Optimal Path Planning For Intelligent Mobile Robot Navigation Using Modified Particle Swarm Optimization

Nadia Adnan Shiltagh, Lana Dalawr Jalal

Abstract - This study investigates the application of Modified Particle Swarm Optimization (MPSO) to the problem of mobile robot navigation to determine the shortest feasible path with the minimum time required to move from a starting position to a target position in working environment with obstacles. In this study, MPSO is developed to increase the capability of the optimized algorithms for a global path planning. The proposed algorithms read the map of the environment which expressed by grid model and then creates an optimal or near optimal collision free path. The effectiveness of the proposed optimized algorithm for mobile robot path planning is demonstrated by simulation studies. The programs are written in MATLAB R2012a and run on a computer with 2.5 GHz Intel Core i5 and 6 GB RAM.

Improvements presented in MPSO are mainly trying to address the problem of premature convergence associated with the original PSO. In the MPSO an error factor is modelled to ensure the PSO converges. MPSO try to address another problem which is the population may contain many infeasible paths; a modified procedure is carrying out in the MPSO to solve the infeasible path problem.

The results demonstrate that this algorithm have a great potential to solve the path planning with satisfactory results in terms of minimizing distance and execution time.

Index Terms - Modified Particle Swarm Optimization, Global Path Planning, Robot Navigation, Intelligent Mobile Robot.

I. INTRODUCTION

The path planning problem for mobile robots has been an active research area for many years which is started from mid-1970. Many methods have been proposed to address this problem; each method has its own strengths and weaknesses. The choice of a good method is necessary in order to achieve both quality and efficiency of a search [1]. Accordingly, a desirable path should result from not letting the robot waste time taking unnecessary steps or becoming stuck in local minimum positions. Furthermore, a desirable path should avoid all known obstacles in the area [2].

[3] describes the various techniques applied for navigation of an intelligent mobile robot. They showed that the heuristic approaches (Fuzzy logic (*FL*), Artificial Neural Networks (*ANN*), Neuro-Fuzzy, Genetic Algorithm (*GA*), Particle Swam Optimization (*PSO*), Ant Colony Optimization (*ACO*) and Artificial Immune System) gave suitable and effective results for mobile robot navigation (target reaching and obstacle-avoidance).

Manuscript received on April, 2013.

Nadia Adnan Shiltagh, Computer Engineering, University of Baghdad, Baghdad, Iraq.

Lana Dalawr Jalal, Electrical Department, Faculty of Engineering, University of Sulaimani, Sulaimani, Kurdistan Region, Iraq.

Using the heuristic approach, the mobile robot can navigate safely among the obstacles without hitting them and reach the predefined target point. These techniques are also helpful for the solution of the local minima problem. Researchers have been seeking for more efficient ways to solve this problem, in the following section, the recent works on robot's navigation and path planning using particle swarm are reviewed.

[4] provided an overview of the research progress in path planning of a mobile robot for off-line as well as on-line environments. Commonly used classic and evolutionary approaches of path planning of mobile robots have been discussed, and showed that the evolutionary optimization algorithms are computationally efficient. Also, challenges involved in developing a computationally efficient path planning algorithm are addressed.

[5] proposed a modified particle swarm optimization algorithm for the robot path planning in dynamic environment, two parameters of particle-distribution-degree and particle dimension-distance are introduced into the proposed algorithm in order to avoid premature convergence.

[6] provided an intelligent approach for the navigation of a mobile robot in unknown environments, the navigation problem becomes an optimization problem, and then it is solved by *PSO* algorithm. Based on position of goal, an evaluation function for every particle in *PSO* is calculated. It's assumed that Robot can detect only obstacles in a limited radius of surrounding with its sensors. Environment is supposed to be dynamic and obstacles can be fixed or movable.

[7] proposed an Immune Particle Swarm Optimization (*IPSO*) algorithm for path planning of the mobile robot which based on the biological mechanism of the immune system. They compared the simulation results with *PSO* optimization results. They concluded that the optimal path and the execution time based on *IPSO* algorithm are reduced separately, and the improved *PSO* algorithm enhances the convergence speed and robustness of time-varying parameters.

From the above literature review for the recently published paper, it is concluded that various works of research have been successfully applied *PSO* to solve the mobile robot path planning problem, due to its simplicity and efficiency in navigating large search spaces for optimal solutions.

II. NAVIGATIONAL PLANNING FOR MOBILE ROBOTS

The navigational planning problem persists in both static and dynamic environments. In a static environment, the position of obstacles is fixed, while in a dynamic environment the obstacles may move at arbitrary directions with varying speeds, lower than the maximum speed of the robot [8].

Navigation consists of two essential components known as localization and planning. Localization in robotics refers to the ability of determining accurate positions in the search space according to the environmental perceptions gathered by sensors. Planning is considered as the computation of a path through a map which represents the environment. As such, a reliable map is essential for navigation without which robots would not be able to accomplish the goals [9].

Navigation and obstacle avoidance are very important issues for the successful use of an autonomous mobile robot [10]. All obstacle avoidance approaches find a path from an actual position of the robot to a desired goal position, while all these parameters stand as the inputs of the algorithm; the output is the optimal path from start to goal [11]. Efficient navigation of mobile robot means generation of collision free path.

Optimal path planning is an important concern in navigation of autonomous mobile robots, which is to find an optimal path according to some criteria such as distance, time or energy while distance or time being the most commonly adopted criterion [12].

