Robotic Path Planning in Static Environment using Hierarchical Multi Neuron Heuristic Search and Probability based Fitness

Rahul Kala*, Anupam Shukla, Ritu Tiwari
Soft Computing and Expert System Laboratory
Indian Institute of Information Technology and Management Gwalior, Gwalior, MP, INDIA

* Corresponding Author
rahulkalaiiitm@yahoo.co.in
Room No. 101, Boys Hostel-1, ABV-IIITM Gwalior, Morena Link Road, Gwalior, MP-474010, INDIA
Ph: +91 9993746487

Citation: R. Kala, A. Shukla, R. Tiwari (2011) Robotic path planning in static environment using hierarchical multi-neuron heuristic search and probability based fitness, *Neurocomputing*, 74(14-15), 2314-2335.

Final Version At: http://www.sciencedirect.com/science/article/pii/S0925231211001470

Abstract
Path Planning is a classical problem in the field of robotics. The problem is to find a path of the robot given the various obstacles. The problem has attracted the attention of numerous researchers due to the associated complexities, uncertainties and real time nature. In this paper we propose a new algorithm for solving the problem of path planning in a static environment. The algorithm makes use of an algorithm developed earlier by the authors called Multi Neuron Heuristic Search (MNHS). This algorithm is a modified A algorithm that performs better than normal A* when heuristics are prone to sharp changes. This algorithm has been implemented in a hierarchical manner, where each generation of the algorithm gives a more detailed path that has a higher reaching probability. The map used for this purpose is based on a probabilistic approach where we measure the probability of collision with obstacle while traveling inside the cell. As we decompose the cells, the cell size reduces and the probability starts to touch 0 or 1 depending upon the presence or absence of obstacles in the cell. In this approach, it is not compulsory to run the entire algorithm. We may rather break after a certain degree of certainty has been achieved. We tested the algorithm in numerous situations with varying degrees of complexities. The algorithm was able to give an optimal path in all the situations given. The standard A* algorithm failed to give results within time in most of the situations presented.*

Keywords: Path Planning, Robotics, Multi Neuron Heuristic Search, A* Algorithm, Heuristics, Probabilistic Fitness, Hierarchical Algorithms.

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 [21]. The highly multi-disciplinary nature makes it an exciting playground for people from various fields to collaborate and contribute. The whole problem of robotics include taking input from sensors, making of robotic world map, path planning, robot control [20], multi-robot coordination, high-end planning, etc [7]. The application of AI and Soft Computing techniques in the recent years deserves a special mention.

Path Planning [18] is a specific problem in case of robots where we are given a map of the world. Through this we can come to know about the various paths and obstacles. The problem is to compute a path for the robot that can make it reach a specific goal starting from a specific position. The solution of this problem is a path using which the robot can reach its goal without colliding with any of the obstacles. The problem is usually studied in two separate heads. These are path planning in static environment and path planning in dynamic environment. In static environment the obstacles are static and do not change their position with respect to time. On the other hand, in dynamic path planning the position of obstacles may change with time. 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 [2]. It must also ensure that the algorithm gives its result within the specified amount of time [16].

The problem of path planning takes its input a map. A robotic map is a representation of the world of the robot [23]. The map depicts the traversable path, obstacles, surface and other information. Various types of paths may be drawn depending upon the problem and solution requirements. Some of the commonly used maps include topological maps [34], voronoi maps [4, 33], hybrid maps [4], etc. The map is built using the results of the various sensors along with the output of the recognition systems [7].

Path planning usually gives its output to the robotic control. This consists of a robotic controller which is supposed to move the robot in the desired path. Various robot controllers have been designed. Some of them are built using the Adaptive Neuro-Fuzzy architecture [24].

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 solve the prevalent research issues to varying extents. A class of algorithms uses the potential method approach to navigate a robot [28]. 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 et al. solved the problem using a potential field approach for obstacle avoidance in [28]. Other potential field methods include [35]. Hui and Pratihar gave a comparison between the Potential Field and the Soft Computing solutions in [11]. Various statistical approaches have also been used. This includes the work of Jolly et al. who used Bezier Curve for path planning [14]. Goel solved the problem for dynamic obstacles using an adaptive strategy [8]. Quad Tree [17], Mesh [12], Pyramid [36] are representations that have been tried for better performance. Another good amount of work exists using the Soft Computing approaches especially Genetic Algorithms [1, 15, 37, 22], A* Algorithms [32] and artificial neural networks (ANN) [16]. Shibata used Fuzzy Logic for fitness evaluation of the paths generated [30]. Various other approaches [6, 26] have also been proposed.

Zhu and Latombe used the concepts of cell decomposition and hierarchal planning in [38]. Here they represented the cells in a similar concept of grayness denoting the obstacles. Urdylis et al. used a multi-level probability based pyramid for solving the path planning problem [36]. Hierarchical Planning can also be found in the work of Lai et al. [19] and Shibata et al. [30].

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 et al. presented his work for solving in 3 degrees of freedom [13].

Chen and Chiang 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 [5]. In the field of multi-robot systems, Carpin and Pagello used an approximation algorithm to solve the problem of robotic coordination using the space-time data structures [3]. They showed a compromise between speed and quality. Pradhan et al. solved a similar problem for unknown environments using Fuzzy Logic [29]. Peasgood et al. solved the multi-robot planning problem ensuring completeness using Spanning Trees [27]. Hazon and Kaminka analyzed the completeness of the multi-robot coverage problem [9]. O' Hara et al. gave the idea of using embedded networks as sensors for solving the problem in [25].

It was earlier shown by the authors that A* generates the best results for the problem of path planning [32]. However, the algorithm was found to be computationally very expensive. It was also earlier shown by the authors that A* algorithm does not perform well in problems like maze solving [31]. The motivation behind this work is to adapt A* algorithm in such a way that it becomes more scalable and performs well in finite time. We do this with the application of probability based map representation [17].

In this paper we have proposed the use of Multi Neuron Heuristic Search (MNHS) for the purpose of solving the path planning problem [31]. This was an algorithm proposed earlier by the authors as an improvement over the standard A* algorithm. This algorithm was found to perform better in situations where the heuristic function may fluctuate very quickly between adjacent nodes in the graph. The MNHS improves performance in situations where the heuristic function behavior is uncertain by expanding multiple heuristic nodes simultaneously, in place of the node with the best heuristic as was the case with conventional A* algorithm.

The map in this paper has been modeled in a way similar to the quad tree approach as used by [17]. In this model the graph was divided into a set of nodes of varying sizes. All nodes are arranged in layered manner from top to bottom, each with different size of nodes, and different level of uncertainty. Here at the top the nodes have high degree of uncertainty regarding the presence or absence of the obstacles. This uncertainty is measured in terms of 'grayness' of the cell. A completely white cell denotes the absence of any kind of obstacle from the entire region. On the other hand full black color denotes the conformation of presence of obstacles in the cell. The grey denotes the intermediate values with the intensity denoting the probability of the cell being free from obstacles.

In practical problems, the map may be too large for the standard MNHS algorithm to perform. This problem is solved by making a hierarchical solution to the problem. The first little iterations are supposed to find the initial vague solutions with lesser details. The probability of collision with obstacles is undetermined to a large extent in the first little iteration. The size of the cells is quite large. Towards the later iterations of the algorithm the cells lying on the possibly optimal path are decomposed. As a result path keeps adding details. Also the presence or absence of the obstacles keeps getting clear.

This paper is organized as follows. Section 2 introduces the MNHS algorithm. In section 3 we introduce the general outline of the algorithm. The hierarchical approach is introduced in section 4. The concept of probability based fitness is discussed in section 5. In section 6 we discuss the simulation model and the results. Section 7 gives the comparative analysis and section 8 presents the results.

