Evolving Robotic Path with Genetically Optimized Fuzzy Planner

Rahul Kala*, Ritu Tiwari, Anupam Shukla,
Soft Computing and Expert System Laboratory,
Indian Institute of Information Technology and Management Gwalior,
Gwalior, Madhya Pradesh-474010, India
Email: rahulkalaiiitm@yahoo.co.in, rt_twr@yahoo.co.in, dranupamshukla@gmail.com

* Corresponding Author

Citation: R. Kala, A. Shukla, R. Tiwari (2010) Evolving Robotic Path with Genetically Optimized Fuzzy Planner. *International Journal of Computational Vision and Robotics*, 1(4), 415-429.

Final Version Available At: http://www.inderscience.com/info/inarticle.php?artid=38196

Abstract

Path planning is one of the highly studied problems in the field of robotics. The problem has been solved using numerous statistical, soft computing and other approaches. In this paper we evolve the robotic path using Genetic Algorithms (GA). The GA generates solutions for the static map which disobeys the non-holonomic constraints. Fuzzy Inference System (FIS) works on the generated path and extend the problem for dynamic environment. The results of GA serve as a guide for FIS planner. The FIS system was initially generated using rules from common sense. Once this model was ready, the fuzzy parameters were optimized by another GA. The GA tried to optimize the distance from the closest obstacle, total path length and the sharpest turn at any time in the journey of the robot. The resulting FIS was easily able to execute the plan of the robot in a dynamic environment. We tested the algorithm on various complex and simple paths. All paths generated were optimal in terms of path length and smoothness. Hence using a combination of GA along with FIS we were able to solve the problem of robotic path planning.

Keywords: Path Planning, Robotics, Fuzzy Inference System, Genetic Algorithms, Genetic Optimization, Hierarchical Algorithms, Computer Vision, Robotics, Hybrid Computing, Soft Computing, Artificial Intelligence.

1. Introduction

Robotics is a highly multi-disciplinary field that incorporates inputs from wireless systems, networks, cognition, image processing, AI, electrical, electronics and other related fields [1]. The whole problem of robotics include taking input from sensors, making of robotic world map [2], path planning, robot control [3], multi-robot coordination, high-end planning, etc [4]. Path Planning [5] is a specific problem in case of robots where we are given a map of the world. The problem is to compute a path for the robot that can guide it to reach a specific goal starting from a specific position. A path planning algorithm must ensure that if a solution is possible, it is found and returned. This is called as completeness of the algorithm [6]. It must also ensure that the algorithm gives its result within the specified amount of time [7]. Some of the commonly used maps include topographical maps [8], Voronoi maps [9, 10], hybrid maps [9], etc. Path planning usually gives its output to the robotic control. Various robot controllers have been designed. Some of them are built using the Adaptive Neuro-Fuzzy architecture [11].

It was earlier shown by the Kala et al. [7] that Genetic Algorithms (GA) generates good results for the problem of path planning. However, GA also generates very sharp paths which the real robot would find it very difficult to follow. These are called the non-holonomic constraints in the robot. The GA used in this paper algorithm gives the basic structure of the solution. This is a path of reasonably small length as well as low complexity or number of turns. This solution of GA acts as a guide for the Fuzzy Inference System (FIS) based planner. The FIS planner generates solution early especially in complex maps. The solutions follow the non-holonomic constraints. The initial fuzzy model used by the system was developed by experimentation and best of the understanding of the authors. Once this basic prototype was ready, GA was used to further optimize the model.

The problem of path planning has been a very active area of research especially during the last decade. The problem has seen numerous methods and means that cater to the needs of the problems. A class of algorithms uses the potential method approach to navigate a robot [4]. In this approach, whenever the robot collides with a robot, a large potential is given. The potential increases if robot moves too close to the obstacle. The aim is the minimization of the potential. Pozna [12] solved the problem using a potential field approach for obstacle avoidance. Other potential field methods include [13]. Hui [14] gave a comparison between the Potential Field and the Soft Computing solutions. Various statistical approaches have also been used. This includes the work of Jolly [15] who used Bezier Curve for path planning. Goel [16] solved the problem for dynamic obstacles using an adaptive strategy. Quad Tree [17], Mesh [18], Pyramid [19] are representations that have been tried for better performance. Another good amount of work exists using the Soft Computing approaches especially Genetic Algorithms [20, 21, 22, 23], A* Algorithms [24] and

neural networks [7]. Shibata [25] used Fuzzy Logic for fitness evaluation of the paths generated. Various other approaches [26, 27] have also been proposed. Zhu [28] used the concepts of cell decomposition and hierarchal planning. Here they represented the cells using a concept of grayness denoting the obstacles. Uridylis [19] used a multi-level probability based pyramid for solving the path planning problem [19]. Hierarchical Planning can also be found in the work of Lai [29] and Shibata [25].

