used to divide the search space into simple cells and builds the adjacency relationships among the cells. It explicitly determines the obstacles and build the cells (Choset, 2007), (Abbadi, et al. 2015). The combination of all the generated cells will produce the exact free space. However, determining the exact free space in a high dimensional environment is not an easy task, hence the approximate method was introduced.

Vonoroi diagram represents regions of influence around a given set of points in a plane. Each region corresponds to one section of the plane and all the points in one region are closer to the section representing the region than to any other section. Solanki et al., generated vonoroi diagran as the obstacle region then used the adjacent vertex of the diagram as the start and goal points. Then using the Dijkstra algorithm; they searched for the shortest route from start to goal. B-splines is a mathematical function that is used to form a curve using a few control points in a segment rather than the entire points in the segmented section. Connors and Elkaim (2007), applied the B-spline function with a modification by introducing additional control points in the neighborhood of each obstacle, they also develop methods to shift these new control points away from obstacles and into clear areas. E. Kan et al, introduced user-specified threshold flying altitude in the enemy terrain and used these new thresholds to generate a path for the Unmanned Aerial Vehicles.

To overcome the limitations of classic methods to path planning; researchers have over time move towards heuristics methods. Heuristics methods has helped to deal with the complexities and computational costs associated with classical methods. However, one is not sure to come across a solution while using the heuristics methods, but if there is a solution it will be found much faster than the classical methods. There is an increase in the development on heuristics methods over the past two decades. Some of the heuristics techniques were inspired from nature and can also be referred to as nature inspired algorithms.

*Corresponding author: cyinkabanjo@unilag.edu.ng

doi: http://dx.doi.org/10.4314/njtd.v17i3.3

Nature-inspired algorithms are stochastic search methods that mimic the behavior of natural biological evolution and/or the social behavior of species. The behavior of these species is guided by learning, adaptation, and evolution. To imitate the efficient behavior of these species, several researchers have developed computational systems that seek fast and robust solutions to complex optimization problems. These algorithms can be broadly classified into Evolutionary algorithms and Swarm Intelligence based algorithms using their form of inspiration (Binitha et al, 2012).

Swarm Intelligence (SI) is the study of the collective behavior and emergent properties of complex systems within predefined environment (Narendra et al, 2013). This functionality creates the possibility of solving problem using collective or distributed approaches. The provision of a centralized control or global model is not required as the field focuses on the collective behaviors that emerges from the local interactions of the agents with each other and their environment. Particle swarm optimization (PSO), ant colony optimization and fruit fly optimization algorithms are examples of the swarm intelligence techniques.

A. Particle Swarm Optimization (PSO)

Particle swarm optimization is a global optimization method proposed by Doctor Kennedy and Eberhart in 1995. PSO is inspired by the social foraging of bird flocking together. In PSO, the bird is represented as a particle. A population of n particles is randomly initialized with random position and velocities. The position of each particle stands for the potential solution in the search space. The particle uses some principles to change its position: its inertia is maintained, its best position and the position of the best positioned particle (Qinghai, 2010). The fitness of a particle is calculated based on its distance to the destination. Each particle updates its position and velocity with its own memory and the social information gathered from other particles.

Ajeil et al (2020), used the PSO algorithm with modified frequency algorithm to solve multi-objective path planning of an autonomous robot. The multi-objective goal is aimed at achieving the shortest and smoothest path. Cholodowicz et al (2017) applied a constrained PSO algorithm in static and dynamic environments where a virtual robot was used to control strategy and also check the efficiency of the proposed methods.

B. Ant Colony Optimization (ACO)

Ant colony optimization is a meta-heuristic stochastic optimization technique developed by Marco Dorigo in the early 1990s (Christian, 2005), (AbWahab et al, 2015). ACO was inspired from the ants searching behavior in finding the shortest path between their nest and food sources. They randomly explore around their nest at the initial stage and then towards other regions in their quest for food. While moving around, a chemical substance called pheromone is being deposited by the ant along the path. As soon as a food source is found, the quality and quantity are evaluated by the ant. Ants use the same path to the food source back to their nest, by doing so, more pheromone will be deposited on that path. This chemical substance can be smelt by all ants which serves as a communication information to other ants. As new ants leave their nest for search of food sources, they tend towards paths with high concentration of these pheromone (fitness value) It is believed that a path with high pheromone concentration may likely lead to a food source. The presence or absence of pheromone trails on a path serves as a positive and negative feedback respectively.

The performance of ACO and Firefly algorithm in different dynamic environments of a rubber plantation was compared by (Gangadharan et al, 2020). Simulations shows that FF outperformed ACO in terms of path length and time of execution. Yue (2019) used a novel ACO algorithm for unmanned vehicle path planning, they introduce the use of the search results of the poor path to enhance the volatilization degree of the pheromone on the poor path and reduce the number of traversal times. In this, they believe that the concentration of the pheromone in the unexplored path will be larger than the worst path and in turn exposing the ants to a better solution in the unknown field.