2. Multi Neuron Heuristic Search (MNHS)

The MNHS is an improved form of A* algorithm that works in the cases where the heuristics are prone to sudden fluctuations [31]. The main aim of heuristic based search algorithm is to constantly drive the system towards better states. But suppose you keep moving in the graph search minimizing the cost, and suddenly there is a sharp increase in the heuristic value of the traversed node. In such a case the entire path might becomes useless, and the algorithm might need to backtrack to find alternative paths. Hence these algorithms suffer. A solution might be in such cases to completely neglect the heuristics and go with the other algorithms like Breadth-First Search and Depth-First Search. But that does not sound a good strategy having some idea of the heuristics.

The MNHS algorithm can be applied to the cases where the following problems occur in heuristic function:

- The heuristic function reaches near goal, but suddenly shows that no way is possible to reach goal.
- The heuristic function keeps fluctuating from the good values to bad values making it hard to predict the goal.
- The heuristic function drops suddenly from very high value to low value.

These conditions can easily be understood from the problem of maze solving, if the heuristic function of any point (x,y) on the maze denotes its distance from the goal. If the search algorithm reaches last but one position and then finds itself surrounded by walls, the heuristics increase suddenly. Similarly if the solution is a series of bad moves followed by another series of good moves, the heuristics decrease from high to low. Hence in such problems though we may take the heuristic function, its performance would be low. In such cases the MNHS serves better that respects all the good, bad and moderate values of heuristics, so that no value suffers.

The basic idea of this algorithm is the use of many neurons working one after the other. Each of these processes the nodes with varying heuristic values in the open list of the search algorithm. The algorithm hence gives respect to all values of the heuristics. It may be seen as the way of employing different neurons for different types of works and whichever finds the target, is rated successful. If you were to find a treasure, it would be justified to divide your team at various places, some at high probability places, and some at low.

In all we take α neurons. The open list consists of heuristic costs, each corresponding to node seen but waiting to be processed. We divide the entire cost range into α equal ranges among these neurons. Each of these neurons is given a particular range. Each neuron selects the minimum most element of the cost range allotted to it and starts searching. At one step of each neuron processes its element by searching and expanding the element. This process is repeated.

The following is the algorithm:

Step 1: open ← empty priority queue
Step 2: closed ← empty list
Step 3: add a node n in open such that position(n) = CurrentPosition, previous(n) = null and f(n), g(n), h(n) are as calculated by respective formulas with priority f(n)
Step 4: while open is not empty
Begin
Step 5: extract the node n_1, n_2, n_3, n_4….. n_α from open with the priority of n_1 as highest and the others equally distributed between other α-1 nodes.
Step 6: if n_i = final position for i=1,2,3,4,5…..α then break
Step 7: else
Step 8: nodes ← nodes from the expanding of node n_i
Step 9: for each node m in nodes
Begin
Step 10: if m is already in open list and is equally good or better then discard this move
Step 11: if m is already in closed list and is equally good or better then discard this move
Step 12 delete m from open and closed lists
Step 13: make m as new node with parent n
Step 14: calculate f(m), h(m), g(m)
Step 15: Add node m to open with priority f(m)
End
Step 16: Add n to closed
Step 17: Remove n from open
End
End

Here for any node n the total cost f(n), is given by (1)

$$f(n) = g(n)+h(n) \quad (1)$$

Here,
h(n) = heuristic cost which is taken to be the Euclidian distance from the node n to the final position
g(n) = historic cost
f(n) = is the total cost.

3. Algorithm

In this section we give a general outline to the algorithm that we have developed for solving the problem of robotic path planning. The basic methodology is to use the MNHS algorithm in a hierarchical manner. The concept of probability based fitness has also been introduced. The general structure of the algorithm is given in figure 1.

The algorithm starts by taking as input the initial graph. This graph may be approximately built by the map building algorithm. This means that we do not know for sure whether obstacle lies at some cell or not [17]. The MNHS starts the search by processing of its open list, which initially comprises of just the source node. On successful termination, the MNHS algorithm returns a path. The path has a certain probability of collision associated with it depending upon the probability of the occurrence of obstacles at the nodes [36]. We then decompose all the cells that were traversed in this path. The decomposition may be defined as adding of more details by the map algorithm or the breaking up of cells into smaller cells for more accuracy in the prediction of presence or absence of obstacles. If the path cannot be further broken down, the algorithm would terminate.

Another concept introduced in the algorithm was the transfer of points from one generation to the other. It would be a waste of computation to start the MNHS algorithm from scratch the next time, since a lot of information is already available from the previous run. Rather than starting the algorithm from scratch, we set some initial points in the open and closed list of the MNHS algorithm. These points are taken from the open list of the previous MNHS run.

They are recomputed according to the present modified graph before transferring them to the next generation. After the transfer of points to the next generation is over, the MNHS is made to run again on a finer graph. This transfer is motivated by Genetic Algorithms [10].

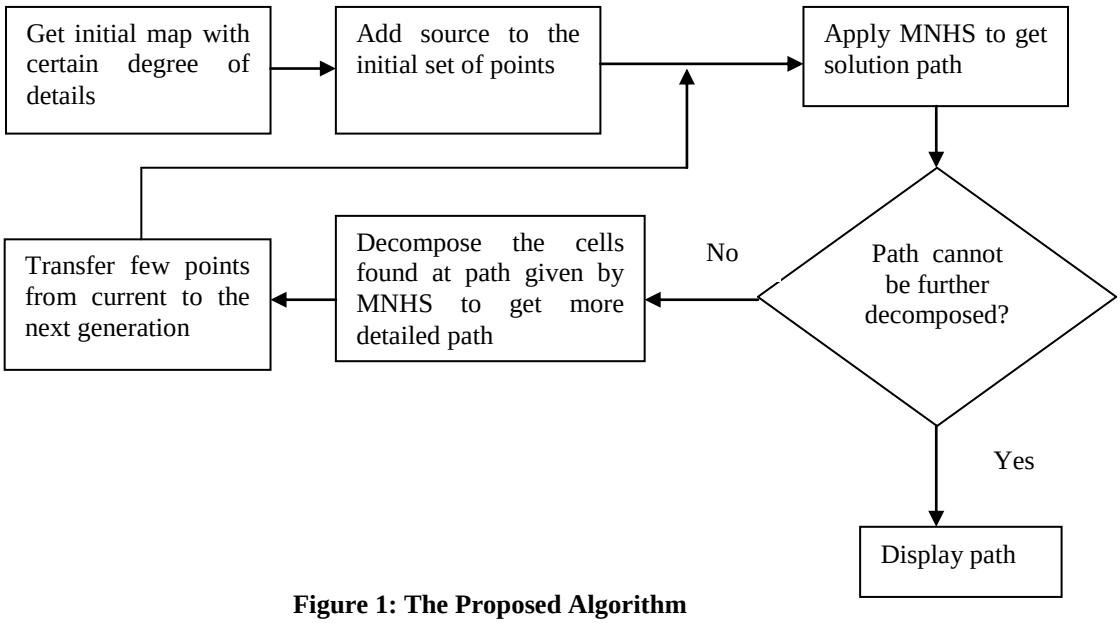


Figure 1: The Proposed Algorithm

The algorithm also has a provision for external termination. Many times, due to the real time nature of the algorithm, the result is required at any specific point of time [16, 32]. Here we may not be in a position to wait for the algorithm to continue. In such a case this algorithm may be externally terminated and robot can be moved in the best path generated so far.

We discuss the various steps of the algorithm one by one.

3.1 Map

Map is the representation of the robotic world. The path planning algorithm uses map to get information about the obstacles and paths that are accessible. In this paper we have used a probabilistic approach to represent a map. The map is represented in the form of grids of variable sizes. Each grid is marked with a color from white to black through grey. The color denotes the probability of the obstacles lying in that region. White means no obstacle and black means a confirmed presence of obstacle. The search operation operates on a graph and it is hence necessary to convert the map into a graph. The graph is a collection of vertices (or nodes) and edges. In the graph each grid represents a vertex and all grids that are accessible by other grids (irrespective of presence or absence of obstacles) are marked as edges. One such representation of the map is given in figure 2(a). The corresponding graph may be given by figure 2(b).