Along with the problem of path planning, the researchers have also studied the degrees of freedom and dimensionality as they have a deep impact on the problem. Jan [30] presented his work for solving in 3 degrees of freedom. Chen [31] made an adaptive intelligent system and implemented using a Neuro-Fuzzy Controller and Performance Evaluator. Their system explored new actions using GA and generated new rules. In the field of multi-robot systems, Carpin [32] used an approximation algorithm to solve the problem of robotic coordination using the space-time data structures. They showed a compromise between speed and quality. Pradhan [33] solved a similar problem for unknown environments using Fuzzy Logic. Peasgood [34] solved the multi-robot planning problem ensuring completeness using Spanning Trees. Hazon [35] analyzed the completeness of the multi-robot coverage problem. Hara [36] gave the idea of using embedded networks as sensors for solving the problem.

2. Algorithm Outline

In this section we give a general outline to the algorithm that we have developed for solving the problem of robotic path planning. The general structure of the algorithm is given in figure 1. The algorithm starts by taking as input the graph. The GA runs on this map to generate a path P. This path is generated using an evolutionary strategy that optimizes both the path length and the complexity. The path P is a collection of points p_i such that p_1 is source node and p_N is the goal node. These points are then used one after the other (other than the source) to act as guide for the Fuzzy Planner. As soon as the robot gets close enough by some predefined threshold amount to the region in which the goal cell is found, the next point in P becomes the goal and the robot has to move a step further. This goes on and on till the robot reaches the final block where the goal node is found. The FIS is used in every block for guiding the robot. After the last iteration, FIS planner is used to make the robot reach the exact cell where the goal is located. The training stage needs to be applied only once to generate the FIS. Afterwards the FIS may be used reputedly for all problems. First the initial FIS is generated and then the FIS parameters are optimized using GA. For this purpose benchmark maps were used.

3. Fuzzy Planner

The movement of the robot in the system is done by a FIS Planner. The initial FIS was generated by hit and trial method. The FIS is supposed to guide the robot to reach the goal position. The FIS tries to find out the optimal next move of the robot.

3.1. *Inputs and Outputs*

The FIS takes 4 inputs. There are angle to goal (α), distance from goal (d_g), distance from obstacle (d_o) and turn to avoid obstacle (t_o). The angle to goal is the angle (α, measured along with sign) that the robot must turn in order to face the goal. This is measured by taking the difference in current angle of the obstacle (φ) and the angle of the robot (θ). The result is always between -180 degrees and 180 degrees. This is shown in figure 2.

The distance from goal (d_g) is the distance between the robot and the goal position. This distance is normalized to lie between 0 and 1 by multiplying by a constant. Similarly the distance from obstacle (d_o) is the distance between the robot and the nearest obstacle found in the direction in which the robot is currently moving. This is also normalized to lie between 0 and 1.

The turn to avoid obstacle angle (t_o) is a discrete input that is either 'left', 'no' or 'right'. These stand for counter-clockwise turn, no turn and clockwise turn respectively. This parameter represents the turn that the robot must make in order to avoid the closest obstacle. This input is measured by measuring the distance between the robot and the obstacle at three different angles. The first distance is the distance between the obstacle and the robot measured at the angle at which the robot is facing (a). The second distance is the same distance measured at an angle of d_θ more (c). The third distance is measured at an angle d_θ less (b). These three distances are given in figure 3.

