Int j simul model 18 (2019) 3, 397-407
ISSN 1726-4529 Original scientific paper

OPTIMAL PATH PLANNING FOR AN AUTONOMOUS MOBILE ROBOT USING DRAGONFLY ALGORITHM

Muthukumaran, S. & Sivaramakrishnan, R.
Dept. of Production Technology, MIT-Campus, Anna University, Chennai, Tamilnadu, India
E-mail: smkumaran90@gmail.com, srk@mitindia.edu

Abstract
Navigation, path generation and obstacle avoidance are considered as the key challenges in the area of autonomous mobile robots. In this article, a new meta-heuristic optimization technique called Dragonfly Algorithm (DA) is employed for the navigation of autonomous mobile robot in an unknown cluttered environment filled with several static obstacles. This new meta-heuristic Dragonfly algorithm is inspired from the static and dynamic swarming behaviours of dragonflies in nature. Two objective functions, target seeking and obstacle avoidance are formulated based on the distance between the robot, target and the obstacles and is optimized using the proposed DA for obtaining optimal path. After every iteration, based on the objective function values the robot proceeds towards the globally best agent in the swarm in a sequence of permutation which finally leads to the target. A variety of static environment is modelled and the algorithm is tested both through simulation and experimentally. The proposed algorithm shows that the robot reaches the target without colliding any obstacles while generating a smooth optimal trajectory.
(Received in December 2018, accepted in June 2019. This paper was with the authors 1 month for 2 revisions.)

Key Words: Mobile Robot Navigation, Dragonfly Algorithm, Autonomous Robot, Optimization

1. INTRODUCTION

In the recent era, autonomous mobile robots have found a wide variety of applications covering the areas from military surveillance, space explorations, agriculture to material transportation in logistics. Several warehouses equipped with automated guided vehicles may be soon replaced with autonomous robots, due to its autonomous nature and improvements in the fields of artificial intelligence. Path generation and obstacle avoidance form the key challenges for an autonomous mobile robot.

In recent years, many researchers have implemented many nature / bio-inspired metaheuristic optimization methods for solving mobile robot navigation problems. Nature/bio-inspired algorithms such as Genetic Algorithm (GA) [1, 2], Ant Colony Optimization (ACO) [3], Cuckoo Search (CS) [4], Invasive Weed Optimization (IWO) [5], Particle Swarm Optimization (PSO) [6], Bacteria Forging Algorithm (BFA)[7], Bats Algorithm [8], Simulated Annealing (SA) [9], Grey Wolf Optimizer [10], Bees Colony [11], Cockroach Swarm Algorithm [12], Frog Leaping Algorithm [13], Firefly Algorithm [14], Fruit Fly Algorithm [15] and many other algorithms have been implemented for solving navigational strategies in autonomous mobile robots. Few vision based solutions have also been proposed to solve the path planning problems [16-18]. All the above-mentioned algorithms have their own advantages and disadvantages.

Genetic algorithm is one of the most famous optimization algorithm based on the evolution methods of natural selection, crossover and mutation. GA is widely used by many researchers for solving the navigation problem. But GA suffers from problems like poor convergence rate, no guarantee of the optimal solution, and time-consuming procedures for choosing mutation rate and population size [19]. ACO is one of the most common probabilistic technique for solving computational intelligence problems inspired by food searching technique for ants. Many researchers have successfully implemented ACO for

solving path planning problems. Dynamic Convergence property of ACO is way more efficient than the GA [19]. PSO is one of the most widely used metaheuristic optimization algorithm inspired by the social behaviour of the flock of birds. PSO is one of the most used bio-inspired methods for solving mobile robot navigation problems. The only drawback in PSO is that there might be premature convergence and local minima trapping when the environment is complex. All the above-mentioned algorithms are also hybridized using methods like Fuzzy Inference systems and Neural Networks [20] to improve the efficiency of the system [21]. Dragonfly algorithm is one such new meta-heuristic algorithm based on the swarming behaviour of dragonflies. It is clearly evident that DA outperforms PSO and GA in several unimodal test functions [22].

1.1 Novelty of the proposed work

In this paper, a systematic effort is taken for solving navigational problems of the autonomous mobile robot by employing Dragonfly algorithm. But to the author's knowledge, the Dragonfly Algorithm is not yet applied for solving path planning problems of autonomous mobile robots. Finally, some simulation and experimental studies are performed to validate the feasibility of the proposed algorithm in the area of mobile robot navigation.

1.2 Dragonfly algorithm

In 2015, Seyedali Mirjalili proposed the concept of Dragonfly algorithm for solving multi-objective optimization problems [22]. Dragonfly algorithm is based on the exceptional and intelligent swarming behaviour of dragonflies which forms the key characteristics of hunting and migration [22]. Static swarming or hunting is regarded as the formation of a small group of dragonflies, rapidly shifting the steps for food. Dynamic swarm or migratory swarm is regarded as the very large number of dragonflies flying over long distances for migration [22, 23]. Static Swarm represents the exploitation phase of the dragonfly algorithm while dynamic swarms represent the exploration capabilities of DA. The swarming behaviours of dragonfly follow the ideas of separation, alignment, cohesion, attraction towards the food and distraction from the enemies as suggested by Reynolds in 1987. Each dragonfly in the swarm corresponds to the solution in the search space and all the above-mentioned parameters directly influence the positioning of dragonflies in the group of the swarm. The fitness function is assessed based on the velocities and positions of the dragonflies.

2. MATHEMATICAL MODELLING OF ARTIFICIAL DRAGONFLIES

2.1 Population size *N*

Let *N* be the size of the dragonflies (number of individuals), the position of i^{th} dragonfly is defined as,

$$X_i = (x_i^1, x_i^d, \ldots \ldots, x_i^N) \tag{1}$$

where i =1, 2, 3..., *N*, x_i^d refers to the position of the i^{th} dragonfly in d^{th} dimension of the search space and *N* is the number of search agents.