Three major concerns in regard to robot navigation problems are efficiency, safety and accuracy. The main concern of efficiency of the algorithm is to find the destination in a short time. The safety issue is another critical part of the algorithm, since a feasible path should avoid all known obstacles in the area. Once the optimum collision-free path is constructed, then the robot should accurately follow the pre-determined path.

III. PATH PLANNING

Path planning research of autonomous mobile robot has attracted attention since the 1970s. Research in this area has increased due to the reason that mobile robots are now applied in various applications [13]. The main goal of the robot path planning problem is to construct a collision-free path from a starting position to destination position and satisfies certain optimization criteria (i.e., shortest path). According to this definition, path planning problem is categorized as an optimization problem [14]. The main difficulties for robot path-planning problems are computational complexity, local optimum and adaptability [1].

There are two categories of path planning algorithms: namely the global path planning (off-line) and the local path planning (on-line), based on the availability of information about the environment. Global path planning of robots in environments where complete information about stationary obstacles and trajectory of moving obstacles are known in advance, so that the robot only needs to compute the path once at the beginning and then to follow the planned path up to the target point. When complete information about environment is not available in advance, mobile robot gets information through sensors as it moves through the environment, this is known as on-line or local path planning [5], [15]. In local planning the robot needs the capability to build a map of the environment, which is essentially a repetitive process of moving to a new position, sensing the environment, updating the map and planning subsequent motion. A mobile robot generally has one or more camera or ultrasonic sensors, which help in identifying the obstacles on its trajectory [16].

IV. MODIFIED PARTICLE SWARM OPTIMIZATION (*MPSO*)

Basically the *MPSO*, just like the *PSO*, consists of a population of particles that collectively search in the search space for the global optimum. Improvements are mainly trying to address the problem of premature convergence associated with the standard *PSO*. These improvements usually try to solve this problem by increasing the diversity of solutions in the swarm. In the *MPSO* an error factor is modelled, to ensure that the *PSO* converges. *MPSO* try to address another problem which is the population may include many infeasible paths, which have undesirable effect on the performance of the algorithm. In the *MPSO*, the infeasible paths are not discarded but can be modified to be feasible path. The flowchart of the proposed *MPSO* is illustrated in Fig 1.

The process of *MPSO* is demonstrated in following steps:

Step 1: (Modelling the Environment and Obstacles in the Algorithm) In this study, mobile robot environment which occupied by a number of static obstacles is represented by a grid-based model, consider a two-dimensional (*2D*) square map overlaid with a uniform pattern of grid points. Grid-based model makes the calculation of distance and representation of obstacle easier.

To verify the effectiveness of the *MPSO*, the simulation has been applied to the working environment which is presented in Fig 2. As displayed in the figure, the blue grids represent obstacle free areas where mobile robot can move freely. There is no unit used to measure the path length because each cell in the environment can represent any unit. The circle sign in the environment is the robot's starting and goal location. The grid size and the coordinate of the start and target point are shown in Table 1.

Table 1: Grid size, start and target point, number of obstacles

grid size [unit]	Start point [unit]	target point [unit]	number of obstacles
31 × 23	x=1 , y=1	x=22 , y=30	76

The number of obstacles for each environment is shown in Table 1. As the number of obstacle increases, the algorithm needs more generations to find the best solution. Obstacle areas in the working environment are represented by Blue Square which containing red grid and has an average area of 1 × 1 unit. Boundaries for obstacles area are formed by their actual boundaries plus a safety distance that is defined with consideration to the size of the mobile robot which treated as a point in the environment.