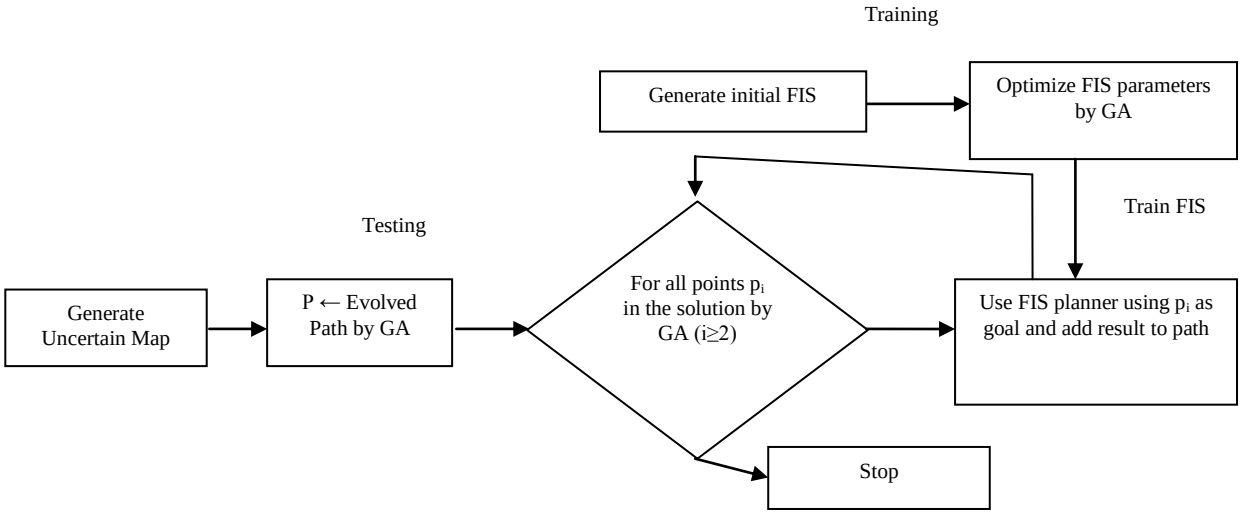


Fig 1. The Algorithm

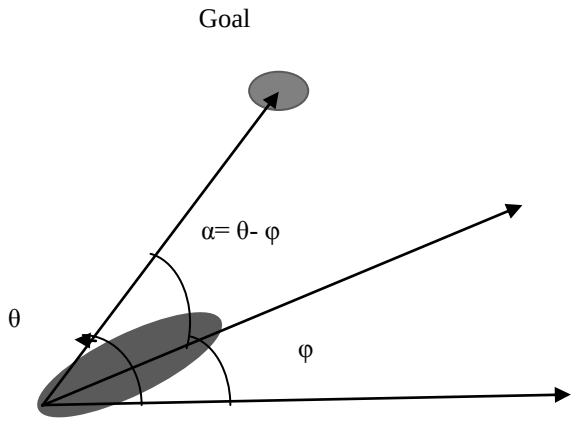


Fig 2. The angle to goal

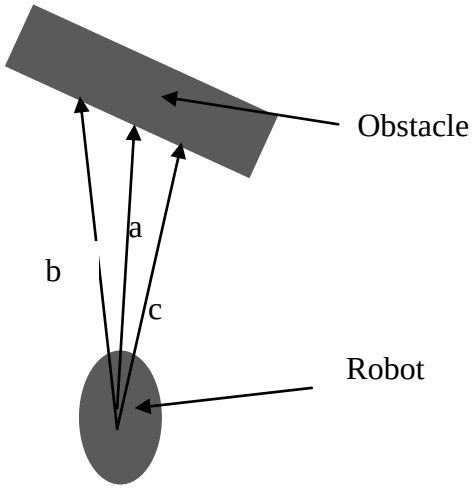


Fig 3.: Determination of Turn to avoid obstacle

Various cases are now possible. The first case is c>b. This means that the obstacle was turned in such a way that turning in the clockwise direction made it even furtherer. In this case the preferred turn is clockwise with an output of 'right'. The second case is b>c. This means that the obstacle was turned in such a way that turning in the anti-clockwise direction made it even furtherer. In this case the preferred turn is anti-clockwise with an output of 'left'. The third case is b=c. This is the case when the robot is vertically ahead of the robot. In such a case we follow a 'left preferred' rule and take a 'left' turn.

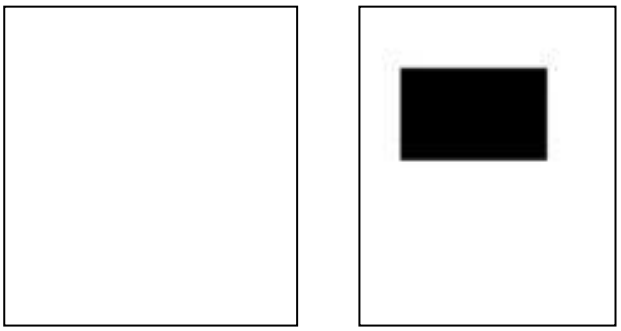
There is a single output that measures the angle (β) that the robot should turn, along with direction.

3.2. *Rules*

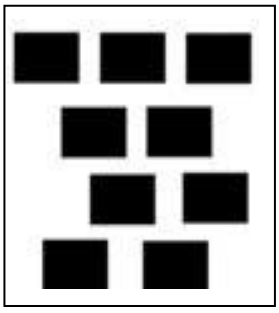
Rules are the driving force for the FIS. Based on these inputs, we frame the rules for the FIS to follow. The rules relate the inputs to the outputs. Each rule has a weight attached to it. Further some inputs have been applied with the not operator as well. All this makes it possible to frame the rules based on the system understanding. The rules can be classified into two major categories. The first category of rules tries to drive the robot towards the goal. The second category of rules tries to save the robot from obstacles. If an obstacle is very near, the second category of rules become very dominant. 11 rules are identified.