2.2 Separation $S_{(i,t)}$

Internal collision avoidance of individuals in the neighbourhood:

$$S_{(i,t)} = -\sum_{j=1}^{N} X_{(i,t)} - X_{(j,t)} \tag{2}$$

where $S_{(i,t)}$ is the separation motion for an i^{th} individual at the t^{th} iteration; $X_{(i,t)}$ is the position of the current individual i at the t^{th} iteration; $X_{(j,t)}$ is the position of the neighboring individual j at the t^{th} iteration.

2.3 Alignment $A_{(i,t)}$

Velocity matching of individuals among the neighbourhood:

$$A_{(i,t)} = \frac{\sum_{j=1}^{N} V_{(j,t)}}{N} \tag{3}$$

where $A_{(i,t)}$ is the alignment motion of the current individual i at the t^{th} iteration; $V_{(j,t)}$ is the velocity of the neighboring individual j at the t^{th} iteration.

2.4 Cohesion $C_{(i,t)}$

The tendency of individuals towards the centre of the mass of the neighbourhood:

$$C_{(i,t)} = \frac{\sum_{j=1}^{N} X_{(j,t)}}{N} - X_{(i,t)} \tag{4}$$

where $C_{(i,t)}$ is the cohesion motion of the current individual i at the t^{th} iteration.

2.5 Attraction $F_{(i,t)}$

The motion of attraction towards a food source:

$$F_{(i,t)} = X_{(food,t)} - X_{(i,t)} \tag{5}$$

where $X_{(food,t)}$ is the food source position at the t^{th} iteration; $F_{(i,t)}$ is the food attraction motion of the current individual i at the t^{th} iteration. Individual dragonfly with best objective function up to the current iteration will be considered as food.

2.6 Distraction $E_{(i,t)}$

The motion of distraction outwards an enemy:

$$E_{(i,t)} = X_{(enemy,t)} + X_{(i,t)} \tag{6}$$

where $X_{(enemy,t)}$ is the enemy position at the t^{th} iteration; $E_{(i,t)}$ is the enemy distraction motion of the current individual i at the t^{th} iteration. Individual dragonfly with worst objective function up to the current iteration will be considered as an enemy.

2.7 Step vector and position vector

The position updating features of the artificial dragonfly individuals are analogous to step vector updating feature of the Particle Swarm Optimization (PSO) technique. Step vector ($\Delta X_{(i,t)}$) and position vector ($X_{(i,t)}$) are controlled to update the position of artificial dragonflies in a search space and control their movements. The step vector gives the direction of movement of the dragonflies, while the position vector provides the position of the individual dragonflies in the search space. The step vector can be calculated as given below:

$$\Delta X_{(i,t+1)} = \left(s * S_{(i,t)} + a * A_{(i,t)} + c * C_{(i,t)} + f * F_{(i,t)} + e * E_{(i,t)}\right) + w * \Delta X_{(i,t)} \tag{7}$$

where s indicates separation weight; a indicates alignment weight; c is cohesion weight; f shows food factor; e indicates enemy factor; w indicates inertia weight. Initially suitable weights are assigned randomly to each operator and they are adaptively tuned to guarantee the convergence of dragonflies towards a best possible solution. After computing the step vector, position vectors can be computed using:

$$X_{(i,t+1)} = X_{(i,t)} + \Delta X_{(i,t)} \tag{8}$$

The neighbouring radius of the dragonfly also increases accordingly as the optimization progresses. If the dragonfly has at least one neighbourhood, then $\Delta X_{(i,t+1)}$ and $X_{(i,t+1)}$ are used to update its position and velocity. If there are no dragonflies in the neighbourhood then levy flight technique will be implemented to update the position and velocity of the dragonflies. Employment of Levy flight also enhances the randomness, chaotic behaviour and greatly improves the global search capabilities.

$$X_{(i,t+1)} = X_{(i,t)} + Levy(d) \times X_{(i,t)} \tag{9}$$

where d is the number of decision variables; $Levy(d)$ is the Levy flight function.

3. OBJECTIVE FUNCTION FORMULATION USING DRAGONFLY ALGORITHM

The prime objective considered in this paper is to navigate an autonomous mobile robot in a cluttered environment with various static obstacles. The mobile robot should sense the environment for obstacles and perform effective obstacle avoidance while reaching the target in an optimized smooth trajectory.

The path planning problem is transformed into a minimization problem and accordingly, two suitable objective functions are formulated based on the distances between the robot, obstacles and the target. The first objective function should enable the robot to traverse the environment while efficiently missing the obstacles. The second objective function should traverse the robot to the target in the shortest possible path. Consider the below Fig. 1 as a robot environment in which the robot starting point and the robot goal point are clearly depicted.