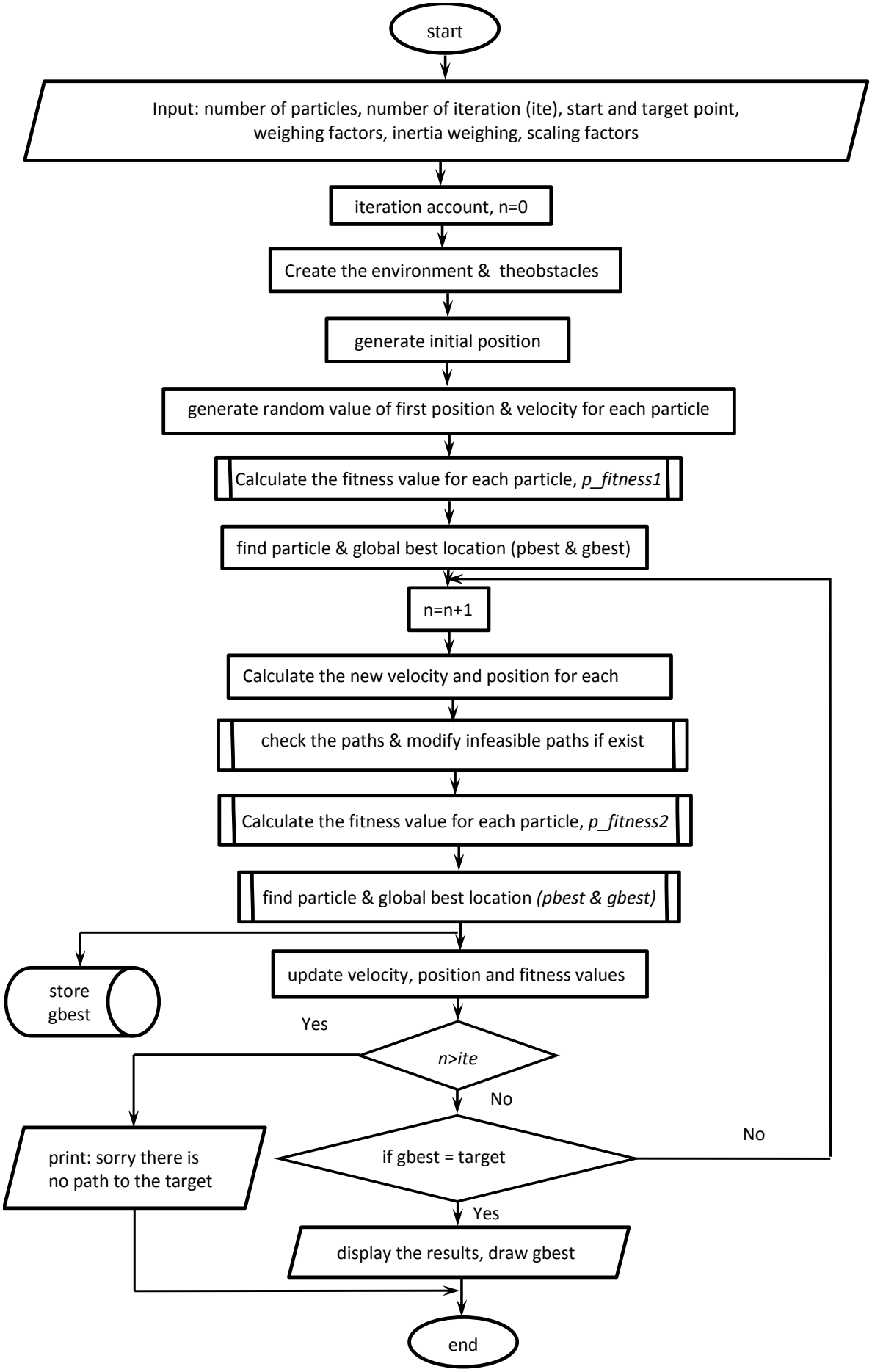

Fig 1: Flowchart of the Modified Particle Swarm Optimization

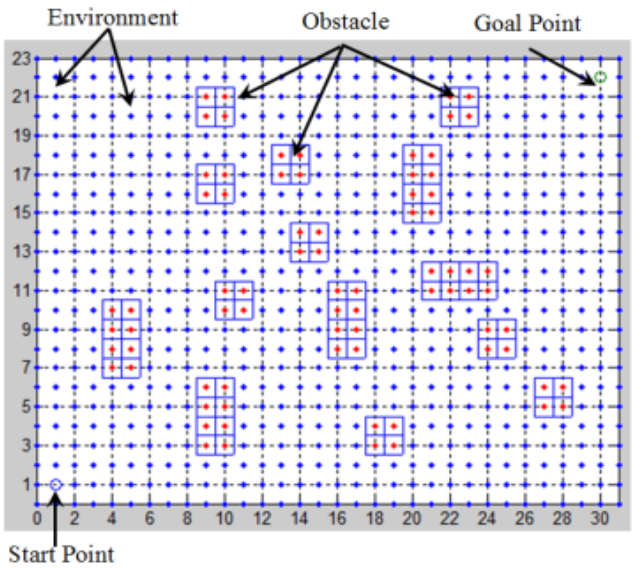

Fig 2: Working environment

Step 2: (Particle swarm initialization) The starting position for each particle is the starting point of the path, then the first position and velocity of each particle is generated randomly but limited to the boundaries of the search space. The particle's position stored in a matrix with size of $2 \times l$, where l is the number of particles. The first and second row of the matrix correspond the x and y-coordinate for all particles, respectively. The coordinates of start and goal points are x_s, y_s and x_g, y_g respectively, and intermediate points in the path are represented by x_n, y_n .

Step 3: (Fitness Function Evaluation) The particle's paths are evaluated during each step. Evaluation function is used to provide a measure of how particles have performed in the problem domain. In the *MPSO*, two parameters (the distance between points and travel time) indicated by the particles are used to calculate the particle fitness (*p_fitness*). Since the fitness should increase as the distance and time decreases, the fitness function for each particle of a feasible path is turned out to be a minimization problem and it is evaluated as:

$$p\_fitness = 1/(w_d * d(p_i, p_{i+1}) + w_t * t(p_i, p_{i+1})) \quad (1)$$

where w_d and w_t are weighing factor for distance and travel time respectively, $d_{(pi,pi+1)}$ denotes Euclidean distance between point P_i to the next point p_{i+1} for the same particle and $t_{(pi,pi+1)}$ denotes time taken by the particle to cover from pi to p_{i+1} :

$$d(p_i, p_{i+1}) = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \quad (2)$$

$$t(p_i, p_{i+1}) = d(p_i, p_{i+1}) / v_i \quad (3)$$

where x_i and y_i are particle's current horizontal and vertical positions, x_{i+1} and y_{i+1} are particle's next horizontal and vertical positions, and vi denotes the velocity of the particle when traveling from pi to p_{i+1}. The calculation of the fitness value is illustrated by the flowchart shown in Fig 3.