4. Genetic Optimization of FIS Planner

The Fuzzy model discussed in section 3 was an initial fuzzy model that was generated. The parameters of this model were decided by trial and error. In order to make this a scalable system, there were further optimizations that were necessary. This was done by the means of GA. The GA was supposed to optimize the fuzzy parameters for most optimal planning of the robotic path. There were a total of 32 fuzzy parameters that were optimized by the GA. These parameters were the parameter values of the various FIS membership functions. For this problem, we first created 3 maps that acted as benchmark maps for the GA. The GA was supposed to optimize the performance of the system against these maps. The first map was a simple map with no obstacles. The second map had single obstacles on the way from source to goal. The robot was supposed to avoid this path. The third map had many obstacles and the robot was supposed to find its way out of them. These maps are given in figure 4.



(a) No Obstacle **(b)Single Obstacle**



(c) Many Obstacles

Fig 4.: The Benchmark maps for Genetic Algorithms

The fitness function for any of the map i tried to optimize three things (i) The total path length (L_i) (ii) The maximum turn taken any time in the path (T_i) (iii)

Distance from the closest obstacle anytime in the run (O_i). All these were normalized to lie on a scale of 0 to 1. The total fitness for any map i may hence be given by (1)

$$F_i = L_i * (1-O_i) * Ti \quad (1)$$

The fitness F_i for any graph hence lies between 0 and 1. 1 is the maximum fitness a map can attain. If the robot at any time moves out of the map, or collides with an obstacle, we assign it a fitness value of 1. The GA would naturally result in deletion of such solutions in the course of generations. In this manner we handle infeasible solutions in GA. The total fitness (F) of any individual in GA is the sum of its score or fitness in all these three maps. This is given by (2). The total fitness can be anywhere between 0 and 3.

$$F = F_1 + F_2 + F_3 \quad (2)$$

5. Evolving Robotic Path by Genetic Algorithms

In this algorithm we try to evolve the robotic path with the help of GA. It is natural that the better paths that GA can return would be as straight as possible without too many turns. We have already stated that the path returned by the GA is in form of a collection of points P. The total number of points in the path is known as path complexity. It is assumed that the robot can reach the goal from the source by travelling in a straight line between every adjacent pair of points (P_i, P_i+1) in this collection P. This is checked at every fitness call of the GA. If a straight line path is obstructed by some obstacle, the path is regarded as infeasible. Since we are interested in increasing the straightness, we claim that good solutions have as fewer points in this collection P as possible. In other words the complexity needs to be the least. It may be easily visualized that for an empty graph only the source and the destination would be present in P. Similarly if there is a unit obstacle in a way from source to destination, P may contain one point additional along with the source and the destination. The source and the destination would always be there in the solution, hence we do not consider them in the collection set rather add them before returning the solution.

In order to control the size of the returned solution, we restrict the maximum number of points that the GA can sustain in any individual to α. We start with very less value of α (α =0). As the generations increase, we keep increasing the value of α. Here α increases as per the Gaussian curve. First the increase in value of α with respect to generations is very rapid. As the generations keep increasing, the rise in the value of α is not very vast. After few more generations α seems to be more or less constant. The relation between α and N, the number of generations is given by (3). Here g is the number of generation and σ and A are constants.

$$\alpha = A \exp(-g^2/2\sigma^2) \quad (3)$$

The solution pool at any time would be consisting of individuals of different length and fitness. Hence it is necessary to adapt a strategy for the Genetic Operators to be used between these chromosomes of unequal length. Also many new Genetic Operators may be required for the purpose of carrying out specialized operations specific to the problem. These are dealt one by one.

5.1. *Individual Representation*

Here each individual is a potential solution to the problem of path planning that we are dealing. The individuals are a collection of points that the robot must access in sequential order in order to reach the goal. The source and the goal are not present in the collection of points. Each point has a x coordinate and a y coordinate. We place a constraint on the individuals. Suppose a line is drawn from the source to the destination. Imagine a line sweeping perpendicular to this line from the source to the destination. All points represented by the solution need to be sorted in the order of sweep of this line. This means that every movement of the robot would take it closer to the destination. If any individual generated during any operation disobeys this rule, it needs to be repaired via a repair operation where this sorting takes place. The chromosomes in this case would be of varying sizes.