C. Fruit Fly Optimization Algorithm (FOA)

Fruit fly algorithm is one of the newest meta-heuristics algorithm in the class of swarm intelligence algorithms. It was proposed by (Wen-Tsao, 2014), (Rizk, 2016). The inspiration came from the foraging behaviors of the fruit flies in their search for food using their sense of vision and smell (Hazim et al, 2014). They have superior sense of smell and vision compared to other species. A fruit fly can smell food at a distance of 70km away from the food source (Ye et al, 2017). The Fruit flies can measure the smell concentration in their current position then compare their fitness. The swarm will then move towards the location with the best fitness (Sheng et al, 2017).

The basic characteristics of the fruit fly algorithm can be deduced to four steps: initialization, olfactory searching, vision searching and termination. The fruit fly optimization algorithm has many advantages such as a simple structure, easy to implement, less parameter to adjust and fast convergence in finding solutions. Some drawbacks in the basic fruit fly algorithm (Rizk, 2014), (Shui-ping et al, 2016) are its premature convergence, poor swarm diversity and lack of mechanism that enables it to jump out of local optimum.

Aiming at improving the standard fruit fly algorithm, Li and Han introduced the fusion immune function at the later stage of the search to enable it escape the fall into local extremum (Rizk, 2014), (Sheng et al, 2017). Xing Guo et. al, also introduced an improved fruit fly algorithm using a traction population of fruit flies (Xing et al, 2017). This is using all the worst recorded fly in each iteration when the algorithm has fallen into its local optimum without finding a global minimum. Then using this new population; it explores a larger solution space in opposite direction in the quest for the best fly.

The aim of this research is to implement the traditional Fruit Fly Optimization Algorithm, to solve the Mobile Robot Path Planning problem to compare its performance with older nature-inspired algorithm. To achieve this aim, the first version

of Ant Colony Optimization and Particle Swarm Optimization Algorithms are implemented and the results compared against some performance criteria.

The following performance criteria were considered to confirm the algorithm with the optimal path.

i. Average Path length
ii. Average Execution time
iii. Average Convergence speed

II. PROBLEM FORMATION

The mobile path planning problem is modelled as a global path planning whereby the positions of the start node, path nodes and goal node are known to the robot prior to its path planning. The nodes are initialized using its x and y coordinates. The start node is positioned at (0, 0) and the goal node is placed at (10, 10). The search space is bounded by the lower and upper coordinates of the start and goal node. The robot is represented as a point in the search space to avoid computational complexities. Within the search space, there exist 13 path nodes that can be selected from the start node to the goal node. The search space is divided into sub swarms such that nodes are interconnected using a matrix that forms the visibility constraint of the robot. This is to say that the robot cannot jump to the goal node from the start node; it must select a path node from the possible nodes that are visible to it. The Sub swarms are bounded by the upper and lower coordinates of the possible nodes. The goal is to determine the shortest path to the goal node from the start node. A feasible solution is represented by a sequence of vertices linking the start node to the goal node.

A. *Mathematical Representation of the Problem*

As shown in Fig. 1, the environment is represented in a 2-dimensional map where the start node, $S\ (node\ 1)$ and the goal node, $G\ (node\ 15)$ are represented in blue while the path nodes $(node_i, i = 2,3,4,5,6,7,8,9\ ,10,11\ ,12\ ,14)$ are represented in yellow.

There are four sub swarms $(P_i,\ i = a,b,c,d)$ within the search space with different number of path nodes. The composition of the sub swarms is given below:

$Subswarm\ A = (P_a, a = 2,3), Subswarm\ B = (P_b, b = 4,5,6,7), Subswarm\ C = (P_c, c = 8,9,10,11), and\ Subswarm\ D = (P_{d,} d = 12,13,14)$.

A robot R, is initialized at the start node $(X_0\ , Y_0)$ in time t. (X', Y') is the next position of the robot in time $t + 1$ The Robot selects the next path node using a criteria that satisfies the constraint function (Shortest distance).

The initial positions of the swarm population for the implementation were initialized every time a new node is selected until the goal node is reached. The coordinates of the path nodes within the sub swarm represents the lower bound and upper bound for which the initialization was done; as shown in Eq. (1)(Xing et al, 2017).

$$X_i = X_{axis} + rand(LR),\ Y_i = Y_{axis} + rand(UR) \quad (1)$$

B. *The Fitness Function*

To ensure proper search is done to avoid exploitation of the algorithms, local and global search techniques were employed. The fitness function is sub divided into two: Firstly, to determine the individual in the swarm with the best fitness value Eq. (2) (Lv et al, 2017).; Secondly using the coordinates of the best fit individual; we can determine the next node the robot can move to Eq. (3) (Lv et al, 2017).

$$BestFit_i = \min F(x) \quad (2)$$

$$NextNode_i = \min N(x) \quad (3)$$

$F(x)$ Can be computed from Eq. (4) (Zhang et al, 2016). while $N(x)$ is determined from Eq. (7)

$$F(x) = S(x) + G(x), \quad 2 \le x \le \quad (4)$$

where the first term (local search) computes the distance of each swarm from the origin $(X_0\ , Y_0)$ to its current position (X', Y') using Eq. (5) (Allah, 2016), (Zhang et al, 2016).