The grayness is the measure of the obstacles in the cell. A gray value of 0 means cell is fully covered by obstacles and vice versa. Please note that this is the convention followed in the algorithm, with lower numeric value denoting a higher obstacle count. The grayness of any cell c may be calculated by using equation (2).

$$Grey(c) = 1 - \frac{Total\ area\ covered\ by\ obstacles}{Total\ area\ of\ the\ cell} \quad (2)$$

Suppose the whole map is built on a unit map consisting of unit cells. The formula given by (1) in such a case converts into (3).

$$Grey(c) = 1 - \frac{Total\ unit\ cells\ occupied\ by\ obstacles}{Total\ unit\ cells\ covered\ by\ c} \quad (3)$$

It is natural that grayness of any cell lies between 0 and 1 and hence denotes the probability of collision of the robot in that cell.

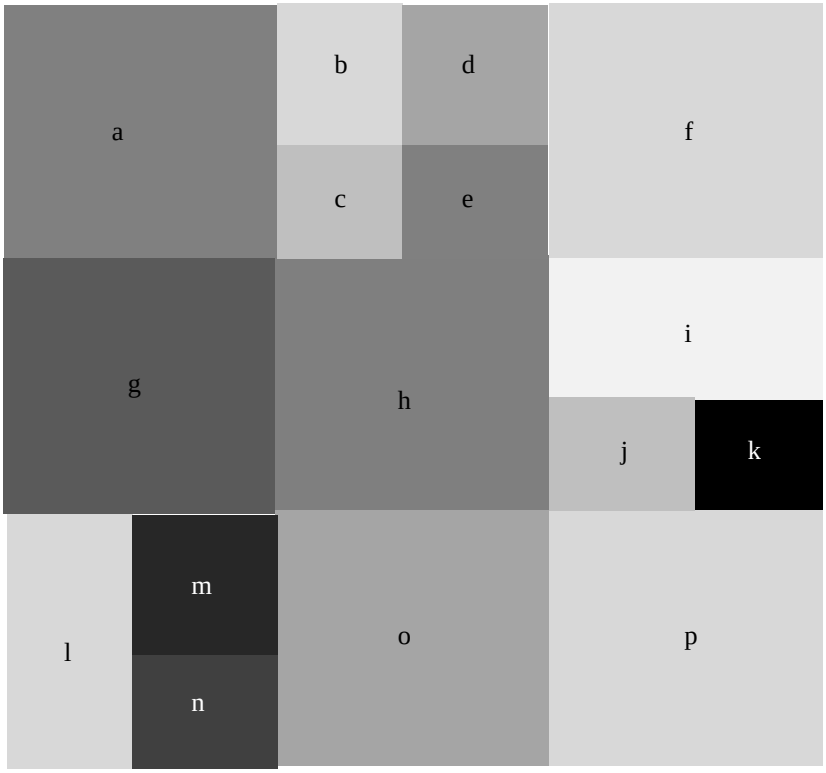


Figure 2(a): The map at any general point of time

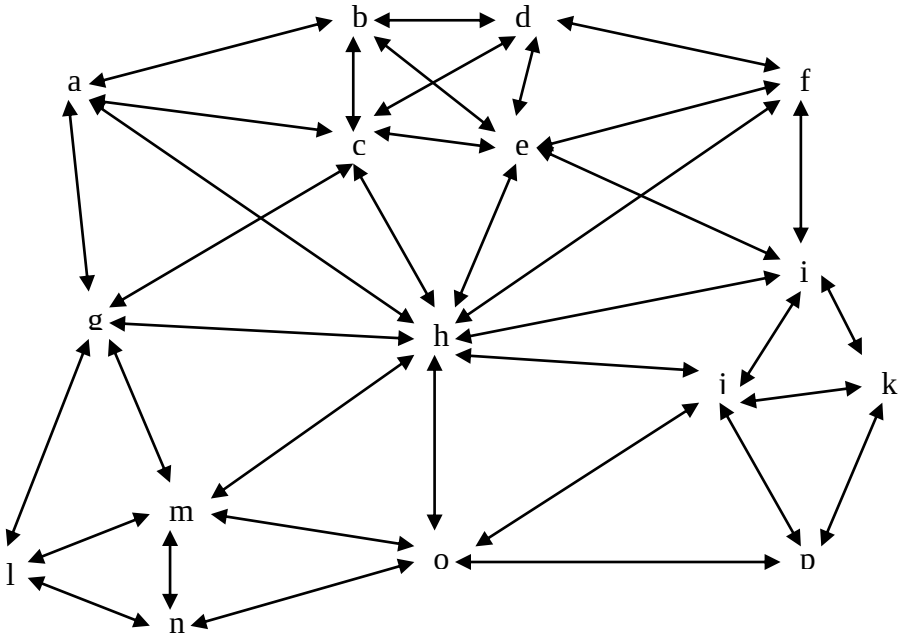


Figure 2(b): The equivalent graph

Initially we start with a graph of size M grids X N grids. A grid represents the smallest possible division in the graph. As the algorithm proceeds, we keep decomposing the cells into smaller cells. This is repeated numerable number of times, unless it is not possible to decompose any grid of the path given by the algorithm. The shape given in figure 2 is formed as a result of multiple decompositions of multiple grids over time.

3.2 MNHS Algorithm

We discussed the principles and working of the MNHS in section 2. In this section we discuss the basic motivation behind the use of MNHS and the key features of the algorithm that we use for solving the problem of path planning.

The research so far has been using path planning for relatively simple paths. Researchers try to place obstacles in the way and try to see the behavior of the robots. In practical life, it can never be assumed that the path would be so simple. The reason is the numerous possibilities of obstacles in a variety of ways. Consider a robot cleaning a house. There would be multiple paths possible with numerous obstacles of varying sizes. The robot is supposed to avoid all of them and reach the destination. The scalability of algorithms is quite limited in nature. Planning with complex maps can hence be difficult. An example would be the maze like structure where a robot has to find its way out of the maze.

The MNHS algorithm takes care of these problems by trying to exploit each and every path possible. This has a multiplying effect on the time complexity, but in return is an assurance in case of maps with rapid change in heuristics. The extra exploration by the algorithm would come to rescue if the robot reaches quite near to the goal only to find that there is no way to reach it.

Along with possibilities of early discovery of goal for complex maps, the benefits of MNHS in this algorithm further extend to maintenance of backup paths between iterations. Since, we are denoting each path by a probability measure at each stage, it is possible that after certain levels of decomposition we realize that the path is infeasible due to the presence of some obstacle. This is a very common problem in such probability based algorithms. In such a situation, starting right from scratch would be wasteful. Even if we start again from scratch, the same results might come again where it is impossible to reach the goal. The MNHS is an improvement as it always generates many paths of varying lengths. Some of these paths are almost near to the goal whereas some are just near the source as the algorithm proceeds. Even if the best path fails, due to the sudden exposure of some obstacle, we can always continue with the second best path which must have been expanded to a good degree.

This concept is shown in figure 3. Here the best path has almost reached the goal. At the same time the other paths have been expanded to a reasonably good degree that are ready to provide a backup. The obstacles have not been shown in the figure. The figure is just meant to explain the general concept of the algorithm. For details please refer [31].