5.2. *Crossover*

Here we derive a strategy that carries out crossover operation between two chromosomes that are of unequal length. The first task that we do is to make the length of the chromosomes equal. This may be done by duplicating random points in the smaller chromosome, until the lengths of both chromosomes are same. Duplication creates the new point at the same index to keep the sorted order same. Now one point crossover is carried out in which half the points are taken from first parent and half from the other. Duplicate points from the generated child are deleted. This many times would result in killing of smaller solutions which do not generate results.

5.3. *Mutation*

Mutation selects points in the individual at random and moves them over the graph by some arbitrary percentage. The percentage depends upon the mutation rate used in the algorithmic implementation.

5.4. *Add Points*

Add Points when applied over an individual causes it to requests for an addition of a point in its collection. If it is less than α, then the request may be granted with a probability p. The probability p is inversely proportional to the chromosome length.

5.5. *Add Individual*

Add Individual operator inserts new large sized solutions in the population pool. As the generations grow the need of larger population size would be necessary. This operator inserts individual at a rate R that increases similar to that of α.

5.6. *Flush Population*

Flush Population operator that is active mainly at start of the algorithm when the total number of points (α) is much less that the least number of points to solve the problem. This operator deletes all small infeasible solutions and replaces them with relatively big feasible solutions.

5.7. *Elite*

Elite passes the best individuals of lower generation to the higher generation. This operator ensures that the best individuals do not get deformed with generations due to genetic operations.

5.8. *Fitness Function*

The fitness function is simply the length of the total path that may be computed by the point wise traversal. If traveling at some path results in a collision, then the maximum fitness function is assigned to the individual and the individual is regarded as infeasible. An extra penalty of L * β is added where L is the path complexity or the number of points in the chromosome representation and β is a constant.

6. Simulations and Results

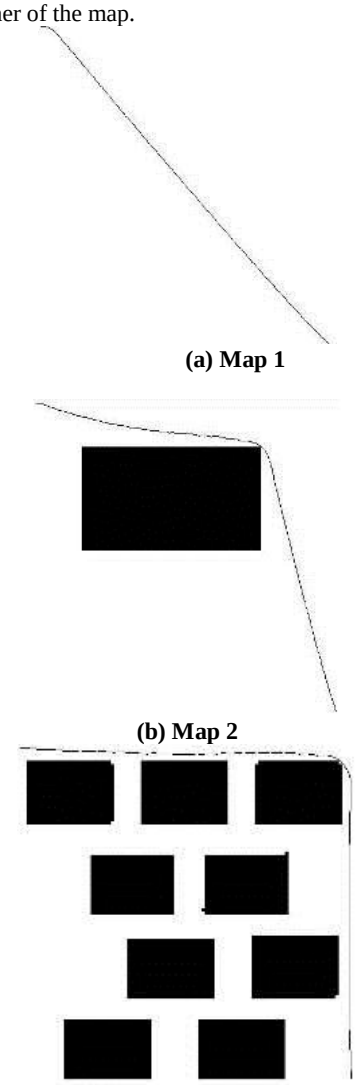
In order to test the working of the algorithm, we made a simulation engine of ourselves. All simulations were done on a 2.0 dual core system with 1 GB RAM. The initial map was taken as an input in form of an image. The image depicted the obstacles as black regions and the path as the white region. First the FIS Planner was made and implemented. Then the FIS planner was optimized by using GA. Then this optimized model was implemented and tried on the benchmark problems. The system was then tried on numerous maps with varying obstacles. These are given in the next sub-sections.

6.1. *Genetic Optimizations of FIS Parameters*

The purpose of the GA was to find out the exact parameter values for the FIS. 32 such parameters were identified that needed optimizations. The Matlab GA toolkit was applied for the optimization purposes. The population was represented by the double vector mechanism. The population size was fixed to 50. Rank based fitness scaling was used. Stochastic Uniform selection was used. The crossover rate was 0.8 with elite count as 2. Gaussian Mutation function was used with a scale and spread of 1 each. The GA was made to run over 100 generations. It took about 1 hr 15 mins for the GA to complete the generations.

The GA search space consisted of the region around the value chosen by the hit and trial method of the 32 parameters on both sides by 10%. This means if the value of any parameter was x, the GA was supposed to optimize it in the search region of x-0.1*L to x+0.1*L where L is the total range of values that x can take. It is natural that on the basis of rules framed, the solution was not likely to be present in any other region. This provided ample of space for GA to locate the global minima and at the same time making the space finite to search for possible solutions.