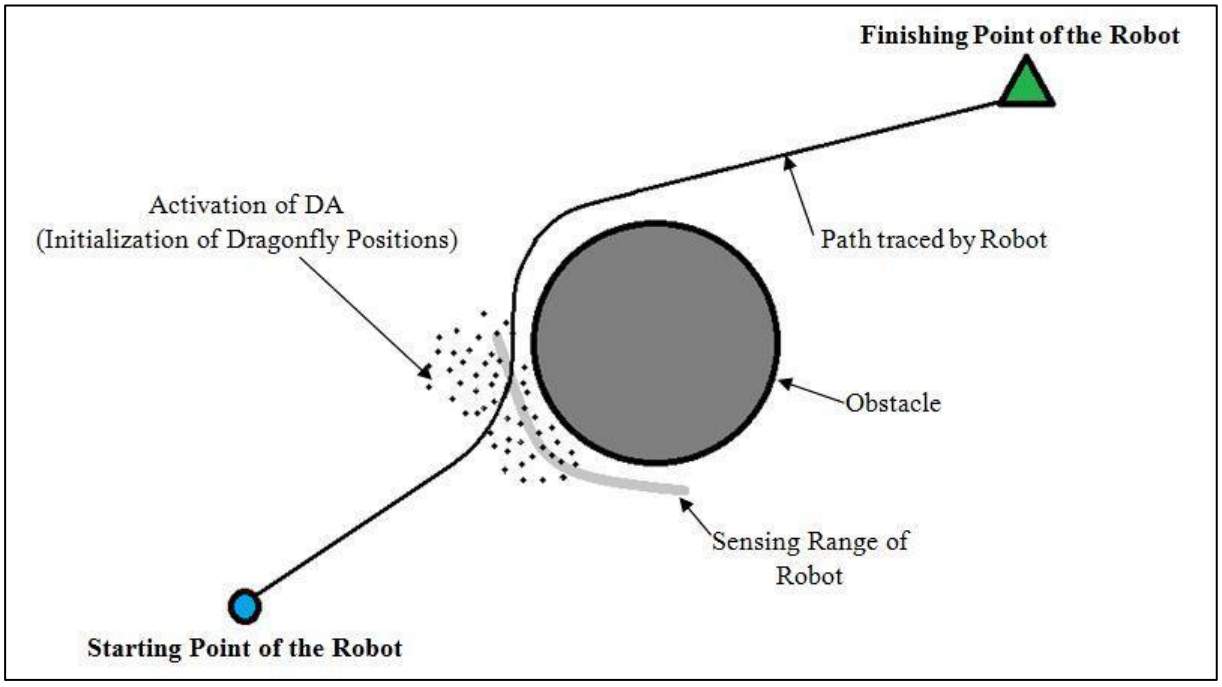

Figure 1: Activation of the Dragonfly Algorithm.

The black line is the preferred route for the robot to reach the goal with optimal path length. While moving in the environment, when the robot reaches near an obstacle, its sensors detect the obstacles and sense its range. Then the dragonfly algorithm is initialized to avoid the obstacles in its path. The dragonfly algorithm will find the best next position (based on the objective function value) for the robot to move to avoid collision with the obstacles while

travelling towards the goal. In the dragonfly algorithm, the food source is chosen from the best solution that the whole swarm is found. Therefore the quality of the food source is directly proportional to the optimized path length of the robot. Each step to be moved by the robot is based on the distance between the position of the food source to the goal and the obstacles present in its environment. Two important behaviours are considered here for effective path planning of mobile robot.

3.1 Obstacle avoidance behaviour

The obstacle avoidance character is modelled to avoid the collision of the robot with the obstacles present in the environment. The position of the food source (global best position) should be maintained at a maximum safe distance from the nearest obstacle. The objective function is derived as the Euclidean distance between the global best and the nearest obstacle in the environment.

$$(Dist.)_{F-OB} = \sqrt{\left(x_{OB} - x_{F_i}\right)^2 + \left(y_{OB} - y_{F_i}\right)^2} \tag{10}$$

where x_{F_i} and y_{F_i} are the global best (food source) positions; x_{OB} and y_{OB} are the nearest obstacle positions; $(Dist.)_{F-OB}$ is the Euclidean distance from the global best (food source) to the nearest obstacle. The nearest obstacle to the robot can be calculated using:

$$(Dist.)_{R-OB} = \sqrt{\left(x_{OB_n} - x_R\right)^2 + \left(y_{OB_n} - y_R\right)^2} \tag{11}$$

3.2 Target seeking behaviour

The position of the food source (global best position) should be maintained at a minimum distance from the goal. The objective function is termed as the Euclidean distance between the global best (food source) position and the goal in the environment.

$$(Dist.)_{F-G} = \sqrt{\left(x_G - x_{F_i}\right)^2 + \left(y_G - y_{F_i}\right)^2} \tag{13}$$

where x_G and y_G are the goal positions. $(Dist.)_{F-G}$ is the minimum Euclidean distance from the food source position to the robot.

Combining the above two behaviours, the objective function of the path optimization problem can be expressed as follows:

$$Objective\ function(f_1) = C_1 \frac{1}{\min_{OB_j \epsilon OB_d} \left\| Dist_{F-OB_d} \right\|} + C_2 . \| Dist_{F-G} \| \tag{14}$$

Whenever the robot moves in the environment, the robot may encounter several obstacles in the environment and are represented as $OB_d \in \{OB_1, OB_2, \ldots, OB_n\}$. It is evident from the objective function that when F_i is nearer to the goal, the objective function value of $\| Dist_{F-G} \|$ will be reduced and when F_i is away from the obstacles, the objective function value of $\min_{OB_j \epsilon OB_d} \| Dist_{F-OB_d} \|$ will be large. From this, it is evident that the path planning problem is a minimization problem. C_1 and C_2 found in the objective function are called as fitting/controlling parameters and it is clear that these parameters have a direct control over the trajectory of the robot. If C_1 is large, the robot will be far away from the obstacles or if C_1 is too small, then the robot might easily collide with the obstacle present in the environment. Similarly, if C_2 is large then the robot has higher the chances of travelling towards the goal in a short/optimal path, else it will result in larger paths. These control parameters will result in faster convergence property of objective function and elimination of local minima. In this work, the control parameters are chosen by trial and error methods.