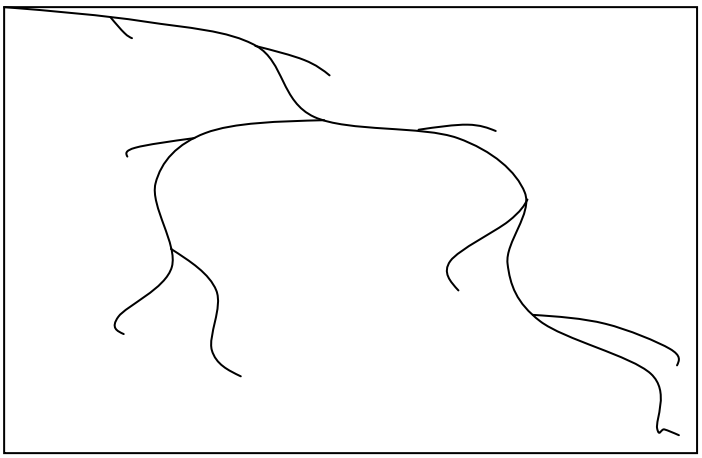


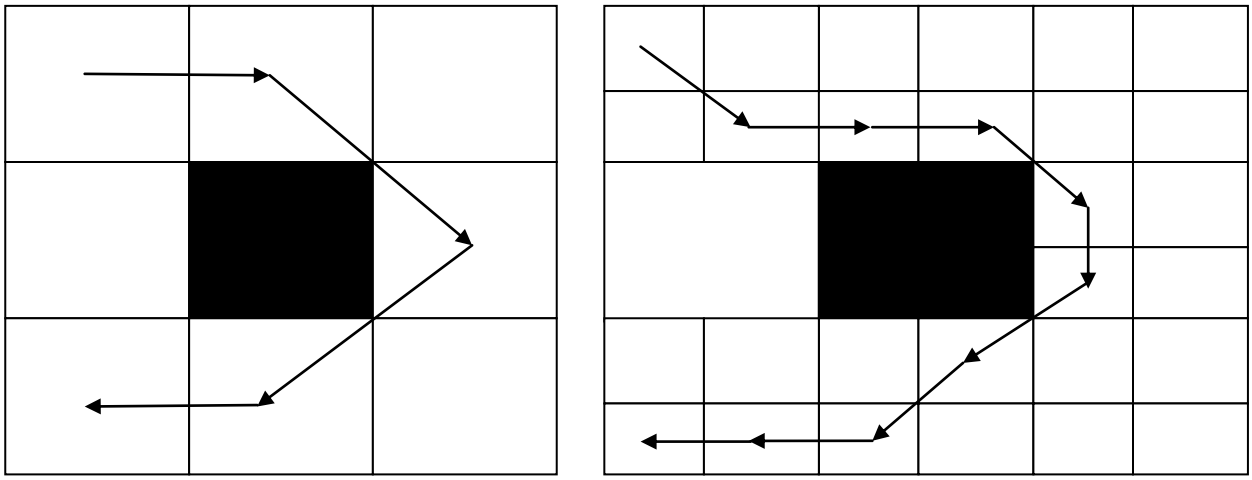
Figure 3: The MNHS path exploration

3.3 Decomposition of cells

The algorithm is advancement over the existing algorithm due to the hierarchical decomposition of cells that the algorithm follows. The decomposition of any cell gives a set of finer cells. These cells are more detailed and smaller in size as compared to the original cell. These cells carry more information regarding the presence or absence of the obstacles.

It is natural that a cell that is already of unit size cannot be further decomposed. In such a case the color of the cell would be either black or white. Black denotes the presence of some obstacle at the position and white denotes the absence of obstacle.

Also it is natural that we would not be interested in a cell if the probability of the obstacle avoidance (or its color) is almost 1. This is because we can be pretty sure that walking on this cell would most probably not result in a collision. Similarly we would not be interested in a cell decomposition if its probability (or its color) is almost 0. This is because of the fact that we can be pretty sure that walking on this cell would most probably result in an error. The only purpose for decomposing these cells is to get finer paths that are more realistic to the shortest path. This is given in figure 4.



(a) The original path **(b) Path after decomposition**

Figure 4: Decomposing gives finer paths

In this algorithm we decompose all cells that lie on a successful path found by the MNHS. This decomposition divides any cell into 4 cells of equal size. The color is recalculated for all the cells. If again these cells form part of a successful path, they are again broken down. This is given in figure 5.

3.4 Generations

In this algorithm, we use the concept of generations similar to that used in Genetic Algorithms (GA) [10]. Here the algorithm operates in generations and the solution keeps improving between generations. The characteristics are passed from one generation to the other. The newer generations are found to adapt well to the changing environment.

Unlike the GA, we do not pass the entire population with all characteristics from one generation to the other. The reason behind this is that the environment changes between generations. We cannot recalculate the entire population set according to the new environment. That would make the whole algorithm very expensive. We can only recalculate the effect of change in environment on some of the populations.

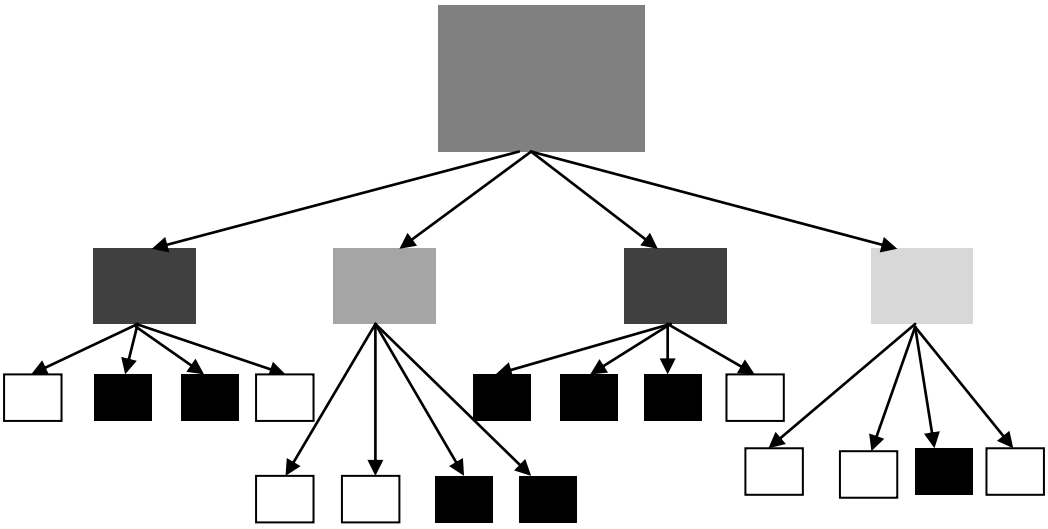


Figure 5: The decomposition of cells

Population in this case refers to the open and the closed list of the MNHS. It is natural that with the change in environment, the entire heuristics would change. The vertices in MNHS that earlier had poor heuristic values may now have good heuristic values and vice versa.

In total we first select β points from the current open list of the MNHS. Since we cannot say with guarantee what would be the effect of the changing environment on the points, we hence select these β populations of varying costs from the best to the worst. The advantages of the MNHS are also applicable to this approach.

Then we need to recalculate the new costs of these β values. So we trace their path according to the old map. All the nodes visited are added in a set called as the reduced graph set. If any point in any of the paths in the reduced graph set was decomposed by the decomposition algorithm, we replace the point with all the decomposed points. This gives us modified reduced graph set with all new nodes. We then run a local A* search on the points in the reduced graph set to calculate the new paths of the selected β nodes. This gives us the new paths that lead us to these β nodes.

As we run this A* algorithm to find the new paths of the selected nodes, we also expand the nodes in the entire graph. This gives us more points whose costs are known and we can directly pass to the next generation. This multiplies the number of points that we pass to the next generation without much computational overhead.

All these β nodes along with the other generated nodes are added in the initial set of points for the next run of the MNHS. If these were in the closed list, they are added in the closed list of next MNHS and same in case of open list. These are hence passed from the previous generation to the next generation. If any of these points corresponded to the goal position, it is not passed to the next generation.

This algorithm is given in figure 6.

4 The Hierarchical Approach

We have already discussed some of the hierarchical concepts of the algorithm in terms of generation, map expansion, etc. In this section we study these concepts and the way they go with each other along with the motivation behind the approach. The basic approach used in the problem is that of the MNHS. The whole map has been generalized to reduce number of cells by introducing the probabilistic approach. The search on the reduced graph can hence be easily performed. In the previous sections we saw how the various fragments of the graph can be broken down or decomposed depending upon the need. We also saw how the information was being passed from one generation to the other.