$$S(x)_i = \sqrt{(X_i' - X_0)^2 + (Y_i' - Y_0)^2} \quad (5)$$

The second term (global search) computes the distance of each swarm from its current position (X', Y') to the goal node $(X_g\ , Y_g)$ as seen in Eq. (6) (Lv et al, 2017.

$$G(x)_i = \sqrt{(X_g - X_i')^2 + (Y_g - Y_i')^2} \quad (6)$$

The X and Y coordinates of the best fit individual in the swarm is then applied to Eq. (7) (Lv et al, 2017). to determine the next path node to be traverse to within the sub swarm.

$$N(x) = \sqrt{(X_n - X_{bestfit})^2 + (Y_n - Y_{bestfit})^2} \quad (7)$$

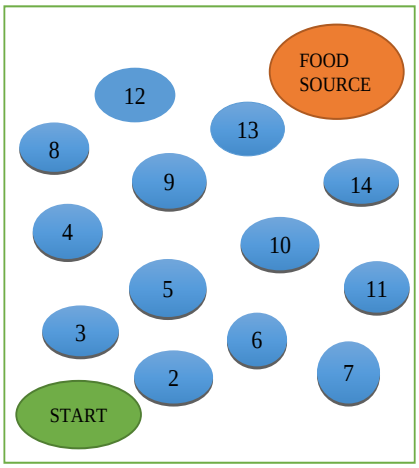

Figure.1: Proposed Pseudo-code for Fruit Fly optimization algorithm.

III. IMPLEMENTATION OF THE SWARM INTELLIGENCE ALGORITHMS

The described problem statement was solved using the three swarm intelligent algorithms (FOA, ACO and PSO). These algorithms have their respective drawbacks; FOA suffers from high processing time due to its poor feedback mechanism and in turn has premature convergence (Lv et al, 2017), (Zhang et al 2016). ACO lack a centralized processor to guide it towards good solutions and performs poorly in large search spaces (AbWahab et al, 2015). PSO suffers from weak local search ability which leads it to slow convergence in a refined search area (AbWahab et al, 2015). However, the implementation did not seek to improve the traditional algorithms rather the implementation of the traditional algorithms is to benchmark the new heuristics algorithm FOA in the obstacle free static environment for mobile path planning.

A. Implementation of Mobile Path Planning Using FOA

In the implementation, the algorithm begins by randomly initializing the initial positions of the fruit flies, then assigning random distance and direction to them. The fitness value of each fruit fly is evaluated to determine the best fruit fly in the swarm. After which the coordinates of the best fruit fly are used to compute the next node the robot can move to from the nodes visible to it. The algorithm terminates once the selected node is same as the goal node. Fig. (2) and (3) gives detailed pseudo-code and flowchart of the proposed method.

```
Set the start, goal and sub swarms nodes
While (goal node =false)
Initialize the swarm initial position using Equation (1)
Assign distance and direction to each fruit fly using Equation (1)
Calculate the Smell Concentration (Fitness Value) of each fruit fly using Equation (4)
Determine the Best Fly using Equation (2)
Calculate the transition probability of the robot using Eq. (7)
Determine the next node using Eq. (3)
Save the next node and its x and y coordinates
End While
```

Fig. 2: Proposed Pseudo-code for Fruit Fly optimization Algorithm

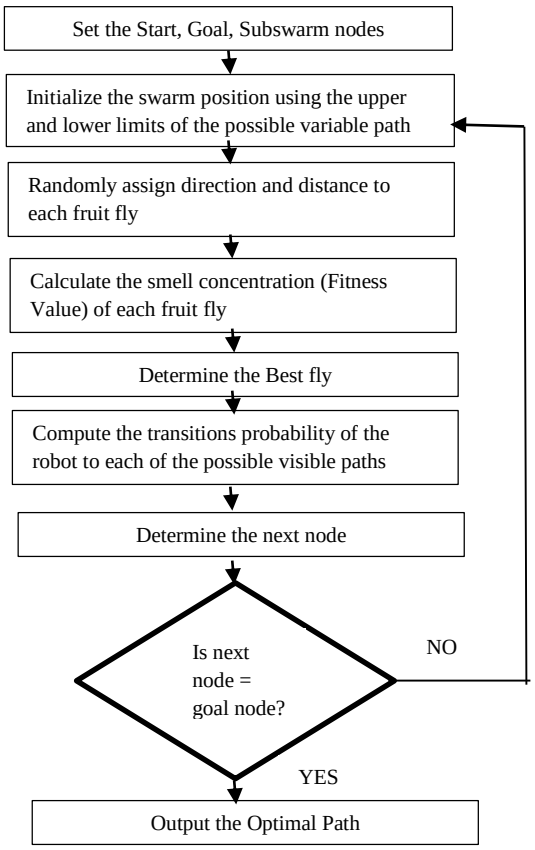

Fig. 3: Flowchart of Fruit Fly optimization algorithm.