3.3 Dragonfly algorithm for mobile robot navigation

1. Initialize the start and goal position of the robot.
2. The robot directly moves towards the goal until it is encountered by an obstacle.
3. When the robot senses an obstacle in its periphery, implement dragonfly algorithm.
4. Initialize the dragonflies' population and initialize step vectors.
5. Calculate the fitness functions of all dragonflies and accordingly the best of the dragonfly with highest fitness function is found.
6. Update *w*, *s*, *a*, *c*, *f*, and *e* and then calculate *S*, *A*, *C*, *F*, and *E*.
7. Update neighbouring radius, velocity vector, and a position vector.
8. The robot will proceed towards the global best position.
9. Repeat the steps 2-8 until the robot avoids all the obstacles and reaches its goal.
   Note: Here the term velocity defines the distance travelled by the robot per iteration.

4. DEMONSTRATIONS OF THE PROPOSED PATH PLANNING ALGORITHM

The DA path planning algorithm has been validated using both simulations and experimental scenarios with partially or totally unknown environments. All the simulations are verified in the MATLAB environment. A laptop with Intel Core i3-2330M processor 2.20 GHz with 4 GB memory and running Windows 7 Home Basic (64-bit) is used for simulation of the experiments. An experimental arrangement containing static obstacles is set up in the laboratory, to evaluate the effectiveness of the proposed algorithm. Finally, the experimental results are compared with the simulated results to verify the efficacy of the proposed path planning algorithm.

4.1 Simulation results

A series of simulation tests have been conducted with the MATLAB software to validate the effectiveness and the robustness of the above algorithm. The modelled workspace with the source, target and obstacles are shown in Figs. 2, 3, 4 and 5. The experiment verifies that the robot travels to the target by avoiding the obstacles in a smooth trajectory while covering optimal path length. The parameters for the Dragonfly algorithm, used in the simulation are listed in Table I. The path lengths traced by the mobile robot in different environments are depicted in Table II.

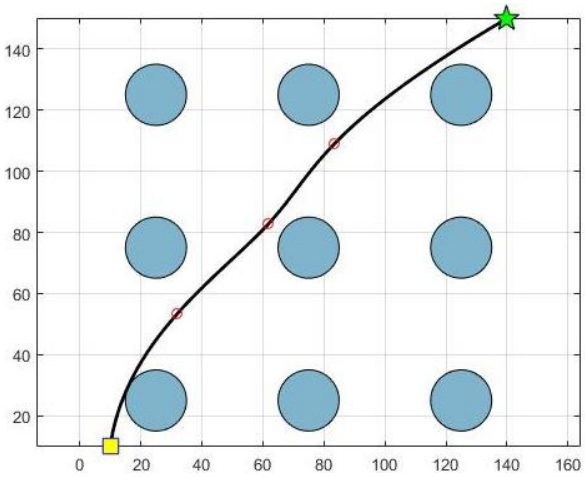

Figure 2: Simulation results of Scene 1.

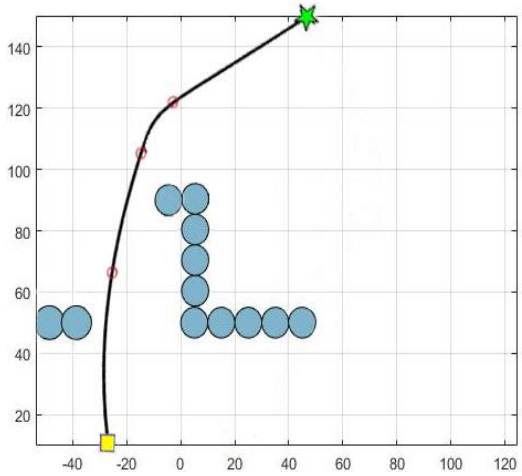

Figure 3: Simulation results of Scene 2.

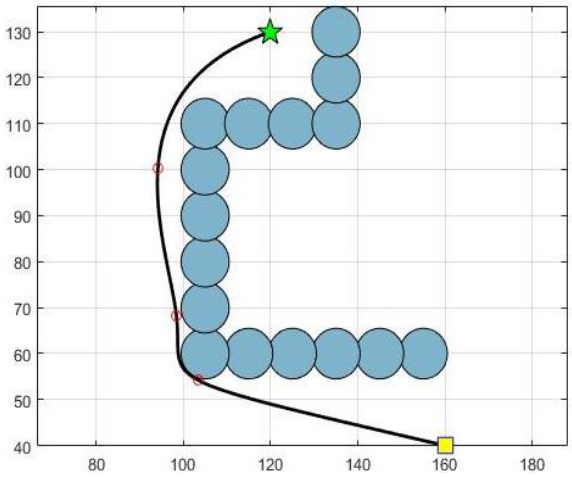

Figure 4: Simulation results of Scene 3.

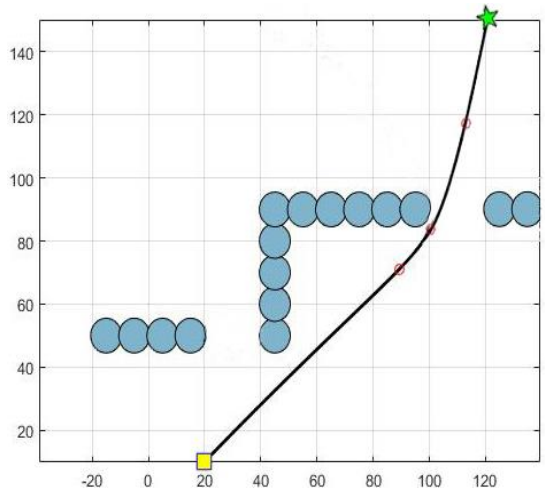

Figure 5: Simulation results of Scene 4.

Table I: Assumed values for the parameters.

