Performance Comparison of PSO Algorithms for Mobile Robot Path Planning in Complex Environments

Dr. Ravi Kumar Jatoth[1], K. Jaya Shankar Reddy[2], K. Karthikeya Yadav[3]

[1]Dept. of Electronics and Communication Engineering, National Institute of Technology, Warangal.

[2, 3] Dept. of Computer Science and Engineering, National Institute of Technology, Andhra Pradesh.

Abstract— **This paper compares the different Particle Swarm Optimization (PSO) Algorithms applied for Mobile Robot Path Planning in Complex Environments. The paper focuses on the paths that are feasible for a Mobile Robot by avoiding static obstacles in a given complex Environment. In this, a constrained environment is chosen where robot is represented as a single point. Particle Swarm Optimization is one of the best evolutionary algorithms applied for Robot Path Planning. There are many improved versions of Particle Swarm Optimization modifying the Classical PSO. In this paper, four different versions of PSO were applied for mobile Robot Path Planning and the results were compared.**

***Index Terms*—Mobile Robot, Path Planning, Classical PSO, Binary PSO, Adaptive PSO, Modified PSO, Complex Environments.**

I. INTRODUCTION

Path planning is a fundamental problem in mobile robotics. Path planning is the process of generating a feasible path for a Mobile Robot in such a way that the robot avoids obstacles. The Robot Path Planning is classified as local and global [1 – 3]. In local Path Planning the robot reaches the goal in steps evolving its next best position each time in an unknown or known environment where as in global path planning the Robot first reaches the goal and tries different paths to avoid obstacles. Global path Planning is also referred as offline path planning and local path planning as real time path planning. Every path which directs the Robot from source to desired target is a feasible path [4]. Generally Path planning involves two main aims: 1) the path should be feasible 2) The path should also avoid obstacles. Achieving the above two aims enables the Robot path planning. In practical cases Robot Path planning is done by detecting the obstacles using image processing both either in known and unknown environments or even in static and dynamic environments. But in optimization techniques we do not use image processing for detecting obstacles. So it becomes a bit difficult to generate a path in an unknown environment using optimization [5]. In this paper we use a known environment in which there are geometrical obstacles.

For Robot Path planning we can use suitable evolutionary algorithms out of which Particle Swarm Optimization is in our Interest. Basic Types of Particle Swarm Optimization algorithms (PSO) are Adaptive PSO, Binary PSO and the Modified PSO. All these are discussed below.

II. PROBLEM FORMULATION

The problem is states as follows. The Robot is considered as a single point and moves in a closed worked space. The workspace is a 2 Dimensional (2 D) environment containing static and geometrical obstacles. The source point and the desired goal point are chosen. The objective is to generate a collision free path taking the robot from the source point and the goal point. The path is divided into segments connecting points from the source to the goal. The area in the workspace occupied by the segments of the path is the configuration space (C – Space). Practically C space is the region obtained by sliding the robot along the edges of the obstacles. The complexity of the path planning increases as the number of dimensions of the C – space increases.

The path is made not to go out of the C – space by applying the limits of position and the velocities. The path will be smooth only if the obstacles do not have sharp corners. But in complex cases there might be sharp cornered obstacles. So we can imagine them blunt by circumscribing or inscribing a circle of fewer radiuses around the obstacles. The path will avoid the circles which imply that the original obstacles are avoided.

Looking for the shorter path does not mean that the time taken is less; we need complex algorithm for a complex environment where the time taken to generate the shortest path might be longer.