D. Implementation of Mobile Path Planning Using PSO

In the implementation, the algorithm begins by randomly initializing the positions and velocities of the particles in the swarm, then the fitness value of each particle is evaluated to determine the local best particle. The local best fitness value is assigned to become the global best. The swarm particles velocities and positions were then updated to enable the particles move towards the global best particle in the swarm using Eq. (8) and Eq. (9) respectively (Qinghai, 2010).

$$New\ V_i = u * current\ V_i + c1 * rand(0,1) * (P_i - X_i) + c2 * rand\ (0,1) * (P_g - X_i) \quad (8)$$

$$New\ postn\ X_i = cur\ postn\ X_i + New\ V_i \quad (9)$$

where, X_i represents the current position of the particle, P_i represents the best previous position, V_i represent the current velocity of the particle, $c1$, $c2$ are two positive constants named learning factors which regulates the speed of moving towards the most optimal particle of the swarm and towards the individual particle; *rand* (0,1) represents the random functions in the range [0, 1] and *u* represents an inertia weight employed as an improvement on the basic PSO.

The fitness value of the particles is re-evaluated again to determine the new global best particle in the swarm. After which the coordinates of the global best particle are used to

compute the next node the robot can move to from the nodes visible to it. The algorithm terminates once the selected node is same as the goal node. Figs. (4) and (5) give detailed pseudo-code and flowchart of the proposed method.

```
Set the start, goal and sub swarms nodes, learning factors (c1, c2), weight (u)
While (goal node =false)
 Initialize the swarm initial position and velocities of each particles using Eq. (1)
 Calculate the Fitness Value of each particle using Eq. (4)
 Assign the Local best value to Global best
 Update the velocities of the particles using Eq. (8)
 Update the positions of the particles using Eq. (9)
 Determine the Global Best particle using Eq. (2)
 Calculate the transition probability of the robot using Eq. (7)
 Determine the next node using Eq. (3)
 Save the next node and its x and y coordinates
End While
```

Fig. 4: Pseudo code for Particle Swarm Optimization Algorithm.

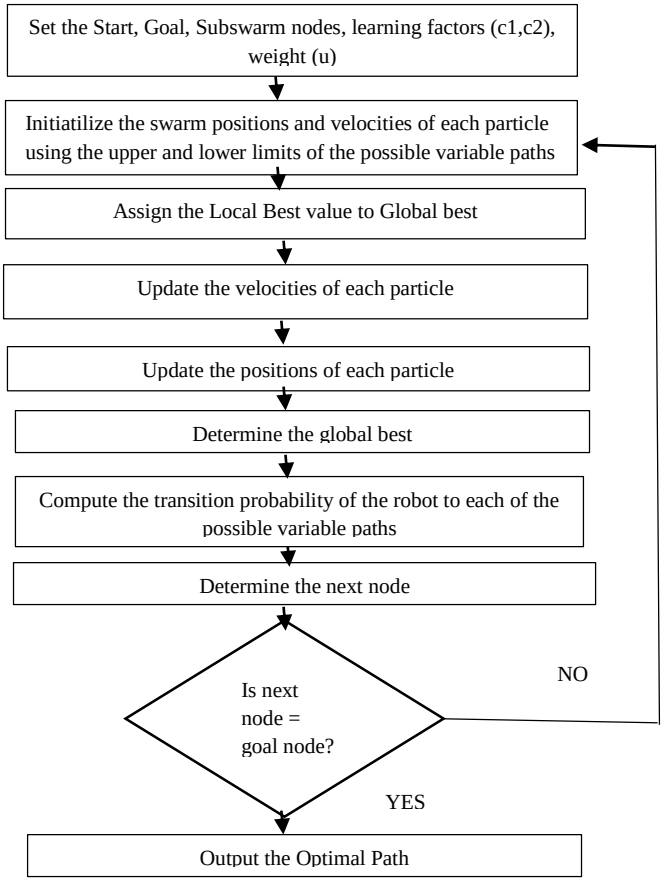

Fig. 5: Flowchart of Particle Swarm Optimization algorithm.

E. Implementation of Mobile Path Planning Using ACO

In the implementation, the coordinates of the nodes were used to determine the weights and length of each path which pheromone was initialized. The algorithm begins by initializing the pheromone concentration on each path to zero. This is to mean that the ants are still in their nest. The pheromone concentration on a path is determined from the length of the path, weight on each node, and attractiveness of the node. As the ants move, more pheromone concentration on each path are updated and consequently evaporated to allow exploration using Eq. (10) (Blum, 2005) and Eq. (11) (AbWahab et al, 2015) respectively.

$$\tau_i = \tau_i + \frac{Q}{l_i}, \qquad Q > 0, \tag{10}$$

$$\tau_{(a,b)}(t+1) = (1-\rho) * \tau_{(a,b)(t)} + \sum_{(k=1)}^{m} \left[\Delta\tau^k_{(a,b)}(t)\right] \tag{11}$$

The ants then apply the transition probability on each node to determine the node with high pheromone concentration using Eq. 12. (Blum, 2005).

$$p^k_{(a,b)}(t) = \frac{\left(\left[\tau_{a,b}(t)\right]^{\alpha} * \left[\cap_{a,b}\right]^{\beta}\right)}{\left(\sum_{b \epsilon bc}\left[\tau_{a,b}(t)\right]^{\alpha} * \left[\cap_{a,b}\right]^{\beta}\right)} \tag{12}$$

If α is higher than β, the searching probability will be dependent on the pheromone concentration otherwise it will be dependent on its visibility knowledge.