Symbol	Description	Value
N	Dragonflies population size	30
t	Iteration count	350
s	Separation weight	0.1
a	Alignment weight	0.1
c	Cohesion weight	0.7
f	Food factor	1
e	Enemy factor	1
w	Inertia weight	0.9-0.4
r_1, r_2	Random values	$[0, 1]$
$£$	Constant	1.5
C_1	Controlling parameter 1	1
C_2	Controlling parameter 2	1×10^{-6}

4.2 Experimental results

A mobile robot has been designed and fabricated to test the above path planning algorithm. The robot has 5 IR sharp sensors and 8 IR proximity sensors to sense the information about the arena in which the mobile robot navigates. An IR array is placed at the bottom of the robot to detect the source and destination in the environment. Two DC motors with inbuilt quadrature encoders are used to drive the differential drive robot. Figs. 6, 7, 8 and 9 show the efficacy of the proposed path planning algorithm. The autonomous mobile robot navigates in the environment in a smooth trajectory while covering an optimal path in the given environment. The experiments prove that the robot path traced by the robot is very similar and close to that of the simulation results.

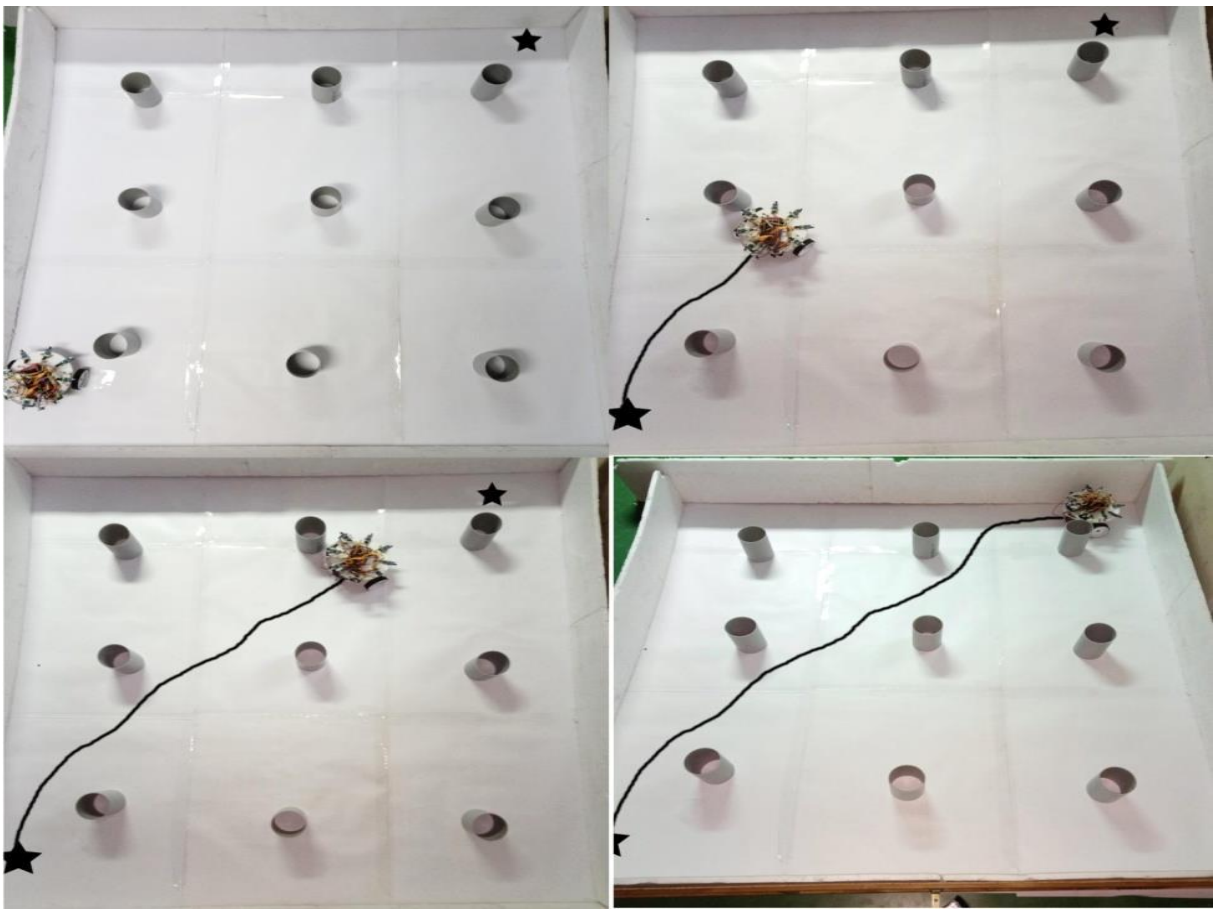

Figure 6: Experimental results for Scene 1.