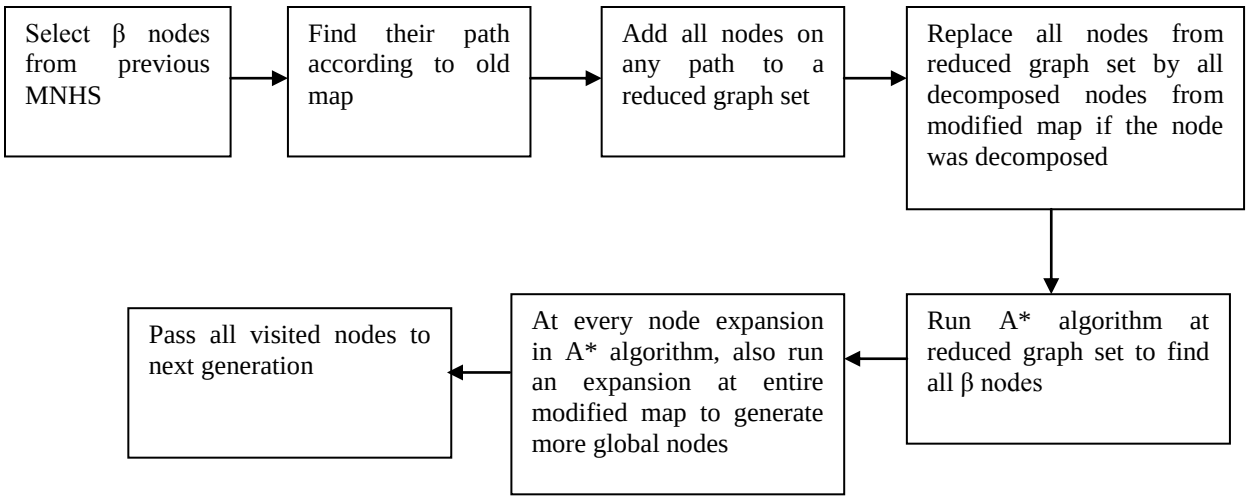


Figure 6: The passing of nodes between generations

Since we are following a probabilistic approach, as we shall see in the next section, every solution generated by the MNHS would be having decent probability of avoidance of obstacles. For this reason, we decompose exactly this path with the hope that the collisions would be avoided. If there are no collisions in the path, then we can expect the algorithm breaking this path down or decomposing this path again and again. The end result would be that we get the entire path. But in reality, the path may face obstacles at any generation. It may also be possible that no other path in close vicinity is possible. The MNHS approach becomes a very useful approach in such cases. In such situations we already have a backup path ready. If this path also fails, another path is ready.

Another important concept that comes from the same problem is that we need proper means to run the MNHS over the solution again and again. This would ideally not be possible as the map changes between generations and all the costs are naturally changed. We saw in the previous section how we pass on the needed information to the next generation so that the MNHS works.

The basic motive behind this approach is that we run a local A* algorithm over a reduced node set. Since this has limited number of nodes in the search domain, it does not take much time. As a result we can get modified values of good number of vertices in smaller amount of time. Re-calculating all values or running the MNHS again would have taken a large amount of time. While we recalculate the values of some points, we somewhere hope that the final path would be using these nodes that the A* algorithm works on and passes to the next generation. If this is true, a large part of the computation of the MNHS is already done. As discussed earlier, if the path being decomposed has no obstacles on further decomposition, the entire path is generated by this A* algorithm in negligible time and there is virtually no need for MNHS to run again. But on the contrary, if we suddenly see an obstacle in this path, then another path needs to be selected. The expansion by the A* algorithm gives a fair idea to the MNHS regarding the points.

The motivation here is to take β points and pass it to the MNHS. If suppose β gets very large, we would end up in recalculating the entire map that was left by the previous MNHS. After recalculation of the entire map, the next iteration of MNHS would continue. The reduced graph set is this case would become very heavy comprising of too many elements making the A* algorithm computationally expensive. However at the same time, the computational load from the MNHS would reduce. If suppose β gets too small. In such a case the A* algorithm would hardly give points. The simulation would be similar to a fresh start of MNHS provided that the previously decomposed graph contains obstacles.

5 Probability based fitness

The MNHS uses the fitness (also called as cost) for its functioning. The cost decides the goodness of the solution. The better solutions have lower costs. Hence the approach is generally to find smaller costs and expand them further.

This algorithm is based on probability that is denoted by the probability of finding obstacles in the cell. Hence each path we traverse has some probability of success associated with it. This probability for a path is given by equation (4).

$$Grey(P) = \prod_i Grey(p_i) \quad (4)$$

Here p_i are the consecutive nodes that make up the total path P.

The other costs that the algorithm uses are Heuristic Cost h(n) and the Historical Cost g(n). For the MNHS, the total cost f(n) is given by the sum of Heuristic Cost h(n) and Historical Cost g(n). In this algorithm we use the historic cost as distance of the center of the node from the source and the historic cost as the cost of the center of node to the goal.

The total fitness C(n) in this case is the probability based total cost. This is given by (5)

$$C(n) = f(n)* Grey(P) +(1-Grey(P)) \quad (5)$$

Here P is the path of traversal till node n.
f(n) has been normalized to lie between 0 and 1.

It may be noted here that (5) has been derived keeping in mind the following points

- If Grey(P) is 0, it means that the path is not feasible. The fitness in this case must have the maximum possible value i.e. 1
- If Grey(P) is 1, it means that the path is fully feasible. The fitness in this case must generalize to the normal total cost value i.e. f(n)
- All other cases are intermediate

The graph of this function is as shown in figure 7.

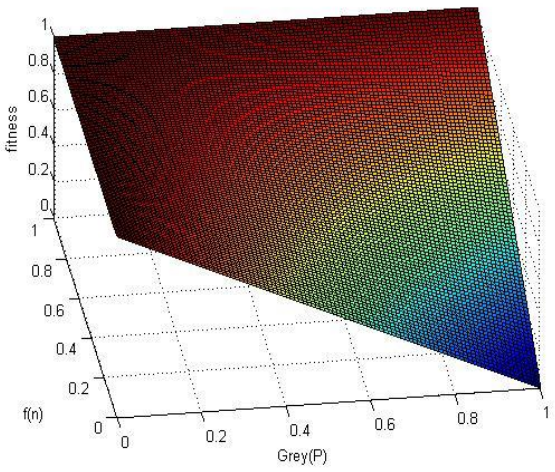


Figure 7: The plot of fitness function

6 Simulation and Results

In order to test the working of the algorithm, we made our own simulation engine. Every attempt was made to ensure that the simulation engine behaves in a way similar to actual robot. This would ensure that the algorithm can be easily deployed to the real robot for the purpose of path planning.

All simulations were done on a 2.0 dual core system with 1 GB RAM using the self generated simulation engine. The simulation engine that was built had the functionality of on-demand cell decomposition, graph generation, time monitoring. Various views were generated during simulation to have a multi-dimensional view of the solution and their creation. The initial graph 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.

We first provide an experimental setup to prove the supremacy of the MNHS over the conventional A* algorithm, as per the claims made in section 2. We take an arbitrary maze-like map as shown in figure 8(a).Here each cell denotes a vertex of the graph used by the graph search algorithm. We give the same map to both the MNHS as well as the A* algorithm. The time any vertex is explored by the corresponding algorithms is noted and displayed in the solutions. It is natural that the time of exploration of the goal needs to be as less as possible for any search algorithm. Figure 8(a) shows the solution generated by the MNHS and figure 8(b) shows the solution generated by A* algorithm. It can be easily seen that MNHS explored much less vertices and reached the goal much early as compared to the A* algorithm. This shows that MNHS is a betterment over the A* algorithm.