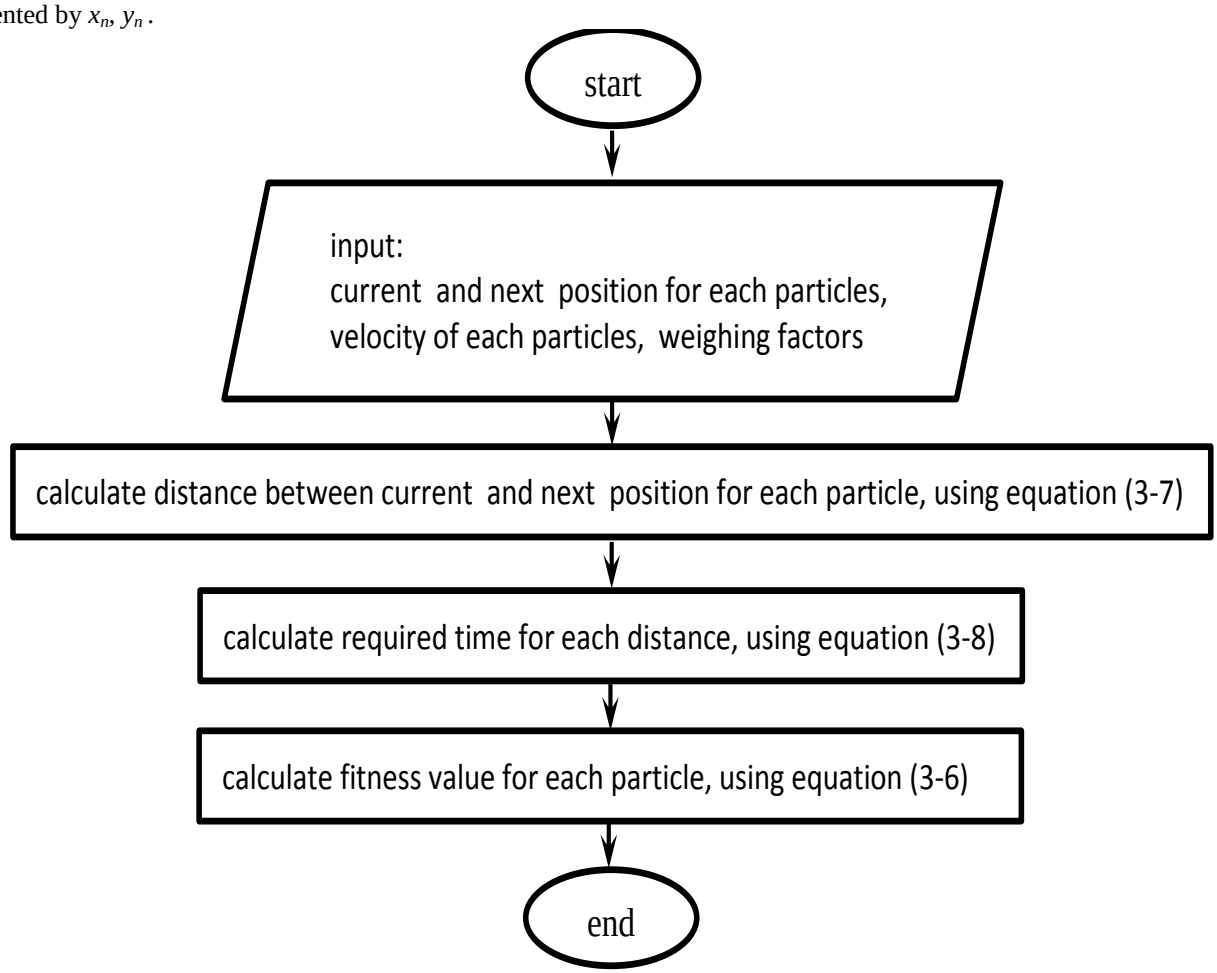

Fig 3: Flowchart for fitness calculation

Step 4: (Particle Position and Velocity Update) The *MPSO* algorithm is initialized with a number of particles and then searches for optimal. The position of a particle is influenced by the best position visited by itself which is referred to as particle best "*pbest*", and the best position in the whole swarm which is referred to as a global best "*gbest*". A particle updates its position and velocity using the following equations [17]:

$$vx_2 = w*vx_1 + c_1*r_1*(pbest - x_1) + c_2*r_2*(gbest - x_1) \quad (4)$$

$$vy_2 = w*vy_1 + c_1*r_1*(pbest - y_1) + c_2*r_2*(gbest - y_1) \quad (5)$$

Where the cognitive component, ($pbest$-x_1), represents the particle's own experience as to where the best solution is and the social component, ($pbest$-y_1) represents the belief of the entire swarm as to where the best solution is. *vx2* and *vy2* are the updated particle's velocity. vx_1,vy_2 and x_1, y_1 are current particle's velocity and position along x and y-direction respectively. Updating velocities is the way that enables the particle to search around its individual best position and global best position.

Self-confidence factor, c_1 and swarm-confidence factor, c_2 makes particles have the function of self-summary and learn to the best of the swarm, and get close to the best position of its own as well as within the swarm. The inertia weight (w) was first introduced by [18]. This term serves as a memory of previous velocities and it employed to control the impact of the previous velocity on the current velocity .The value of inertia factor (w) is in the range of [0, 1]. r_1 and r_2 are random numbers within the interval of [0, 1]. These parameters have considerable effects on performance of *PSO*.