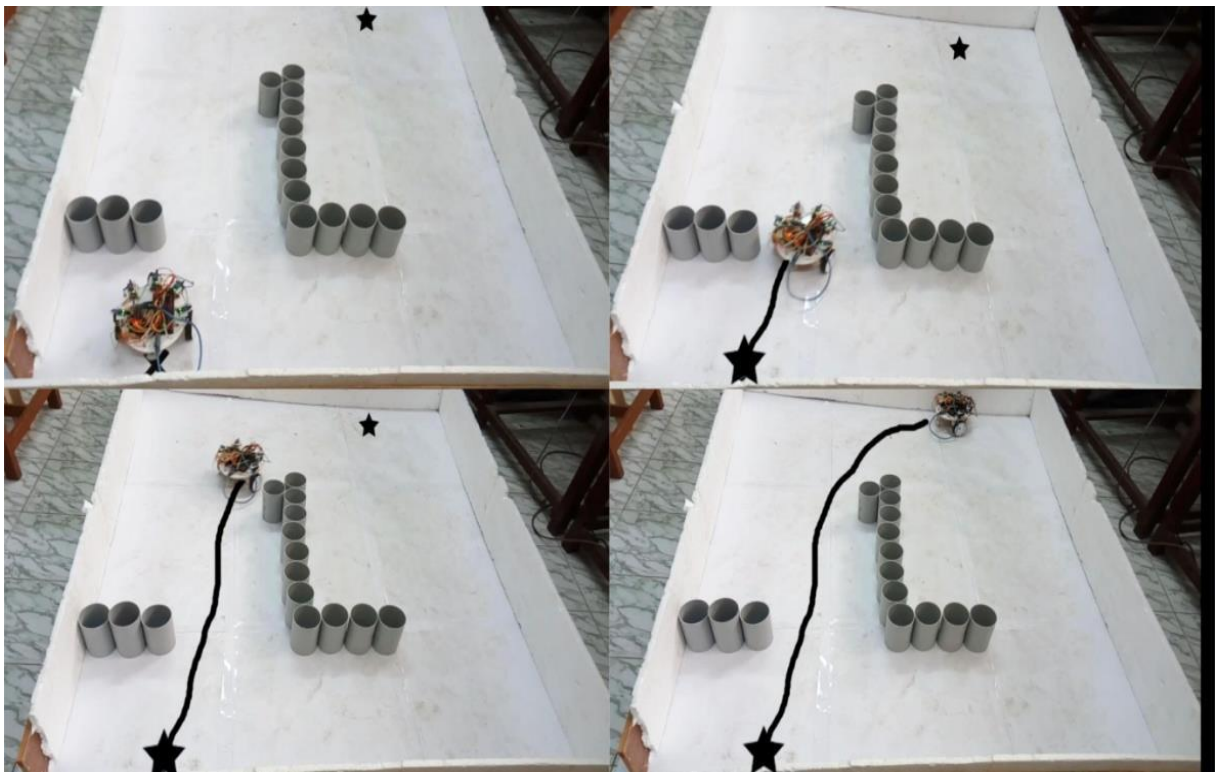

Figure 7: Experimental results for Scene 2.

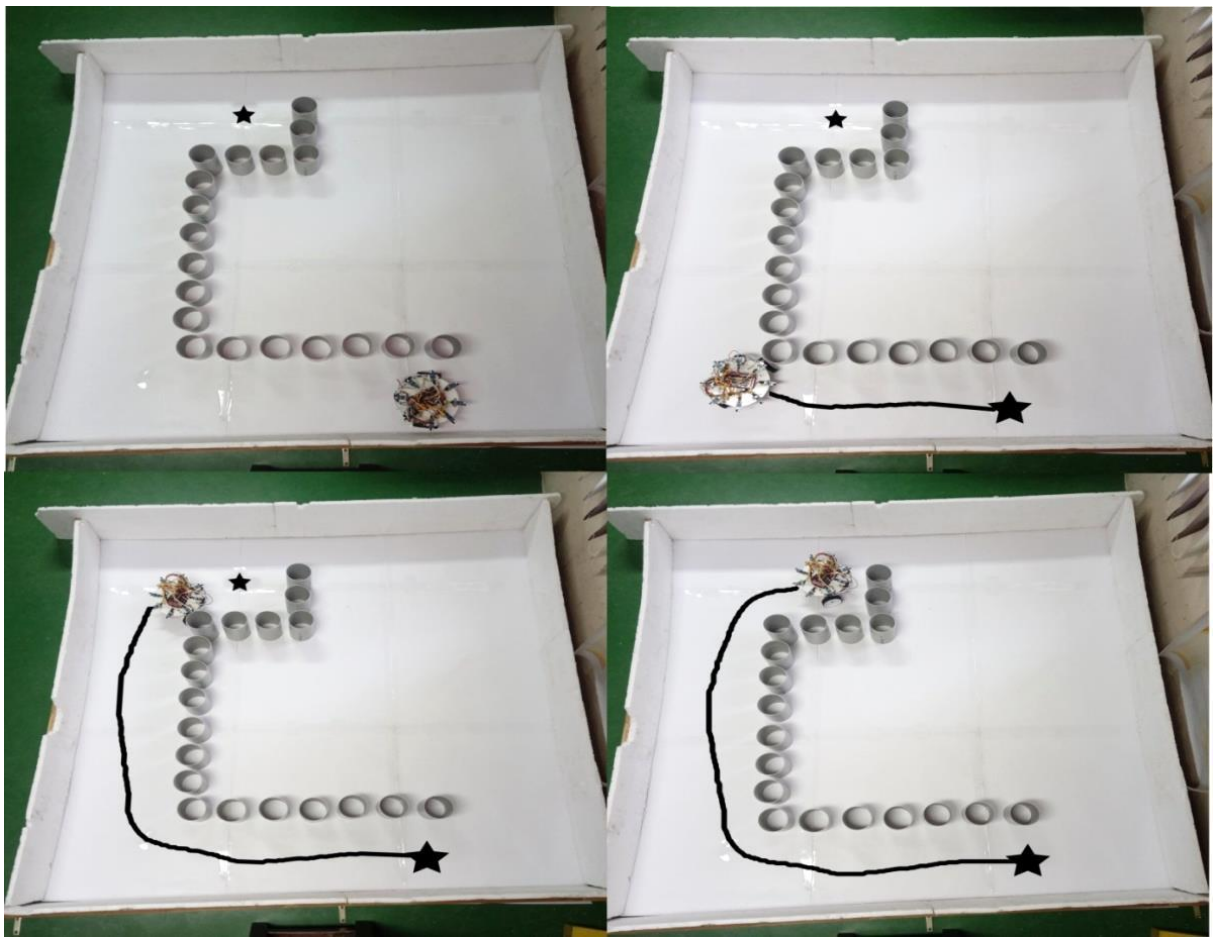

Figure 8: Experimental results for Scene 3.