01	02	04	06	08	09	11	13	15	17
03	00	00	00	00	00	00	00	00	19
05	10	26	28	35	40	00	00	00	21
07	00	00	00	00	00	00	22	00	00
12	32	00	00	00	00	00	00	00	23
14	00	00	00	00	00	00	00	00	25
16	24	38	00	00	00	00	00	00	27
18	00	00	00	00	00	00	00	00	00
20	29	30	31	33	34	26	37	39	41

Figure 8(a): The solution to maze generated by MNHS

01	02	03	04	05	06	07	08	09	10
17	00	00	00	00	00	00	00	00	11
18	19	20	21	22	23	24	25	00	12
30	00	00	00	00	00	00	26	00	13
31	32	33	34	35	36	00	27	00	14
39	00	00	00	00	37	00	28	00	15
40	41	42	43	00	38	07	29	00	16
44	00	00	00	00	00	00	00	00	00
45	46	47	48	49	50	51	52	53	54

Figure 8(b): The solution to maze generated by A* algorithm

We conducted 4 tests to see the functionality of the algorithm. (1) The first test was on a path with no obstacles. This was done to ensure the robot performs well in situations where there are no obstacles. (2) The second test was using a single obstacle. This tested the capability of the robot to surpass simple obstacles. (3) The third test was the scalability test where the robot was given good number of obstacles. This ensured the capability of the algorithm to solve complex problems. (4) The last was the scattered obstacle test. Here the algorithm was given random small obstacles. This tested the capability of the algorithm to slide through regular small obstacles.

After these tests we study the effect of the change in the initial size of the cell to the performance metrics of the algorithm. We use the 4th test case to study these metrics. The various results are given in the next sub-sections.

6.1 Free Path

The first test we applied was on a plane path. There was no obstacle on the way from source to destination. We saw that the robot was easily able to travel from the source to destination in a simple line. The map given was of dimensions 1000X1000 grids, which was initially broken down into chunks of 50X50 grids. The value of α was kept as 5 and β was kept as 10.

The path traced by the robot at various generations is given in figure 9. Here we observe that the same path was traced at every iteration.

The condition of the grid at start, 3rd generation and 6th (last generation) are given in figure 10(a), (b) and (c) respectively (only the cells are shown, the grayness is not shown in figure).

The total cost, fitness and grayness of the final path at various generations are given by figure 11. Note that for this case no major change is seen in the costs as the total path at every generation was found to be the same.

The cells that are passed from one generation to other are shown in figure 12(a), (b) and (c). These were recorded at three consecutive runs of the algorithm.

It can be seen form figure 12 that the algorithm first doubted the fitness of the path and tried to explore distant points that could have acted as a backup in case the main path failed. As the algorithm continued, due to the very high probability (probability of 1), the effect of other nodes reduced and finally the main path was converted into the final path. The other nodes are still expanded and tried to pass in case the main path fails.

Figure 13 shows the time required (in total) between the various generations.