III. METHODOLOGY

The Figure below is a small example illustrates the Robot path planning in a C - space

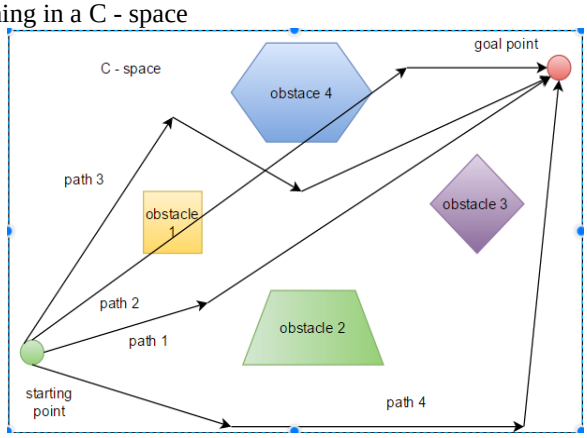

Fig. 1 Example of Path Planning in a C – space having obsatcles, a source and a goal point.

Path 1: avoids obstacles and optimizes the path
Path 2: does not avoids the obstacles
Path 3: avoids obstacles but does not optimize the Path
Path 4: the path that did not meet our conditions.

Obstacles

- Occupied spaces of the world
- Robot should not go into that space

C space

- Unoccupied spaces within the world
- Where the robot can move
- Also referred as the work space or the boundaries

Inputs

- Geometry of the robot
- Geometry of the obstacles
- Geometry of the free space
- A starting and a desired goal position

Outputs

- A continuous path connecting the source and the goal

First some points or locations in the C-space are to be chosen, they are connected with each other to form a path from the source to the goal point and then try to avoid the obstacles. The optimization goals of the path planning are

- The distance travelled by the robot should be least.
- The path should not run into obstacles.
- The path should be smooth.
- The path should not lead the robot outside the configuration space.

IV. OBJECTIVE FUNCTION

The objective function is the basis by which the path optimizes in the robot path planning. There are different objective functions by which robot path planning is done. One of them is by calculating the length of each particle, i.e. the sum of each segment in the particle.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Where the line joining the points (x_1, y_1) and (x_2, y_2) is one of the segment of the path.

Now add all such segments to d
D = sum (d); (sum of all segments)
Now D gives the distance of the entire path from source to goal.

This just optimizes the distance travelled by the robot but we also have to avoid the obstacles, for that we introduced a term foul, i.e. if any of the coordinates of the path are inside the obstacles, and then add this foul to the Objective function. Before adding to it, the foul is increased by a factor, K which is chosen as 100 in this path planning.

Since each obstacle is circumscribed or inscribed by a circle, we can say that the path is foul if any of the coordinates of the path is inside a circle.

If the distance between the point and the Centre of a circle is less than the radius then the point is said to be inside the circle.

Foul = 0 (Initially)
For i = 1: no of points
d= $\sqrt{(x_i - c_1)^2 + (y_i - c_2)^2}$
[(c_1,c_2) - Centre of the circle]
If d < r r – radius of the circle.
Foul = Foul + (r - d)
Now the objective function becomes
D = D * (K* foul)
Example of a foul path with square obstacle

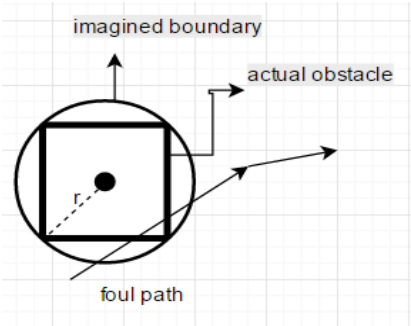

Fig. 2 Example of a path that meets obstacles and with imagined boundary.

So the objective function's function value will be less if the path distance is shorter and there is no foul in the path.

Hence, this is our objective function for robot path planning.

V. OPTIMIZATION ALGORITHMS

Classical PSO

In the Particle Swarm Optimization, the possible solutions of an objective functions are initialized the particles of the

swarm. As an algorithm, the main strength of PSO is its fast convergence. These particles are distributed in the configuration space.

For each iteration update the velocities and positions of each particle.