The first tests were applied at the benchmark problems. This was an attempt to see the behavior of the algorithm at the benchmark problems. Since these benchmark problems were applied on low resolution graph of size 100X100, there was no need to run the GA algorithm. Direct FIS planner was used. The maps are shown in figure 4. The path traced by the algorithm after optimization is given in figure 5. In all cases the source is the top left corner of the map and the goal is the bottom right corner of the map.



(a) Map 1

(b) Map 2

(c) Map 3

Fig 5.: The path traced by robot for benchmark maps

The solutions generated by the algorithm were all smooth so that the robot can be easily turned as per requirements. They were all of optimal length. In each case the robot was easily able to steer its way by avoiding obstacles and reach the goal node. In all the cases the robot was initially facing at the direction of the X-axis and not towards the goal. This is the reason why we see a smooth transition in path and angle at the starting points in the graph.

6.2 Results

The actual testing of the algorithm was done by making it solve new maps. In all the maps the algorithm was supposed to find out the final path of the robot. This happened in two stages. In the first stage the algorithm used GA to compute the set of landmark points in the entire map. In the second stage the trained FIS from section 6.1 was used to smooth the path obeying the non-holonomic constraints.

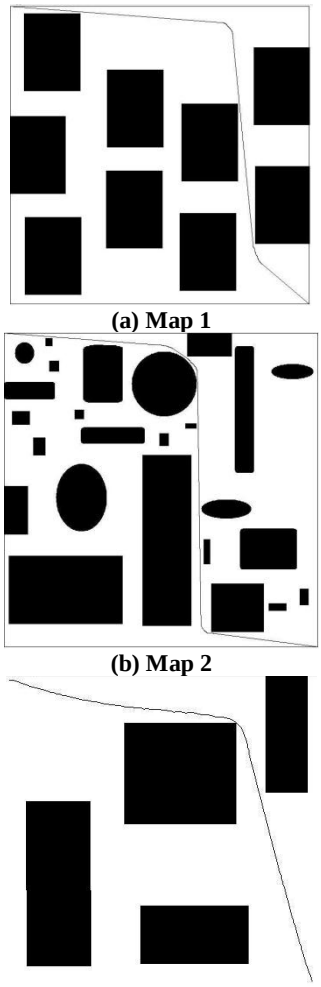
The population was represented by the discussed technique. In all simulations, the population size was fixed to 700. Rank based fitness scaling was used. Stochastic Uniform selection was used. The crossover rate was 0.7 with elite count as 2. Mutation rate was fixed to be 0.06. The GA was made to run over 500-1000 generations depending upon the map used. The maximum value of the complexity (A) was fixed to be 6 and associated σ was fixed to 0.3 times the total generations. β was fixed to 0.15 times the total map perimeter.

We tested the algorithm against 3 maps. This time the maps were of larger size of 1000X1000. Many obstacles were places in between the source and the goal. The robot was supposed to reach the goal from the source. The top left corner is the source and bottom left corner is the goal. In all the cases, the source was fixed as the top left corner and the goal was fixed as the bottom right corner. The results along with the maps for these cases are shown in figure 6(a), (b) and (c).

In all the cases we can easily see that the algorithm tried to optimize not only the total path length but also the complexity. Both these parameters have different roles to play in the entire algorithm. The total path length tries to optimize the net traversal of the robot. At the same time the complexity factor tries to return a path that would be simple to work with using the FIS planner as it may be easily smoothened. Further lesser turns would mean more straightness and a higher speed of travelling. This would in turn give a lot of advantage to the robot to run at a high speed and reach the goal. The FIS planner further did a lot of smoothening of the path generated by the GA. This is clearly seen in the results that show a smooth change in angles as the robot travels. The path thus generated has the discussed advantages.

7. Conclusions

In this paper we had proposed a method to solve the problem of path planning using a combination of evolutionary strategy for evolving robotic path along with FIS Planner. We tested the algorithm for various test cases. In all the test cases we observed that the algorithm was able to find the correct solution. The solutions generated by the algorithm were very optimal. They took relatively small times to solve for each of the map presented. Further, the solutions obeyed the non-holmic constraints and the path generated by the algorithm can be easily used by any robotic controller to move the robot physically.



(a) Map 1

(b) Map 2

(c) Map 3

Fig 6.: The path traced by robot for more maps

The evolutionary strategy adopted was very useful in path planning in static environments where the planning led to very small paths being generated which had a broad area that was highly straight. The straightness of the path was mainly due to the preference given to the shorter solutions with lesser complexity.

The FIS Planner used GA for parametric optimizations. This would enable the solution to be adapted to any type of robot or any type of map as per the problem requirements. This gives sufficient freedom for the person implementing to decide its own set of requirements and constraints. The algorithm would adapt itself to these constraints and generate the system accordingly.