To allow exploration; the greedy selection method was not used rather the robot randomly select a node from the possible visible nodes. The algorithm terminates once the selected node is same as the goal node. Figs. (6) and (7) gives detailed pseudo code and flowchart of the proposed method.

```
Set the start, goal and subs warms nodes, alpha, beta, attractiveness, evaporation rate
 While (goal node =false)
 Initialize the paths pheromone concentration to zero
 Compute the length of each path using its weight (x, y coordinates)
 Calculate the pheromone Concentration on each path using the length
 Update the pheromone concentration of each path using eq. (10)
 Apply the pheromone evaporation rate using Eq. (11)
 Calculate the transition probability of the robot using Eq. (12)
 Randomly determine the next node
 Save the next node and its x and y coordinates
End While
```

Fig 6: Pseudo code for Ant Colony Optimization Algorithm.

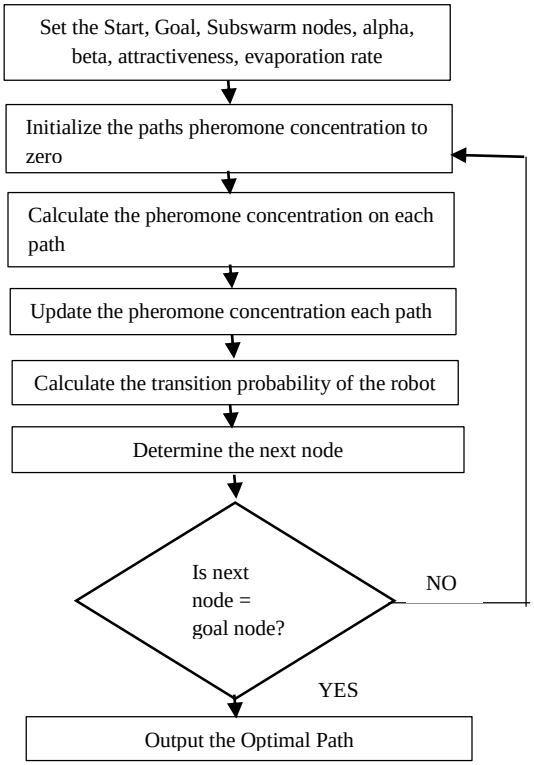

Fig.7: Flowchart of Ant Colony Optimization Algorithm.

IV. SIMULATION RESULT AND DISCUSSION

The robot environment was represented as a graph bounded with the coordinates of the start and target nodes. The start node is at (0.0) and the target is at (10, 10). The search space consists of 15 static nodes where Node 1 is the start node and Node 15 is the target node. The experiment is to generate a path from the start node to the goal node. The three swarm intelligence algorithms were applied to generate a path. The performance of these algorithms is compared using the following performance criteria: Average shortest distance, Computational time, Convergence Speed. The code was written with Python programming language on the spider editor. The experiment is performed on a 1.70GHz dual core CPU computer with 4GB RAM. The user must input the swarm size.

The experiment is presented in obstacle free environment. We used the three swarm intelligence algorithm to find the shortest path for a Mobile robot to move from the start node to the target node. A transition matrix of nodes in the environment was generated and the positions were known to the robot. Each node was assigned a computed weight which is called the cost of selecting the node (CN). Number of runs (NN) was given as a parameter before simulation begins. The results of the simulation are to generate average path length, average execution time and average convergence speed.

The path length (PT) is calculated using sum of the cost of the selected nodes from start node (s) to the goal node (g).

$$PT = \sum_{g}^{s} CN \quad (13)$$

The execution time (ET) is calculated using the time the robot reaches the goal node minus the start time of the algorithm.

$$ET = Endtime - Start\ time \quad (14)$$

The convergence speed (CS) is calculated using sum of the path length divided by execution time for each run.

$$CS = \frac{PT}{ET} \quad (15)$$

The average path length is calculated using sum of the cost of the selected nodes for each run divided by the number of runs the algorithm ran before reaching the target node.

$$Avg.PT = \sum_{i=1}^{nn} PT/NN \quad (16)$$

The average execution time is calculated using sum of the time the robot reaches the goal node minus the start time of the algorithm for each run divided by the number of runs the algorithm ran before reaching the target node.

$$Avg.ET = \sum_{i=1}^{nn} ET/NN \quad (17)$$

The average convergence speed is calculated using sum of the convergence speed for each run divided by the number of runs the algorithm ran before reaching the target node.

$$Avg.CS = \sum_{i=1}^{nn} CS/NN \quad (18)$$

In Table I, the parameters for each algorithm is stated. Table II gives the computed results for each algorithm when executed for 50 runs. It is shown that FOA generated the shortest average path with 19.51m when compared with PSO and ACO; while PSO and ACO were in close range with 21.27m and 21.41m respectively. Again, the Convergence speed of FOA can be seen in Table 2 to outperform that of PSO and ACO. FOA was able to converge with a speed of 314921m/s as against that of ACO and PSO which are 51051.93m/s and 3655.371 m/s respectively. However, the execution time of FOA was worst compared with ACO whose execution time outperformed the FOA and ACO algorithm.