Velocity update:

$$V_i = w * V_i + C_1 * r_1 * (P_i - X_i) + C_2 * r_2 * (G_i - X_i) \quad (1)$$

Where C_1, C_2 and w are the coefficients of self-component, Social component and Inertial weight respectively.

r_1, r_2– Random numbers in [0, 1]

Position Update:

$$P_i = P_i + V_i \quad (2)$$

This replaces the old particle with the updated particle and calculates the function value of these particles. Now update the new particle best and find out the global best among these and store such at each iteration. At the end of iterations the best function value can be seen in the plot or can be displayed.

Adaptive PSO (PSO – TVIW)

This version is same as classical PSO except that the inertial weight differs at each iteration. As the number of iterations increases, the inertial weight goes on decreasing by using a formula

$w = w_{\max} - (^{w_{max} - w_{min}}/_{maxiter})$*iter

maxiter – the total number of iterations

iter – the present iteration

This can also be done by a damping factor *wd* whose value varies the performance of the PSO by a larger extent which is shown in the example of Robot Path Planning later in this paper.

At the end of each iteration $w = w * wd$ is to be done. Initially $w = 1$.

For Robot path planning the values used were

$$w_{max} = 0.9, w_{min} = 0.4 \ and \ wd = 0.98$$

The Adaptive PSO is also known as PSO – TVIW (Time Varying Initial Weight)

Binary PSO

The Binary PSO was proposed by Eberhart and Kennedy to optimize functions even in Binary PSO [8]

This is the extension of the ADAPTIVE PSO and varies from it in the velocity updating. The ADAPTIVE PSO does not take into account of the particles' velocities reaching the maximum. [10]

This PSO is only the first type of the Binary PSO. In binary PSO, the position of the particles is updated only after the velocities are reflected. This operation is termed as mutation. This uses a mutation factor 'rmu'.

While (number of dimensions)
r = rand ();
If r < rmu (3)

$v_{id} = - v_{id}$

The mutation factor rmu is chosen as 0.4.

In case of Robot Path Planning the Binary PSO gives better results in much complex environment than other versions of PSO.

The pseudo code is as follows:

1. Start
2. G = 0 (generation index)
3. Initialize the swarm with some random positions
4. Evaluate their function values using the objective function
5. G = 0 (generation index)
6. Update the velocity using (1)
7. Mutate the velocity using (3)
8. Update the position using (2)
9. Evaluate the function value using objective function
10. Replace with the old particle
11. Evaluate the global best for each iteration
12. If satisfied
13. Stop;
14. Else
15. Go to step 6

Modified PSO

The classical PSO takes into account only the particles' best position and the global best among all the particles. But the modified PSO takes even the particles' worst position and the global worst among all the particles. And also includes them in the velocity updating formula of the particles.

$$V_i = w * V_i + C_1 * r_1 * (P_i - X_i) + C_2 * r_2 * (G_i - X_i) + C_3 * r_3 * (W_i - P_i) \quad (4)$$

W_i – Worst Function value of the particle

C_3 - Another acceleration coefficient

r_3 – Random number in [0, 1].

And at the end of each iteration, both the global best and the worst best have to be found out. Rest all the steps remain same as the classical PSO.[11]

All the above versions seem to be changed just a little bit in terms of algorithm from the classical version of PSO, but

the path planning has got varied by a larger extent in different environments.

VI. Simulation and Results

Robot Path Planning in Environment 1

Classical PSO based Robot Path Planning :

The Classical PSO algorithm generated a path that is free from obstacles but has taken many turns thereby increasing the length of the path. The path is also not so close to the obstacles as shown in Fig. 1.