The experimentation done in this paper was mainly on static environment. The algorithm may be further adapted for modeling mobile obstacles in a space time graph. This would make an effective algorithm that is also able to predict their movements. Formulation of more benchmark problems covering all cases that a robot may face in real life may also be done for the genetic optimizations. Further the FIS rules may be more intensively studied and modified in the future. The algorithm is yet to be studied

in a way that balances or tunes the effects of the Evolutionary Path Planning with that of the FIS planner.

References

1. Lima, Pedro U. & Custodio, Luis M., "Multi-Robot Systems", Innovations in Robot Mobility and Control, Springer 2005, pp 1-64
2. Lozano-Pérez, T. & Wesley, M. A., "An algorithm for planning collision-free paths among polyhedral obstacles", Communications of the ACM, pp 560–570, 1979.
3. Lee, Ching-Hung & Chiu, Ming-Hui, "Recurrent neuro fuzzy control design for tracking of mobile robots via hybrid algorithm", Expert Systems with Applications, (2009), doi:10.1016/j.eswa.2008.11.051
4. Ge, Shuzhi Sam & Lewis, Frank L., "Autonomous Mobile Robot", Taylor and Francis, 2006
5. Kavraki, L.E.; Svestka, P.; Latombe, J.-C. & Overmars, M.H., "Probabilistic roadmaps for path planning in high-dimensionalconfiguration spaces", IEEE Transactions on Robotics and Automation, Volume: 12, Issue: 4, pp 566-580, Aug 1996
6. Bohlin, R. & Kavraki, L.E., "Path planning using lazy PRM", Proceedings. IEEE International Robotics and Automation ICRA '00, Vol: 1, pp: 521-528, April 2000, San Francisco, CA, USA
7. Kala, Rahul; et. al., "Mobile Robot Navigation Control in Moving Obstacle Environment using Genetic Algorithm, Artificial Neural Networks and A* Algorithm", Proceedings of the IEEE World Congress on Computer Science and Information Engineering (CSIE 2009), ieeexplore, April 2009, Los Angeles/Anaheim, USA
8. Thrun, Sebastian, "Learning metric-topological maps for indoor mobile robot navigation", Artificial Intelligence, Volume 99, Issue 1, February 1998, Pages 21-71
9. Castejon, Cristina; Blanco, Dolores; Boadai, Beatriz L.& Moreno, Luis, "Voronoi-Based Outdoor Traversable Region Modelling", Innovations in Robot Mobility and Control, Springer 2005, pp 1-64
10. Sud, Avneesh, et. al., "Real-Time Path Planning in Dynamic Virtual Environments Using Multiagent Navigation Graphs", IEEE Transactions on Visualization and Computer Graphics, Vol. 14, No. 3, May/June 2008, pp 526-538.
11. Ng, K.C. & Trivedi, M.M., "A neuro-fuzzy controller for mobile robot navigation and multirobotconvoying", IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, Volume: 28, Issue: 6, ,pp 829-840, Dec 1998
12. Pozna, Claudiu, et. al., "On the design of an obstacle avoiding trajectory: Method and simulation", Mathematics and Computers in Simulation, (2009), doi:10.1016/j.matcom.2008.12.015
13. Tsai, Chi-Hao; Lee, Jou-Sin & Chuang, Jen-Hui, "Path Planning of 3-D Objects Using a New Workspace Model", IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews, Vol. 31, No. 3, August 2001, pp 405-410
14. Hui, Nirman Baran & Pratihar, Dilip Kumar, "A comparative study on some navigation schemes of a real robot tackling moving obstacles", Robotics and Computer-Integrated Manufacturing, (2009), doi:10.1016/j.rcim.2008.12.003
15. Jolly, K.G.; Kumar, R. Sreerama & Vijayakumar, R., "A Bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits", Robotics and Autonomous Systems, Volume 57, Issue 1, 31 January 2009, pp 23-33
16. Goel, Ashok K, "Multistrategy Adaptive Path Planning", IEEE Special Feature, pp57-65, December 1994
17. Kambhampati, Subbarao & Davis, Larry S., "Multiresolution Path Planning for Mobile Robots", IEEE Journal of Robotics and Automation, Vol RA-2, No. 3, September 1986, pp135-145
18. Hwang, Joo Young; Kim, Jun Song; Lim, Sang Seok & Park, Kyu Ho, "A Fast Path Planning by Path Graph Optimization", IEEE Transaction on Systems, Man, and Cybernetics—Part A: Systems and Humans, Vol. 33, No. 1, January 2003, pp 121-128
19. Urdiales, C.; Bantlera, A.; Arrebola, F. & Sandova1, F., "Multi-level path planning algorithm for autonomous robots", IEEE Electronic Letters, 22nd January 1998, Vol. 34 No. 2, pp 223-224
20. Alvarez, Alberto; Caiti, Andrea & Onken, Reiner, "Evolutionary Path Planning for Autonomous Underwater Vehicles in a Variable Ocean", IEEE Journal of Oceanic Engineering, Vol. 29, No. 2, April 2004, pp 418-429
21. Juidette, H. & Youlal, H., " Fuzzy dynamic path planning using genetic algorithms", IEEE Electronics Letters, 17th February 2000 Vol. 36 No. 4, pp 374-376
22. Xiao, Jing; Michalewicz, Zbigniew; Zhang, Lixin & Trojanowski, Krzysztof, "Adaptive Evolutionary Planner/Navigator for Mobile Robots", IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, April 1997, pp 18-28
23. Lin, Hoi-Shan; Xiao, Jing & Michalewicz, Zbigniew, "Evolutionary Algorithm for Path Planning in Mobile Robot Environment", Proceedings of the First IEEE Conference on Evolutionary Computation (ICEC'94), pp 211-216, Piscataway, New Jersey, June 1994. Orlando, Florida
24. Shukla, Anupam; Tiwari, Ritu & Kala, Rahul; "Mobile Robot Navigation Control in Moving Obstacle Environment using A* Algorithm", Intelligent Systems Engineering Systems through Artificial Neural Networks, ASME Publications, Vol. 18, pp 113-120, Nov 2008
25. Shibata, Takanori; Fukuda, Toshio & Tanie, Kazuo, "Fuzzy Critic for Robotic Motion Planning by Genetic Algorithm in Hierarchical Intelligent Control", Proceedings of 1993 International Joint Conference on Neural Networks, pp 77-773
26. Cortes, Juan; Jaillet, Leonard & Simeon, Thierry, "Disassembly Path Planning for Complex Articulated Objects", IEEE Transactions on Robotics, Vol. 24, No. 2, April 2008, pp 475-481