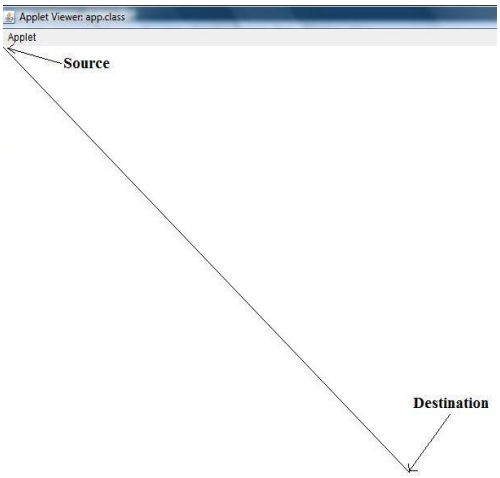


Figure 9: The path traced with no obstacles

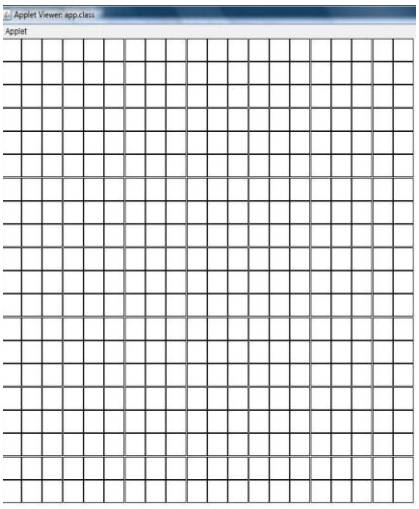


Figure 10(a): The condition of the grid at Start

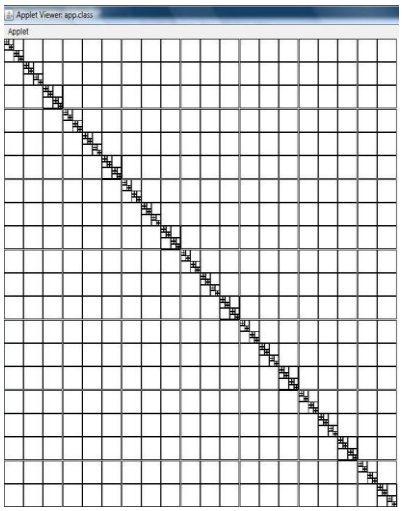


Figure 10(b): The condition of the grid at Generation 3

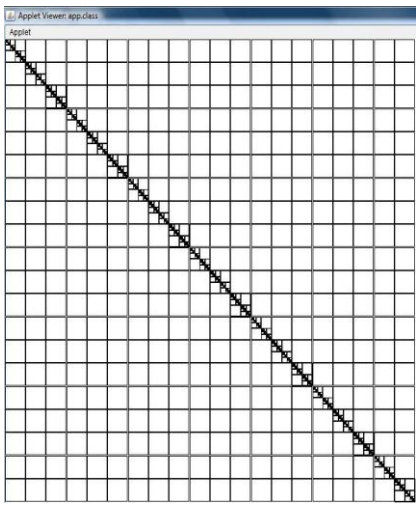


Figure 10(c): The condition of the grid at Generation 6

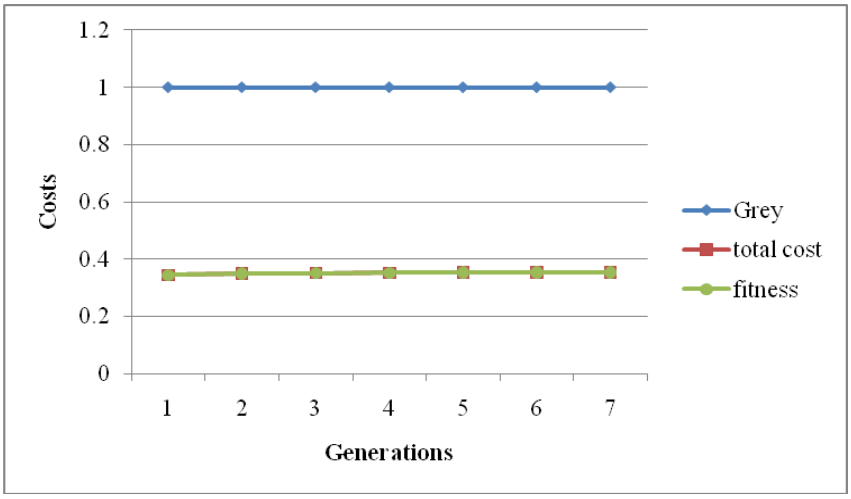


Figure 11: The various costs between generations

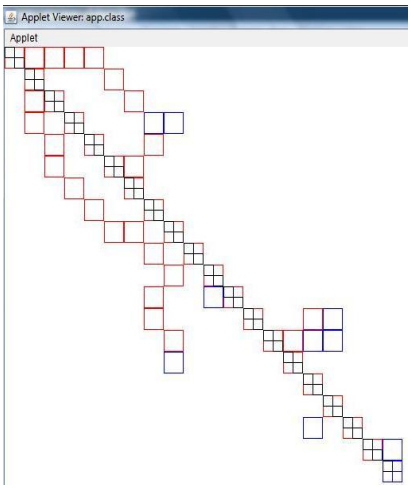


Figure 12(a): The passing of nodes between generation 1st to 2nd generation

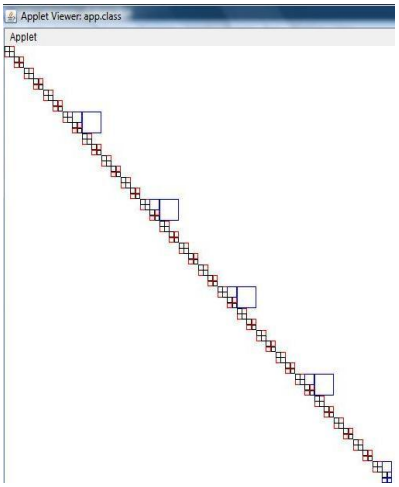


Figure 12(b): The passing of nodes between generation 2nd to 3rd

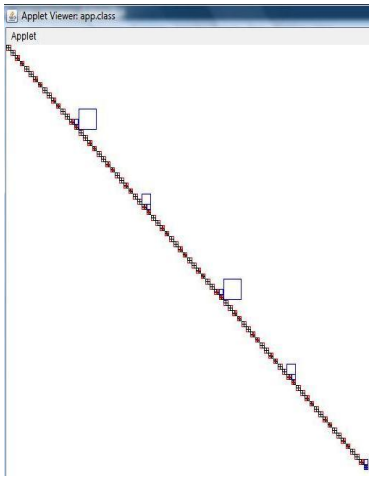


Figure 12(c): The passing of nodes between generation 3rd to 4th

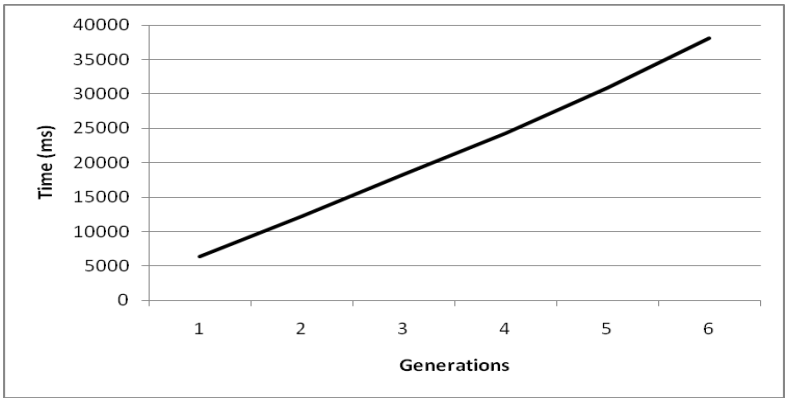


Figure 13: Time required by algorithm v/s generations

6.2 Single Obstacle

The second test we applied on a single obstacle placed on the path between the source and destination. The map is given in figure 14. We saw that the robot was again easily able to travel from the source to destination in a simple line. The map given was of dimensions 1000X1000 grids, which was initially broken down into chunks of 50X50 grids. The value of α was kept as 5 and β was kept as 10.

The path traced by the robot at various generations is given in figure 14(a), (b) and (c). Here we observe that the path traced at the 1st iteration was later found to be with obstacle. The algorithm hence in the next generation discovered another path.

The condition of the grid at 2nd generation, 4th and higher generation are given in figure 15(a), (b) and (c) respectively (only the cells are shown, the grayness is not shown in figure).

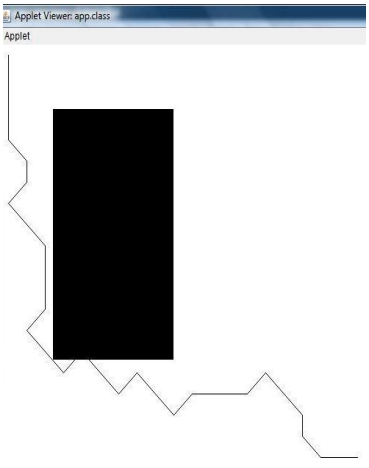


Figure 14(a): The path traced with one obstacle at 1st generation

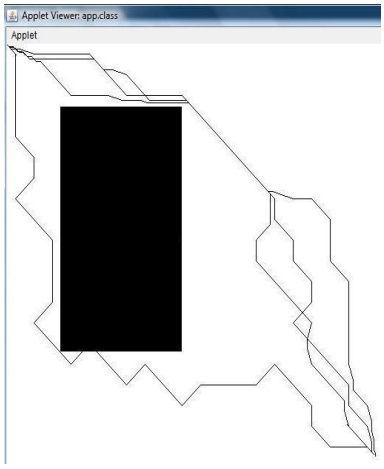


Figure 14(b): The path traced with one obstacle at 2nd generation

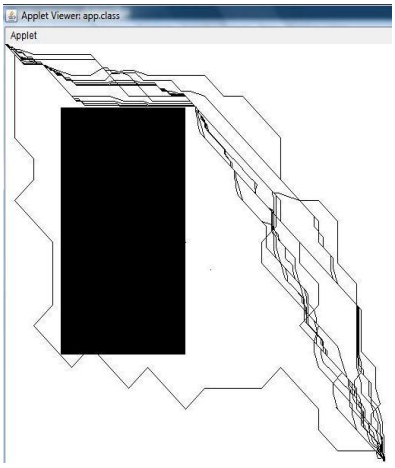


Figure 14(c): The path traced with one obstacle at higher generations

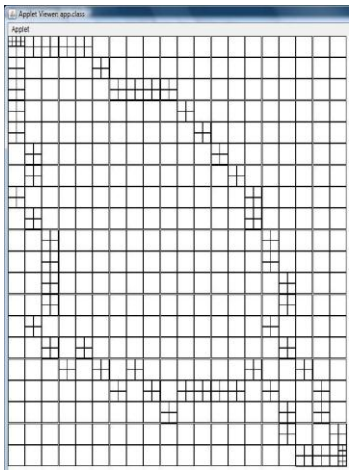


Figure 15(a): The condition of the grid at Generation 2

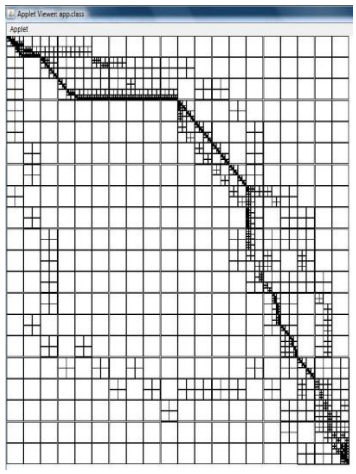


Figure 15(b): The condition of the grid at Generation 4

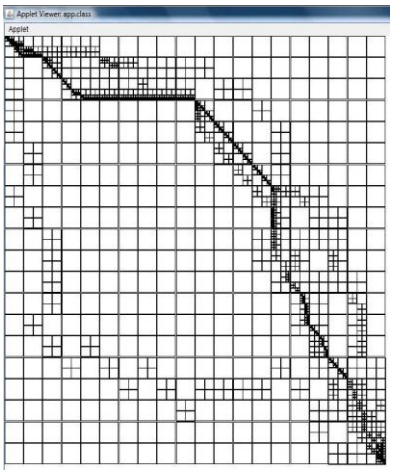


Figure 15(c): The condition of the grid at Higher Generations

The total cost, fitness and grayness of the final path at various generations is given by figure 16. Note that for this case no major change is seen in the costs as the algorithm very easily found the correct path.

The cells that are passed from one generation to other are shown in figure 17(a), (b) and (c).

It can be seen form figure 16 that the algorithm always generated multiple paths. Even though the 1st generated path failed, the information given to the 2nd generation carried points that helped it come up with another path that was quite different. Later this path became predominant as generations increased and hence gave the final path.

Figure 18 shows the time required (in total) between the various generations.