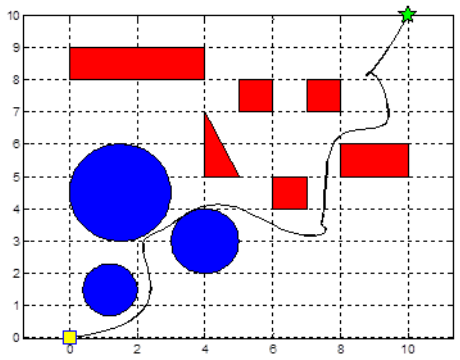

Fig.3 Path obtained using classical PSO Algorithm

Adaptive PSO based Robot Path Planning

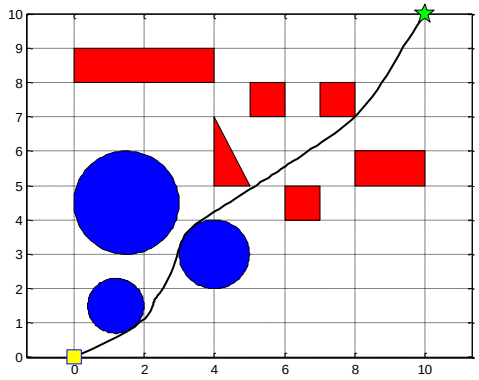

Fig. 4 Path obtained using Adaptive PSO Algorithm

The Adaptive PSO generated a perfect path free from obstacles and also the shortest path but the path is too close to the obstacles as shown in Fig. 2.

Bianry PSO based Robot Path Planning

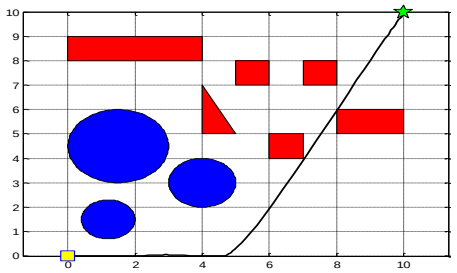

Fig. 5 Path obtained using Binary PSO Algorithm

This algorithm generated a path that is free from the obstacles but it did not avoid the obstacles in the shortest path. The path is also very close to the obstacles as shown in Fig. 3.

Modified PSO based Path Planning

This case is same as the classical PSO. The path has taken too many turns which practically not possible for a mobile Robot. The path has just tried to avoid the obstacles but did not optimize the path travelled as shown in Fig. 4.

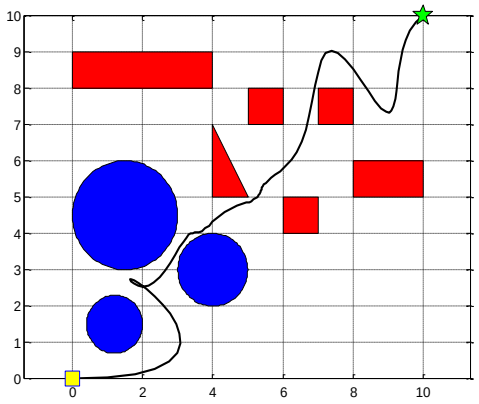

Fig. 6 Path obtained using Modified PSO Algorithm

TABLE I

COMPARISON OF THE ABOVE ALGORITHMS IN ENVIRONMENT 1

TYPE OF PSO	PATH LENGTH
PSO classical	19.3366
PSO – TVIW	14.5991
BPSO	16.0705
Modified PSO	21.2

Robot Path Planning in Environment 2

This environment is a bit much complex than the environment 1 as this path has to follow a zigzag pattern to reach the goal avoiding obstacles.

Adaptive PSO based Robot Path Planning

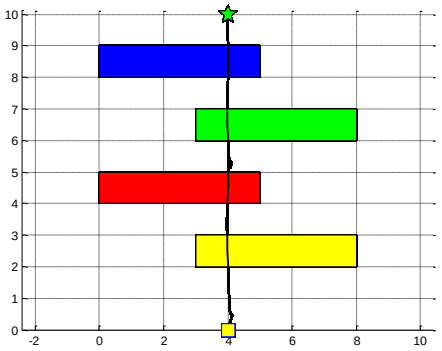

Fig. 7 Path obtained using the Adaptive PSO Algorithm

The Classical PSO and the Modified PSO had resulted in generating the same path as that of the Classical PSO as shown in Fig. 4.