In the basic concept of *PSO* algorithm no error model is used for updating the position. However, in the *MPSO* these errors are modelled to ensure that the *PSO* converges, the modified position update equation for each particle with the error is determined as follow:

$$x_2 = x_1 + vx_2 + r*Errorx \quad (6)$$

$$y_2 = y_1 + vy_2 + r*Errory \quad (7)$$

where x_2 and y_2 are the updated positions along x and y-direction, r is a random number, large values of this parameter leads to global search, while small values leads to fine, local search, which is suitable when algorithm converges. *Errorx* and *Errory* are position error along x and y-coordinate for all particles, respectively, and it's between current particle position and the target position. These errors are used to changing the particle's direction to head toward the target.

$$Errorx = x_p - x_t \quad (8)$$

$$Errory = y_p - y_t \quad (9)$$

The error value is determined for each particle where x_p and y_p are the current coordinates of the individual particle and x_t and y_t are the target coordinates. Based on the above equations, the next possible velocity and position of each particle is determined. If the next possible position resides within the obstacle space, the obstacle avoidance part of the algorithm is employed and the Distinguish Algorithm (*DA*) is used to check the paths

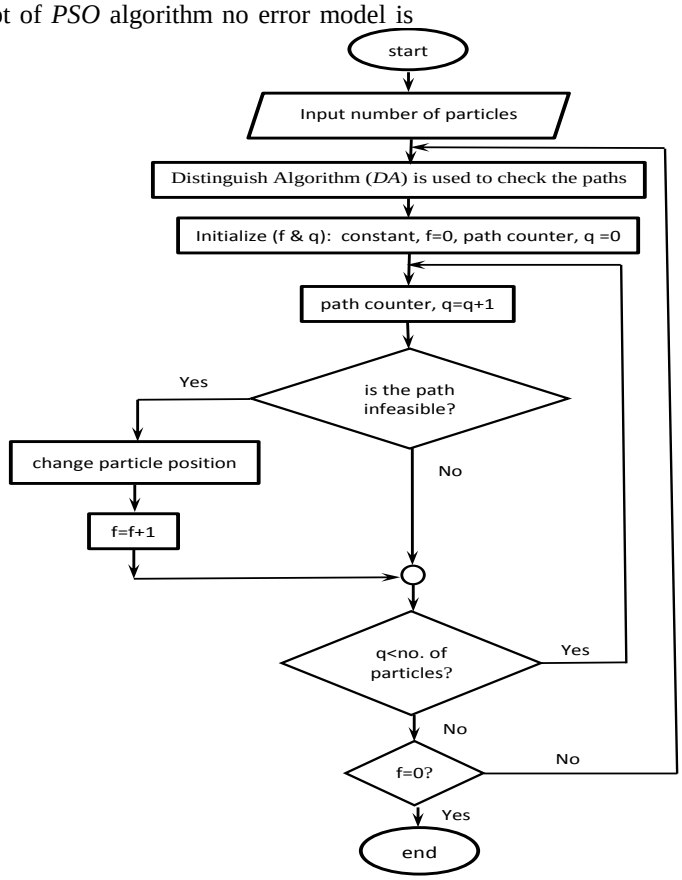

Fig 4: Flowchart for modifying the infeasible paths position

Step 5: (Generation of Feasible Paths) As the particles move through the search space from current position to new position, a conditional statement is used to check all the particles' paths to see if the particles are passing through obstacles. If this condition is true, the path is considered to be infeasible and the obstacle avoidance section of the algorithm is initiated. If particle path does not interfere with the position of the obstacle, the path is considered to be feasible. In the *MPSO*, the infeasible paths are not discarded but can be modified such that the selected path be a collision free by using the proposed algorithm, which is illustrated by the flowchart in Fig 4. As shown in the figure, if the particle path falls with an obstacle boundary, it is relocated to a position outside of the obstacle.

As the next step, the particle's fitness value is compared between current and new location at each iteration to determine the best position *pbest* for each particle, and the particle with the best fitness value is assigned as the global best *gbest*, as illustrated by the flowchart shown in Fig 5.