Table 1: Parameters used in the experiment.

Algorithm/Parameters	FOA	ACO	PSO
Swarm Size	100	100	100
Number of Runs(NN)	50	50	50
C1 (Learning factor)	*	*	1.49445
C2 (Learning factor)	*	*	1.49445
W (Inertia)	*	*	0.729
Q (Attractive Constant)	*	0.1	*
Alpha(Influence factor)	*	0.1	*
Beta(Inf. of adjacent nodes)	*	0.1	*
P(Evaporation rate)	*	0.1	*

* the parameter is not applicable.

Figs (8-12) give the paths generated by the FOA, PSO and ACO at run 10, 20, 30, 40 and 50. The individual result of the execution runs at 10, 20, 30, 40 and 50 can be seen on Table III. With the individual runs, the performance of FOA in determining the shortest path with a faster speed can also be concluded.

Table 2: Comparison of Results for FOA, ACO and PSO algorithms.

Algorithm	FOA	ACO	PSO
Avg. Path Length (m)	19.5128	21.4148	21.2746
Avg. Execution Time(s)	0.008117	0.000576	0.007203
Avg. Speed (m/sec)	3149.217	51051.93	3655.371

Table 3: Result for Execution No.10, 20, 30, 40 and 50.

Exec. No.	Criteria	FOA	ACO	PSO
10	Path Length	16.41	27.32	15.76
	Exec. Time	0.00563236	0.000354	0.0052055
	Conv. Speed	2913.521153	77219.64	3027.567
20	Path Length	18.33	24	26.72
	Exec. Time	0.0055828	0.000336	0.0051934
	Conv. Speed	3283.298703	71369.1	5144.9521
30	Path Length	21.65	26.86	19.81
	Exec. Time	0.005564	0.000334	0.0057814
	Conv. Speed	3891.08555	80467.35	3426.5057
40	Path Length	18.17	21.39	19.47
	Exec. Time	0.0055883	0.000328	0.0053256
	Conv. Speed	3251.436036	65246.24	3655.9261
50	Path Length	16.57	19.16	21.39
	Exec. Time	0.005766	0.000351	0.0054766
	Conv. Speed	2873.742629	54527.86	3905.7151

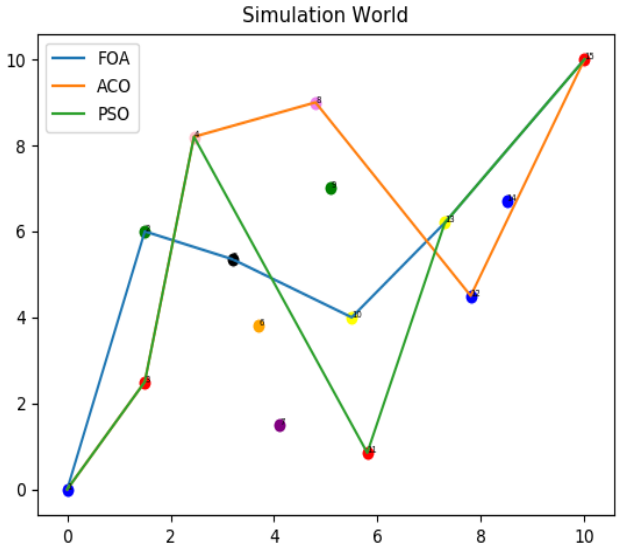

Fig 8: Path generation at execution no. 10.

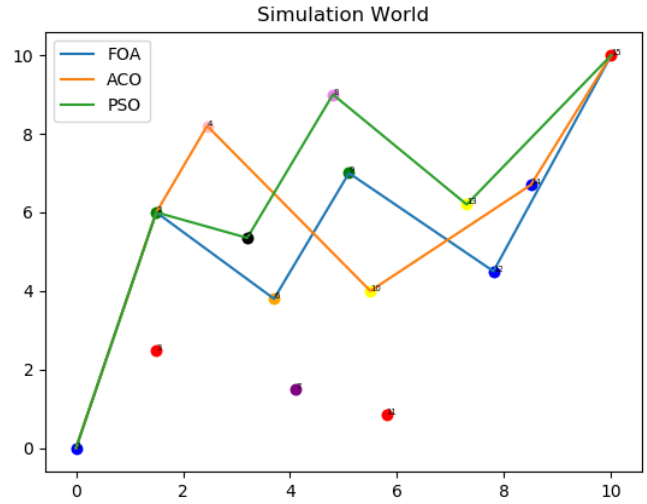

Fig 9: Path generation at execution no. 20.