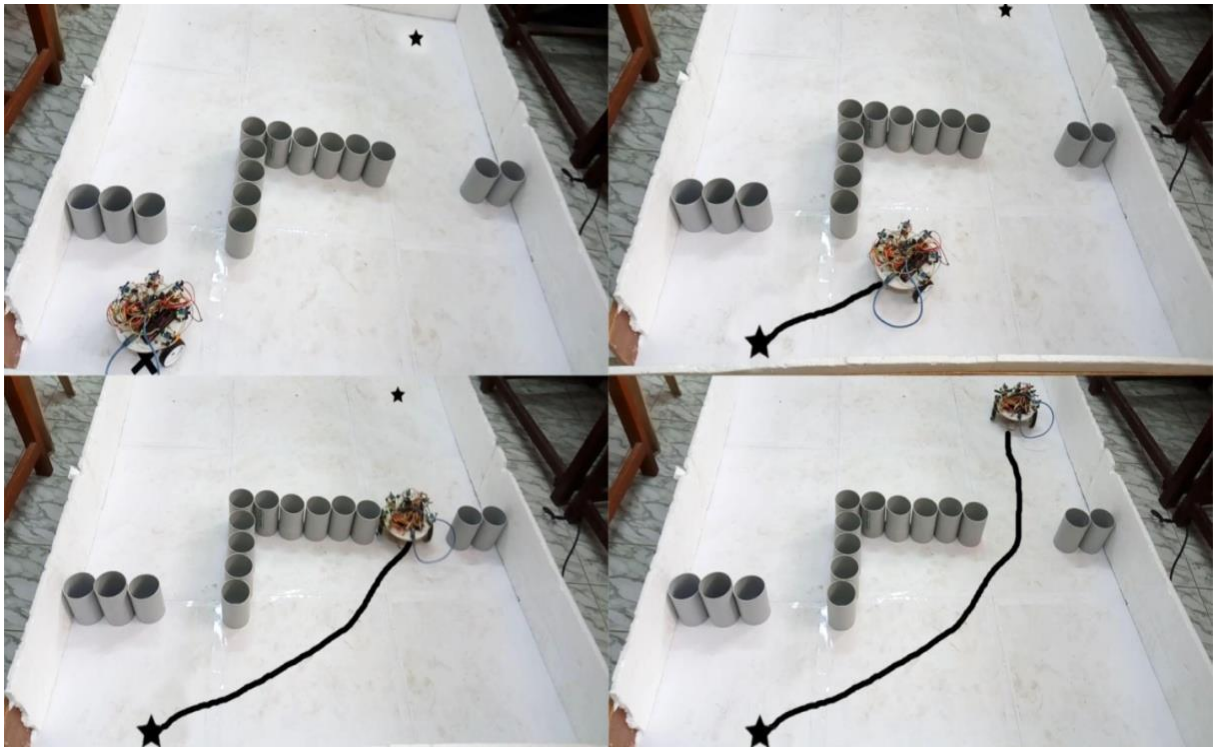

Figure 9: Experimental results for Scene 4.

Table II: Analysis and comparison of simulated and experimental results.

Scenario	Simulated result		Experimental result		Average error deviation (%)	
	Average path length after 10 runs (cm)	**Average time taken after 10 runs** (sec)	**Average path length after 10 runs** (cm)	**Average time taken after 10 runs** (sec)	**Path length**	**Time taken**
Scene 1	205	18.23	197	18.97	3.9	4.05
Scene 2	185	17.83	176	18.54	4.8	3.98
Scene 3	216	19.21	206	20.12	4.62	4.7
Scene 4	181	17.25	172	17.92	4.97	3.88

5.CONCLUSIONS AND FUTURE WORK

In this proposed work, a new meta-heuristic algorithm based on the swarming behaviour of Dragonfly is introduced for solving navigational problems in the areas of autonomous mobile robotics. After every iteration, the global best position is found, and the robot moves accordingly towards the global best position until the target location is reached. Hence, the trajectory is generated using the above global best positions of every iteration. Several experiments are carried out to fine tune the controlling parameters which are directly proportional to the smoothness of the trajectory formed. Using the proposed algorithm, the robot is capable of reaching the target by avoiding the obstacles, escaping the traps while achieving a smooth trajectory and achieved a much-optimized path.

The capability of the proposed algorithm is validated both through experimentation and simulation. Both the experimental results and simulated results are compared in terms of path length and time. Both simulation and experimental results match with each other with an average mean error of less than 5 %. Though the proposed algorithm proved to be versatile and very robust for local path planning, still an implementation of reinforcement learning will lead to better results. Further, in this proposed work, only path planning of the single mobile robot with static obstacles is taken into consideration and the capabilities of the algorithm are evaluated. In the future, the work will be extended with multiple mobile robots and dynamic obstacles (moving obstacles).