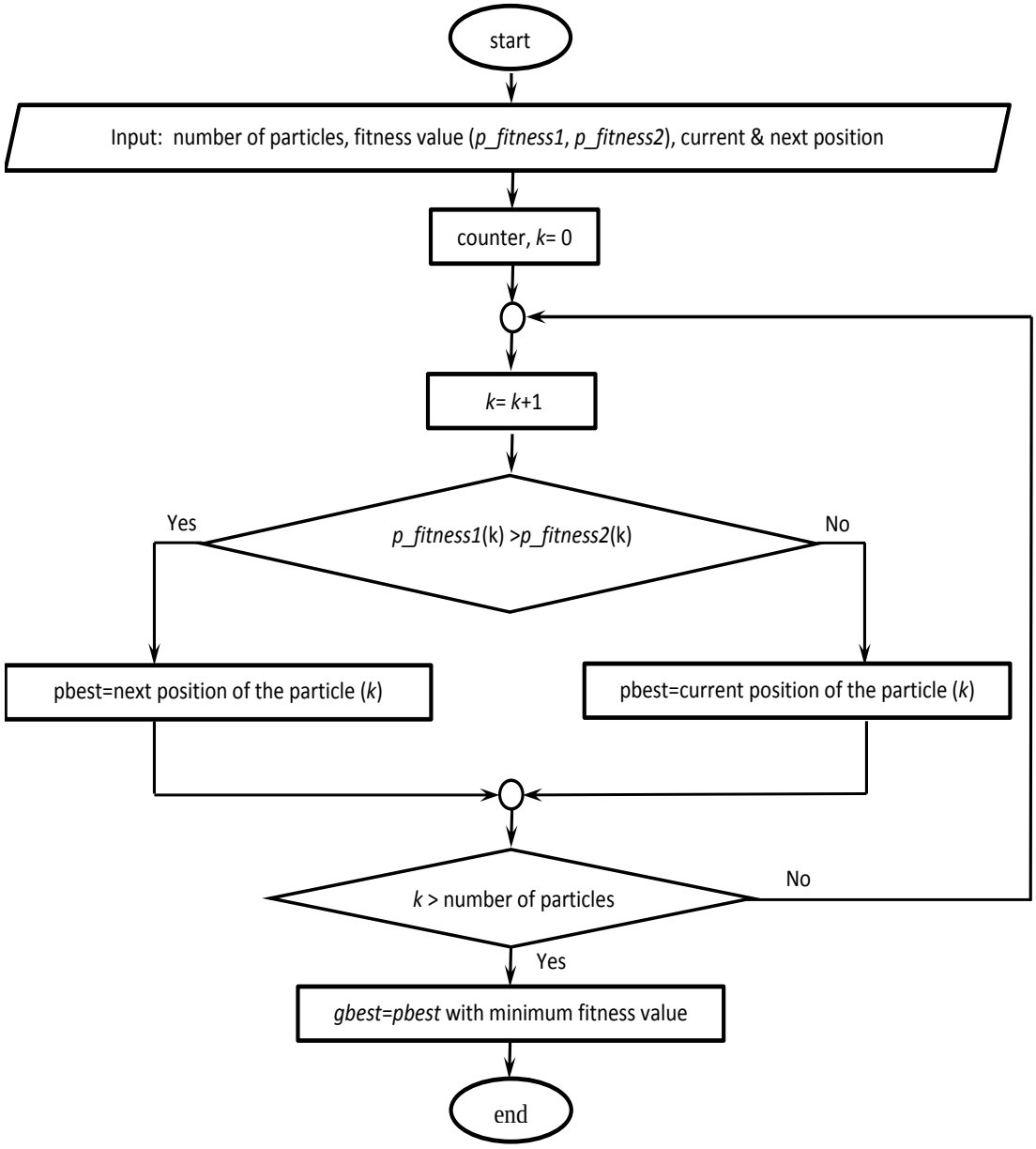

Fig 5: Flowchart for finding the particle and global best location

The *MPSO* algorithm has a main nested loop terminated when the total number of iterations exceeds a certain limit or solution convergence; otherwise the program is terminated when the program could not find the path from start position to the target position.

The path generated by *MPSO* is a group of "*gbest*" points (include start and target point). In all cases, an optimal path is formed by line segment which is connecting the global best position "*gbest*" falling on the grids of the working environment.

IV. THE IMPLEMENTATION OF *MPSO* IN PATH PLANNING

To investigate the effect of different swarm size on the performance of the *MPSO*, different number of particles has been taking into account (i.e., 5, 10, 50, and 100). The particle's size has been selected on the base of trial and error, because there has been no recommendation in the literature regarding swarm size.

In the *MPSO* various combinations of parameters were tested and the best obtained results are: the values of weighing factor in fitness function w_d=100 and w_t=1, inertia weighing factor w=0.02. The swarms with small initial inertia weight converged relatively fast at the beginning of optimization. The chosen values of cognitive scaling and social scaling factors are c_1=c_2=2. Usually $c_1$equals c_2 and ranges from 0 to 4 [12].

To study the performance of the *MPSO*, the proposed algorithm has been applied to working environment presented in Fig 2 to find the optimal path. The best obtained simulation results with various numbers of particles (swarm size) for the working environment are shown in Table 2.

Table 2: The simulation results for the working environment using *MPSO*

Simulation Results			
No. of Particles	Iterations	Distance	Elapsed time [s]
5	33	39.4558	0.95
10	31	39.4558	0.97
50	30	39.4558	1.14
100	31	39.4558	1.41

The best solution obtained after running the *MPSO* algorithm to this environment is showed in Fig 6 to Fig 9.

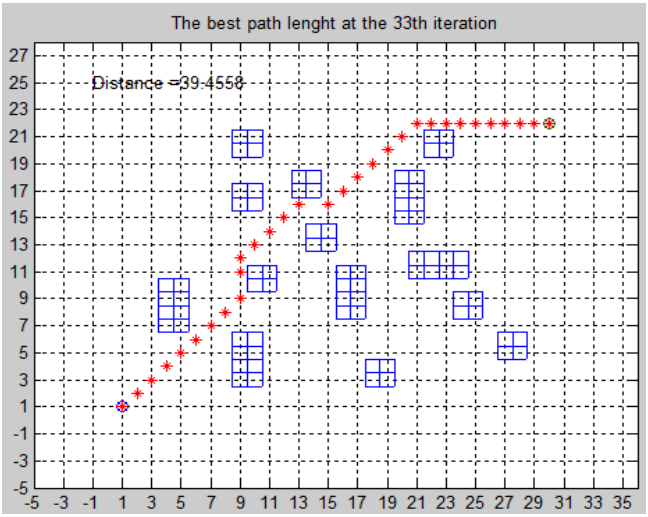

Fig 6: Path generated for the working environment using *MPSO* (5 particles)

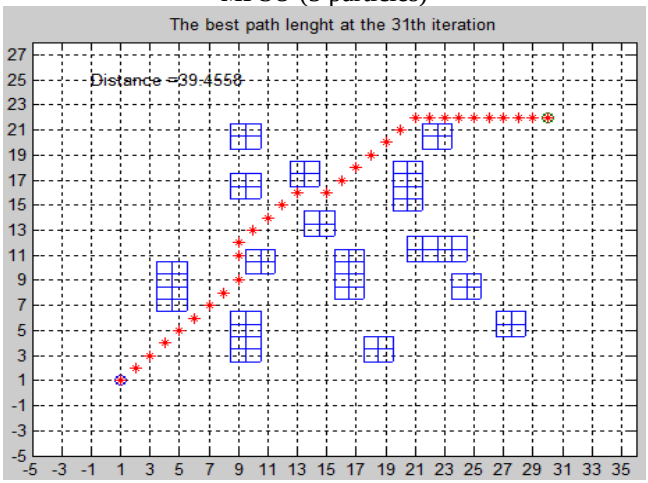

Fig 7: Path generated for the working environment using *MPSO* (10 particles)