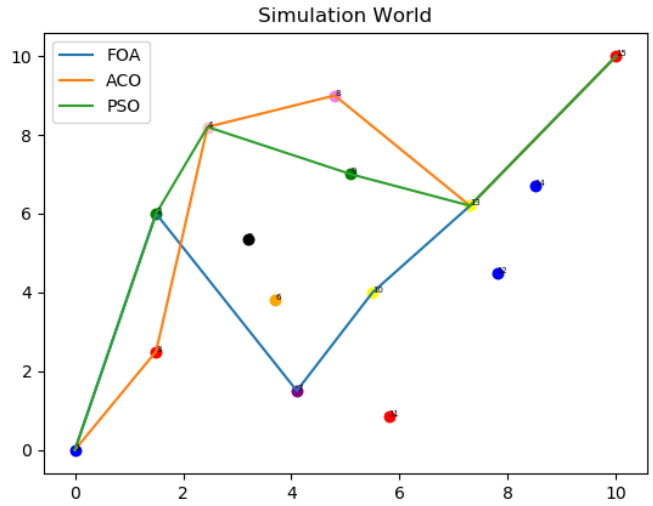

Fig 10: Path generation at execution no. 30.

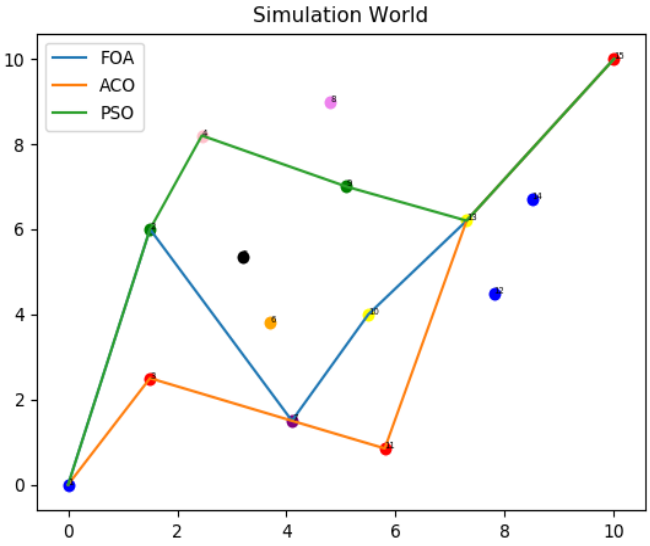

Fig 11: Path generation at execution no. 40.

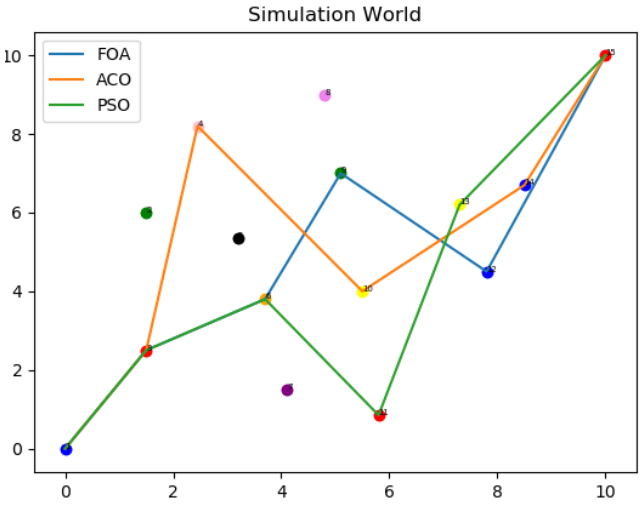

Fig 12: Path generation at execution no. 50.

V. CONCLUSION

In this study we presented the application of three swarm intelligence algorithms namely: Fruit-fly optimization Algorithm (FOA), Particle Swarm Optimization Algorithm (PSO) and Ant Colony Optimization Algorithm (ACO) to the Mobile robot path planning problem in an obstacle free static environment. The three algorithms were able to generate a path from the start node to the target node within the search space. We can conclude that the algorithms have similar execution times, path length and convergence speed irrespective of the number of runs it is executed with same parameters thus the number of runs do not affect the performance of the algorithms. ACO was observed to have the least execution time than FOA and PSO but did not achieve the best path length. With this we can conclude that ACO falls into premature convergence than FOA and PSO. FOA has the highest execution time which tells us about the high processing time due to its feedback mechanism.

We also observed that the basic FOA was able to produce path with shorter length than the basic PSO and ACO counter parties. This could be attributed to simple implementation method FOA implores in the search. FOA which is one of the newest swarm intelligence algorithms in the optimization world is seen to out-perform the older swarm intelligence algorithms in convergence speed. It is also worthy to note that the implementation of FOA is far easier than ACO and PSO due to its minimum parameter. In the next paper, we intend to apply these basic versions of the three algorithms in an environment with obstacles to also compare their performance in such condition.