REFERENCES

[1] Manikas, T. W.; Ashenayi, K.; Wainwright, R. L. (2007). Genetic algorithms for autonomous robot navigation, *IEEE Instrumentation and Measurement Magazine*, Vol. 10, No. 6, 26-31, doi:10.1109/MIM.2007.4428579

[2] Lamini, C.; Benhlima, S.; Elbekri, A. (2018). Genetic algorithm based approach for autonomous mobile robot path planning, *Procedia Computer Science*, Vol. 127, 180-189, doi:10.1016/J.PROCS.2018.01.113

[3] Ganapathy, V.; Sudhakara, P.; Jia Jie, T. T.; Parasuraman, S. (2016). Mobile robot navigation using amended ant colony optimization algorithm, *Indian Journal of Science & Technology*, Vol. 9, No. 45, Paper 102431, 10 pages, doi:10.17485/ijst/2016/v9i45/102431

[4] Mohanty, P. K.; Parhi, D. R. (2015). A new hybrid optimization algorithm for multiple mobile robots navigation based on the CS-ANFIS approach, *Memetic Computing*, Vol. 7, No. 4, 255-273, doi:10.1007/s12293-015-0160-3

[5] Mohanty, P. K.; Parhi, D. R. (2014). A new efficient optimal path planner for mobile robot based on invasive weed optimization algorithm, *Frontiers of Mechanical Engineering*, Vol. 9, No. 4, 317-330, doi:10.1007/s11465-014-0304-z

[6] Masehian, E.; Sedighizadeh, D. (2010). Multi-objective PSO and NPSO based algorithms for robot path planning, *Advances in Electrical and Computer Engineering*, Vol. 10, No. 4, 69-76, doi:10.4316/aece.2010.04011
[7] Montiel, O.; Orozco-Rosas, U.; Sepúlveda, R. (2015). Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles, *Expert Systems with Applications*, Vol. 42, No. 12, 5177-5191, doi:10.1016/j.eswa.2015.02.033
[8] Kar, A. K. (2016). Bio inspired computing: A review of algorithms and scope of applications, *Expert Systems with Applications*, Vol. 59, 20-32, doi:10.1016/j.eswa.2016.04.018
[9] Panigrahi, P. K.; Ghosh, S.; Parhi, D. R. (2014). Comparison of GSA, SA and PSO based intelligent controllers for path planning of mobile robot in unknown environment, *International Journal of Electrical and Computer Engineering*, Vol. 8, No. 10, 1633-1642
[10] Tsai, P.-W.; Nguyen, T.-T.; Dao, T.-K. (2016). Robot path planning optimization based on multiobjective grey wolf optimizer, *Proceedings of the ICGEC 2016: Genetic and Evolutionary Computing*, 166-173, doi:10.1007/978-3-319-48490-7_20
[11] Lin, T.-C.; Chen, C.-C.; Lin, C.-J. (2018). Wall-following and navigation control of mobile robot using reinforcement learning based on dynamic group artificial bee colony, *Journal of Intelligent & Robotic Systems*, Vol. 92, No. 2, 343-357, doi:10.1007/s10846-017-0743-y
[12] Liu, W.; Cheng, L. (2012). A new cockroach swarm optimization for motion planning of mobile robot, *Applied Mechanics and Materials*, Vol. 263-266, 834-838, doi:10.4028/www.scientific.net/AMM.263-266.834
[13] Hidalgo-Paniagua, A.; Vega-Rodríguez, M. A.; Ferruz, J.; Pavón, N. (2015). MOSFLA-MRPP: Multi-objective shuffled frog-leaping algorithm applied to mobile robot path planning, *Engineering Applications of Artificial Intelligence*, Vol. 44, 123-136, doi:10.1016/j.engappai.2015.05.011
[14] Patle, B. K.; Parhi, D.; Jagadeesh, A.; Sahu, O. P. (2017). Real time navigation approach for mobile robot, Vol. 12, No. 2, 135-142, doi:10.17706/jcp.12.2.135-142
[15] Jiang, T.; Wang, J. Z. (2014). Study on path planning method for mobile robot based on fruit fly optimization algorithm, *Applied Mechanics and Materials*, Vol. 536-537, 970-973, doi:10.4028/www.scientific.net/AMM.536-537.970
[16] Finžgar, M.; Podržaj, P. (2017). Machine-vision-based human-oriented mobile robots: A review, *Strojniski vestnik – Journal of Mechanical Engineering*, Vol. 63, No. 5, 331-348, doi:10.5545/sv-jme.2017.4324
[17] Zolghadr, J.; Cai, Y. (2015). Locating a two-wheeled robot using extended Kalman filter, *Tehnicki vjesnik – Technical Gazette*, Vol. 22, No. 6, 1481-1488, doi:10.17559/tv-20140531190647
[18] Sudharsan, J.; Karunamoorthy, L. (2016). Path planning and co-simulation control of 8 DOF anthropomorphic robotic arm, *International Journal of Simulation Modelling*, Vol. 15, No. 2, 302-312, doi:10.2507/IJSIMM15(2)9.339
[19] Pandey, A.; Pandey, S.; Parhi, D. (2017). Mobile robot navigation and obstacle avoidance techniques: A review, *International Robotics & Automation Journal*, Vol. 2, No. 3, 96-105, doi:10.15406/iratj.2017.02.00023
[20] Švaco, M.; Šekoranja, B.; Šuligoj, F.; Vidaković, J.; Jerbić, B.; Chudy, D. (2017). A novel robotic neuronavigation system: RONNA G3, *Strojniski vestnik – Journal of Mechanical Engineering*, Vol. 63, No. 12, 725-735, doi:10.5545/sv-jme.2017.4649
[21] Bozic, M.; Ducic, N.; Djordjevic, G.; Slavkovic, R. (2017). Optimization of Wheg robot running with simulation of neuro-fuzzy control, *International Journal of Simulation Modelling*, Vol. 16, No. 1, 19-30, doi:10.2507/IJSIMM16(1)2.363
[22] Mirjalili, S. (2016). Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems, *Neural Computing and Applications*, Vol. 27, No. 4, 1053-1073, doi:10.1007/s00521-015-1920-1
[23] Sree Ranjini, K. S.; Murugan, S. (2017). Memory based hybrid dragonfly algorithm for numerical optimization problems, *Expert Systems with Applications*, Vol. 83, 63-78, doi:10.1016/j.eswa.2017.04.033