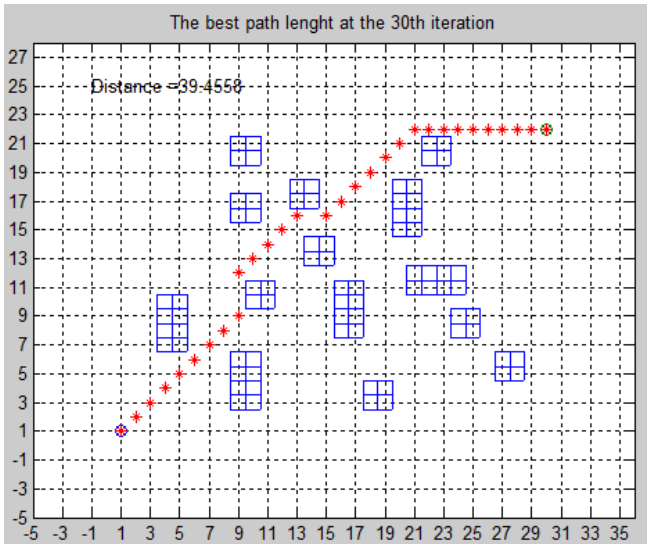

Fig 8: Path generated for the working environment using *MPSO* (50 particles)

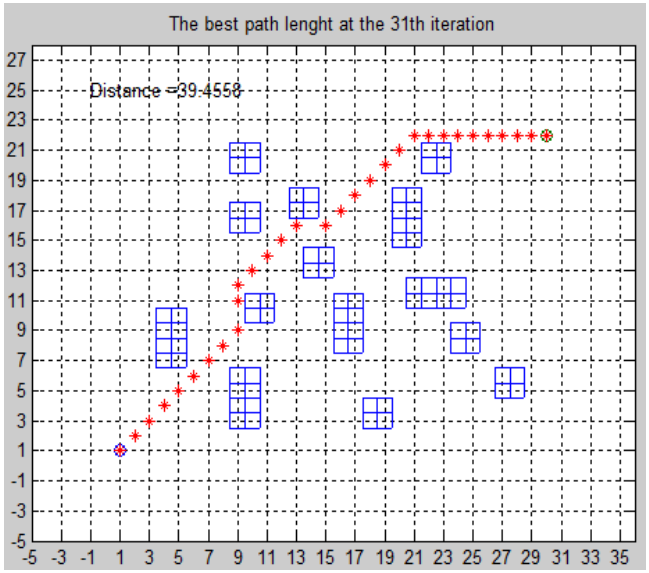

Fig 9: Path generated for the working environment using *MPSO* (100 particles)

It can be observed from the above figures that no improvement is achieved for more than 5 particles in term of the path length, therefore 5 particles are sufficient to find the optimal path with length of 39.4558 in less than 1 second.

The *MPSO* algorithm has been carried out to the working environment which turned to closed environment, in order to demonstrate that the proposed algorithm does not converge when there is no path to the target. **Error! Reference source not found.** shows the elapsed time for the closed environment with various number of particles and different number of iterations. As shown in the table, for closed environment the elapsed time is increased by increasing the number of particles and the number of iterations because of that the large of number of particle and iteration, require larger computations.

Table 3: The simulation results for the working environments using *MPSO*

No. of Particle	10 iterations	50 iterations	100 iterations
5	0.567463	0.639178	0.669309
10	0.624700	0.696363	0.753709
50	0.661831	1.223272	1.656838
100	0.682113	1.636615	2.893451

He results illustrated in Fig 10 showed that if the proposed algorithm could not find a path between the start point and the target point for the working environment an error message is returned. In all cases, the *MPSO* algorithm succeeded to avoid the obstacle wall, even a collision free path did not exist in the closed environments.

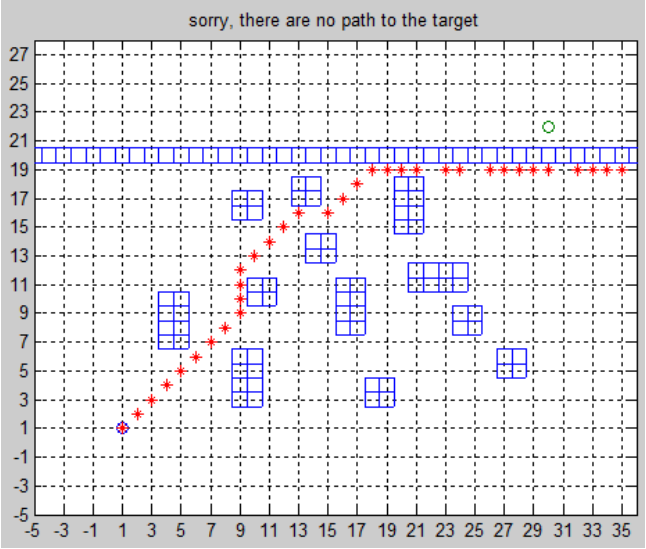

Fig 10: closed environment using *MPSO*

V. CONCLUSIONS

This study investigated path planning for a mobile robot by application of Modified Particle Swarm Optimizer (*MPSO*). A MPSO algorithm is used to find optimal path for the mobile robot in working environment with obstacles. From the simulation results the following conclusions are drawn:

1. The proposed algorithm (*MPSO*) is capable of effectively guiding a robot moving from start position to the goal position in complex environment and find optimum/shortest path without colliding any obstacles in the environment.
2. In the *MPSO* an error factor is modelled, to ensure that the *PSO* converges. These errors are used to changing the particle's direction to head toward the target.
3. A modified procedure carried out in the *MPSO* to solve the infeasible path problem. When the particle path falls with an obstacle boundary, it is relocated to a position outside of the obstacle.
4. The *MPSO* algorithm does not converge in the closed environment in which there is no path between the start and the target point.
5. *MPSO* can rightfully be regarded as a good choice due to its convergence speed and robustness in global search.

Directions for future research are briefly summarized below:

1. Future research can investigate the performance of *MPSO* in dynamic environments.
2. Another future direction is to examine the effectiveness of *MPSO* with physical robots in a real-world application.
3. Future research can investigate the application and examine the performance of *MPSO* to solve obstacle avoidance to real objects limited to three dimensions (*3D*) environment.
4. Testing other swarm optimization approaches like Honeybees Mating Algorithm (*HBMA*) or Ant Colony Optimization (*ACO*).
5. Testing other types of environment like maze-type, and enhancements in terms of obstacles like including the different shapes of obstacles. The different shapes like circle, upward U, inverted U, upward V, and inverted V can be included in the environment.

REFERENCES

[1] Yun, S.C., Ganapathy, V., Chong, L.O. (2010), "Improved genetic algorithms based optimum path planning for mobile robot", International Conference on Control, Automation, Robotics and Vision, ICARCV, pp. 1565-1570.

[2] Han, K. M., (2007), "Collision free path planning algorithms for robot navigation problem", Master Thesis, University of Missouri-Columbia.

[3] Mohanty, P. K., and Parhi D. R. (2013), "Controlling the Motion of an Autonomous Mobile Robot Using Various Techniques: a Review", Journal of Advance Mechanical Engineering, 1: 24-39.

[4] Raja, P., Pugazhenthi, S. (2012), "Optimal path planning of mobile robots: A review", International Journal of Physical Sciences, 7(9): pp. 1314 - 1320

[5] Zhao, Y.,Zu, W. (2009)," Real-Time Obstacle Avoidance Method for Mobile Robots Based on a Modified Particle Swarm Optimization", International Joint Conference on Computational Sciences and Optimization, pp.269-272.

[6] Ahmadzadeh,S., Ghanavati, M. (2012)," Navigation of Mobile Robot Using the PSO Particle Swarm Optimization", Journal of Academic and Applied Studies (JAAS), 2(1): pp. 32-38.

[7] Yu-qin, W., Xiao-peng, Y. (2012), "Research for the Robot Path Planning Control Strategy Based on the Immune Particle Swarm Optimization Algorithm", 2nd International Conference on Intelligent System Design and Engineering Application, pp. 724-727.

[8] Konar, A. (2000), "Artificial Intelligence and Soft Computing Behavioral and Cognitive Modeling of the Human Brain", CRC Press LLC.

[9] Atyabi, A., Phon-Amnuaisuk, S., C.K. Ho, (2010), "Navigating a robotic swarm in an uncharted 2D landscape", Applied Soft Computing, 10(1):149-169.

[10] Dutta, S., (2010), "Obstacle Avoidance of Mobile Robot using PSO-based Neuro Fuzzy Technique", International Journal of Computer Science and Engineering, 2(2): 301-304.

[11] Saska, M., Macas, M., Preucil, L., and Lhotska, L. (2006), "Robot Path Planning using Particle Swarm Optimization of Ferguson Splines", In Proceedings 11th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2006), Prague, Czech Republic, pp. 833-839.

[12] Raja, P., Pugazhenthi, S. (2009), "Path Planning for Mobile Robots in Dynamic Environments using Particle Swarm Optimization", International Conference on Advances in Recent Technologies in Communication and Computing, pp. 401-405.

[13] Sariff, N., and Buniyamin, N., (2006), "An overview of autonomous mobile robot path planning algorithms", 4th Student Conference on Research and Development, Scored 2006, pp. 183-188

[14] Al-Taharwa, I., Sheta, A., Al-Weshah, and M. (2008), "A Mobile Robot Path Planning Using Genetic Algorithm in Static Environment", Journal of Computer Science, 4 (4): 341-344.

[15] Li, P., Huang, X., Wang, M., (2010), "A New Hybrid Method for Mobile Robot Dynamic Local Path Planning in Unknown Environment", Journal of Computers, North America, 5(5): 773-781.

[16] .Velagic, J., Lacevic, B., ans Osmic, N., (2006), "Efficient path planning algorithm for mobile robot navigation with a local minima problem solving", In Proceedings of IEEE International Conference on Industrial Technology, IEEE, Mumbai, pp. 2325–2330.

[17] Jatmiko, W., Sekiyama K., and Fukuda, T. (2006), "Modified Particle Swarm Robotic for Odor Source Localization in Dynamic Environment, the International Journal of Intelligent Control and Systems: Special Issue on Swarm Robotic, 11(3): 176-184.

[18] Eberhart, R., and Shi, Y. (1998), "Comparison between Genetic Algorithms and Particle Swarm Optimization". In Proceedings of the Seventh Annual Conference on Evolutionary Programming, Springer-Verlag, pp. 611-619.