REFERENCES

AbWahab M. N.; S. Nefti-Meziani and A. Atyabi. (2015). A Comprehensive Review of Swarm Optimization Algorithms. PLOS ONE 10(5): 1-36.

Ajeil F.H.; I. K. Ibraheem and M. A. Sahib. (2020). Multi-objective path planning of an autonomous mobile robot using hybrid PSO-MFB optimization algorithm, Applied Soft Computing Journal, 89, 1-27.

Allah, R. M. (2016). Hybridization of Fruit Fly Optimization Algorithm and Firefly Algorithm for Solving Nonlinear Programming Problems. International Journal of Swarm Intelligence and Evolutionary Computation, 5(2), 1-10.

Blum, C. (2005). Ant Colony Optimization: Introduction and recent trends. Elsevier, Physics of Life Review 2, 353–373.

Cholodowicz E. and Figureurowski D. (2017). Mobile Robot Path Planning with Obstacle Avoidance using Particle Swarm Optimization. Research Gate, DOI: 10.14313/PAR_225/59, 59–68.

Closet H. (2007). Robotic Motion Planning: Cell Decompositions. Available online at: https://www.cs.cmu.edu/~motionplanning/lecture/Chap6-CellDecomp_howie.pdf, Accessed on June 7, 2020.

Connors, J. and Elkaim G. (2007). Manipulating B-Spline Based Paths for Obstacle Avoidance in Autonomous Ground Vehicles, Proceedings of the National Technical Meeting of The Institute of Navigation, San Diego, CA, 1081-1088

Gangadharan M. M. and Salgaonkar A. (2020). Ant colony optimization and firefly algorithms for robotic motion planning in dynamic environments: University of Mumbai, India. Engineering Reports published by John Wiley & Sons, Ltd.

Hazim I. and Mesut, G. (2014). Parameter Analysis on Fruit Fly Optimization Algorithm. Journal of Computer and Communications, 2: 137-141.

Kan E.; M. Lim; S. Yeo; J. Ho and Z. Shao. (2011). Contour Based Path Planning with B-Spline Trajectory Generation for Unmanned Aerial Vehicles (UAVs) over Hostile Terrain. Journal of Intelligent Learning Systems and Applications, 3(3): 122-130. doi: 10.4236/jilsa.2011.33014

Li, Y. and Han, M. (2020). Improved fruit fly algorithm on structural optimization. Brain Informatics, 7(1): 1-13.

Mansi, A. and Priyanka, G. (2013). Path planning of Mobile robots using Bee Colony Algorithm. MIT International Journal of Computer Science & Information Technology, 3(2): 86–89.

Narendra, S. P. and Sanjeev, S. (2013). Robot Path planning using Swarm Intelligence: A Survey. International Journal of Computer Applications 83(12): 0975 – 8887.

Pratap, B. S.; V. R. Harsha and M. Amitabha. (2013). Voronoi Diagram Based Roadmap Motion Planning. Available online at: https://cse.iitk.ac.in/users/cs365/2013/submissions/~prabhanu/cs365/project/report.pdf Accessed on June 7, 2020.

Qinghai, B. (2010). Analysis of Particle Swarm Optimization Algorithm. Computer and Information Science, 3(1): 180-184.

Rizk, M. A. (2016). Hybridization of Fruit Fly Optimization Algorithm and Firefly algorithm for Solving Nonlinear Programming Problems. International Journal of Swarm Intelligence and Evolutionary Computation 5(2): 1-10, DOI: 10.4172/2090-4908.1000134.

Sheng-Xiang, L.; Z. Yu-Rong and W. Lin. (2018). An effective fruit fly optimization algorithm with hybrid information exchange and its applications. International Journal of Machine Learning and Cybernetics. 9 (10): 1623-1648.

Shui-ping, Z.; C. Yang and G. Yang-dan. (2016). Fruit fly algorithm Based on Extremal optimization. 12th International Conference on Computational Intelligence and Security, Chicago, USA. 534-537.

Wen–Tsao, P. (2014). A New Evolutionary Computation – Fruit Fly optimization Algorithm second edition. Taiwan, China: Canghai Press.

Xing, G.; Z. Jian; L. Wei and Z. Yiwen. (2017). A fruit fly optimization algorithm with a traction mechanism and its applications. International journal of distributed sensor network. 13(11): 1-12, DOI: 10.1177/1550147717739831.

Ye, F.; X.Y. Lou and L.F. Sun. (2017). An improved chaotic fruit fly optimization based on a mutation strategy for simultaneous feature selection and parameter optimization for SVM and its applications. PLOS ONE, 12(4): 1-36.

Yue, L. and Chen, H. (2019). Unmanned vehicle path planning using a novel ant colony algorithm EURASIP Journal on Wireless Communications and Networking. 136, 1-9. https://doi.org/10.1186/s13638-019-1474-5

Zhang, L.; L. Liu; X. Yang and Y. Dai. (2016). A Novel Hybrid Firefly Algorithm for Global Optimization.11(9): 1-17.