27. Ordonez, Camilo; Collins Jr., Emmanuel G.; Selekwa, Majura F. & Dunlap, Damion D., "The virtual wall approach to limit cycle avoidance for unmanned ground vehicles", Robotics and Autonomous Systems, Volume 56, Issue 8, 31 August 2008, pp 645-657
28. Zhu, David & Latombe, Jean-Claude, "New Heuristic Algorithms for Efficient Hierarchical Path Planning", IEEE Transactions on Robotics and Automation, Vol. 7, No. 1, February 1991, pp 9-20
29. Lai, Xue-Cheng; Ge, Shuzhi Sam & Mamun, Abdullah Al, "Hierarchical Incremental Path Planning and Situation-Dependent Optimized Dynamic Motion Planning Considering Accelerations", IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics, Vol. 37, No. 6, December 2007, pp 1541-1554
30. Jan, Gene Eu; Chang, Ki Yin & Parberry, Ian, "Optimal Path Planning for Mobile Robot Navigation", IEEE/ASME Transactions on Mechatronics, Vol. 13, No. 4, August 2008, pp 451-460
31. Chen, Liang-Hsuan & Chiang, Cheng-Hsiung, "New Approach to Intelligent Control Systems With Self-Exploring Process", IEEE Transactions on Systems, Man and Cybernetics—Part B: Cybernetics, Vol. 33, No. 1, pp 56-66, February 2003
32. Carpin, Stefano & Pagello, Enrico, "An experimental study of distributed robot coordination", Robotics and Autonomous Systems, Volume 57 , Issue 2, pp 129-133, February 2009
33. Pradhan, Saroj Kumar; Parhi, Dayalramakrushna & Panda, Anup Kumar, "Fuzzy logic techniques for navigation of several mobile robots", Applied Soft Computing, Volume 9 , Issue 1, pp 290-304, January 2009
34. Peasgood, Mike; Clark, Christopher Michael & McPhee, John, "A Complete and Scalable Strategy for Coordinating Multiple Robots Within Roadmaps", IEEE Transactions on Robotics, Vol. 24, No. 2, April 2008. pp 283-292
35. Hazon, Noam & Kaminka, Gal A., "On redundancy, efficiency, and robustness in coverage for multiple robots", Towards Autonomous Robotic Systems 2008,Volume 56, Issue 12, 31 December 2008, pp 1102-1114
36. O'Hara, Keith J.; Walker, Daniel B. & Balch, Tucker R., "Physical Path Planning Using a Pervasive Embedded Network", IEEE Transactions on Robotics, Vol. 24, No. 3, June 2008, pp 741-746