Binary PSO based Path Planning

Binary PSO was avoiding obstacles and reached the Goal position. It has followed a zigzag pattern as shown in Fig. 4

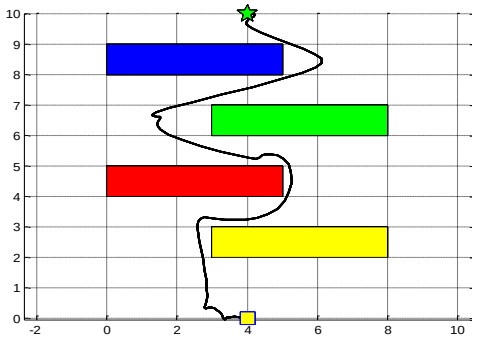

Fig. 8 Path obtained using the Binary PSO Algorithm

VII. CONCLUSION

Four different versions of PSO with the same objective function to optimize the path were applied for a single point Mobile Robot Path Planning. These include the Classical PSO, Binary PSO, Adaptive PSO and the Modified PSO. All of these tried to avoid the obstacles but only the Adaptive PSO optimized the path travelled to a large extent in the environment 1. In another much complex environment 2, the Binary PSO gave the best path that is feasible for a Robot. This concluded that the type of algorithm to be used depends on the geometry of the obstacles and also the environment.

TABLE IIIII

OVERALL COMPARISON OF DIFFERENT PSO ALGORITHMS IN ENVIRONMENT 1 AND ENVIRONNMENT 2

PSO	Environment 1	Environment 2
Classical	Yes (but not optimize the path length	No
adaptive	Yes	No
binary	Yes (too close to the obstacles)	Yes
modified	Yes (too many turns in the path)	No

VIII. REFERENCES

[1] R. Regele and P. Levi, “Cooperative Multi-Robot Path Planning by Heuristic Priority Adjustment”, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006.

[2]K.H. Sedighi, K. Ashenayi, T.W. Manikas, R.L. Wainwrightand H. Tai, Autonomous Local Path Planning for a MobileRobot Using a Genetic Algorithm, in: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1338–1345,2004

[3] M. Saska, M. Macas, L. Preucil and L. Lhotska, Robot Path Planning using Particle Swarm Optimization of Ferguson Splines, in: Proceedings of 11th IEEE International Conference on Emerging Technologies and Factory Automation, , pp. 833–839, 2006.

[4] S.M. LaValle and S.A. Hutchinson, “Optimal Motion Planning for Multiple Robots Having Independent Goals”, IEEE Transaction on Robotics and Automation, pp. - 912–925, 1998.

[5] C. Hocaoglu, A.C. Sanderson, "Planning Multiple Paths with Evolutionary Speciation", IEEE Transaction on Evolutionary Computation, Vol 5, No.3, JUNE 2001.

[6] Behnke, S. Local Multiresolution Path Planningin 7th RoboCup Int.Symposium, Padua, Italy Springer.2004.

[7] Eberhart, Y., Shi. “Particle swarm optimization: developments, applications and resources. In Evolutionary Computation.” Proceedings of the 2001 Congress on Evolutionary Computation. 2001.

[8] Glavaski, D., Volf M., Bonkovic M., Mobile robot path planning using exact cell decomposition and potential field methods. WSEAS Transactions on Circuits and Systems, 2009.

[9]Eberhart RC. A discrete binary version of the particle swarm algorithm. In: Proceedings of conference systems man cybernetics,NJ: Piscataway; p. 4104–4108,1997

[10] Lee S, Park H, Jeon M. Binary particle swarm optimization with bit change mutation. IEICE Transaction Fundamental, Electronic and Communication, Computer Science ;E-90A(10):2253–2256,2007.

[11] Yang B., Zhang Q., Frame Sizing and Topological Optimization Using a Modified Particle Swarm Algorithm, Second WRI Global Congress , Intelligent Systems (GCIS), 2010.