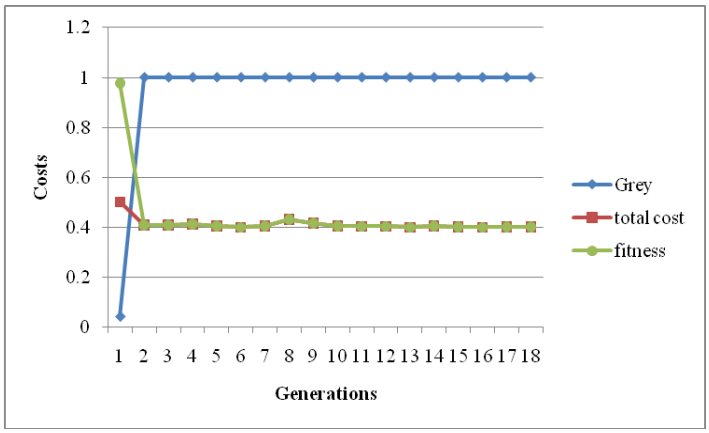


Figure 16: The various costs between generations

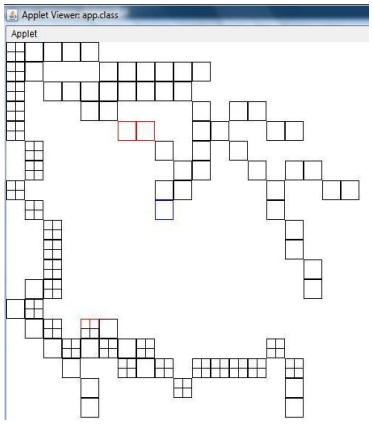


Figure 17(a): The passing of nodes between generation 1st to 2nd generation

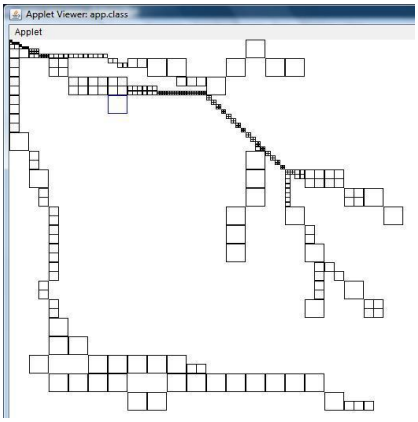


Figure 17(b): The passing of nodes between generation 3rd to 4th generation

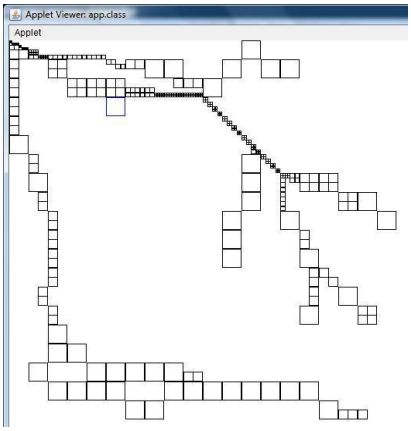


Figure 17(c): The passing of nodes between generation higher generations

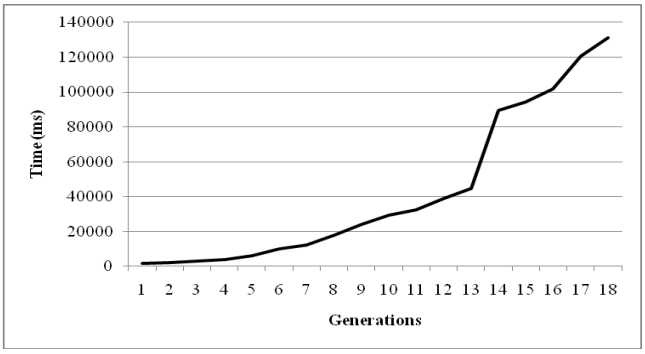


Figure 18: Time required by algorithm v/s generations

6.3 Scalability Test

The next test we applied to test the scalability of the algorithm. Here multiple obstacles were placed in the path of the robot. We saw that the robot was easily able to travel from the source to destination in a simple line. The map given was of dimensions 1000X1000 grids, which was initially broken down into chunks of 50X50 grids. The value of α was kept as 5 and β was kept as 10.

The path traced by the robot at various generations is given in figure 19(a), (b) and (c). The condition of the grid are given in figure 20(a), (b) and (c) respectively (only the cells are shown, the grayness is not shown in figure).

The total cost, fitness and grayness of the final path at various generations is given by figure 21. The cells that are passed from one generation to other are shown in figure 22(a), (b) and (c). Figure 23 shows the time required (in total) between the various generations.

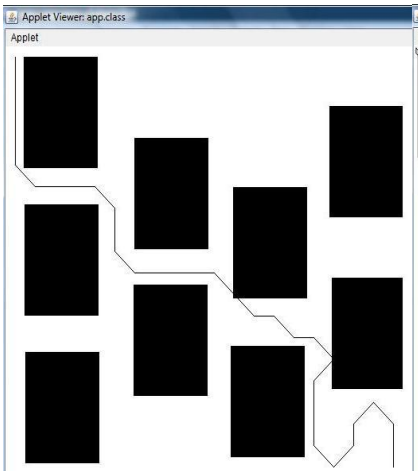


Figure 19(a): The path traced in scalability test at 1st generation

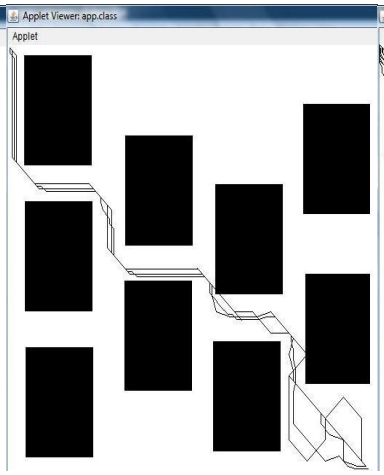


Figure 19(b): The path traced in scalability test at 3rd generation

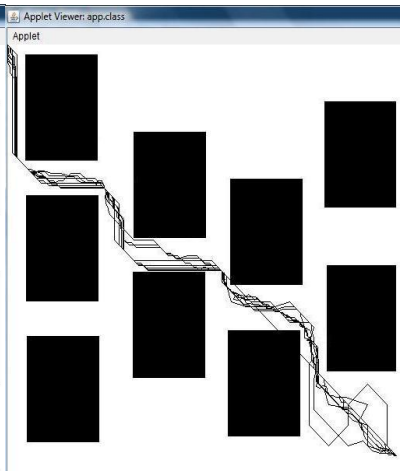


Figure 19(c): The path traced in scalability test at higher generations

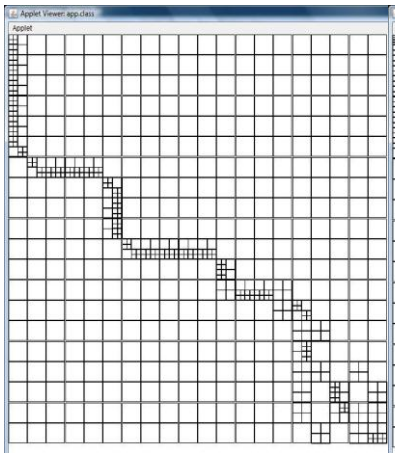


Figure 20(a): The condition of the grid at Generation 2

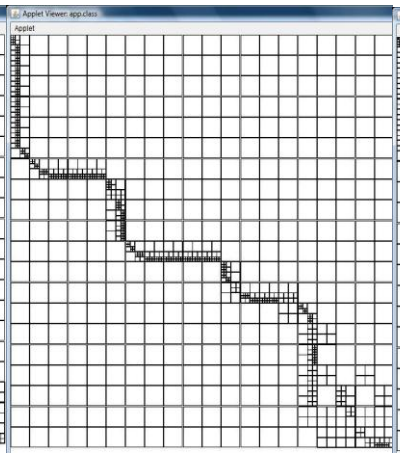


Figure 20(b): The condition of the grid at Generation 4

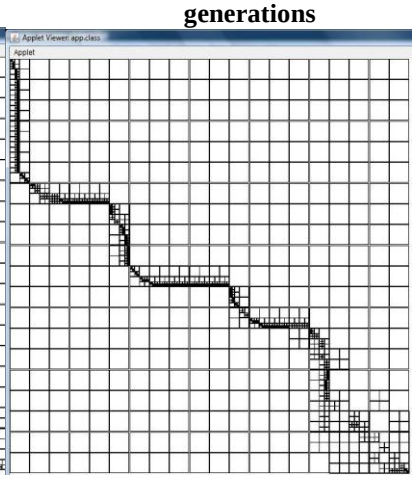


Figure 20(c): The condition of the grid at Higher Generations

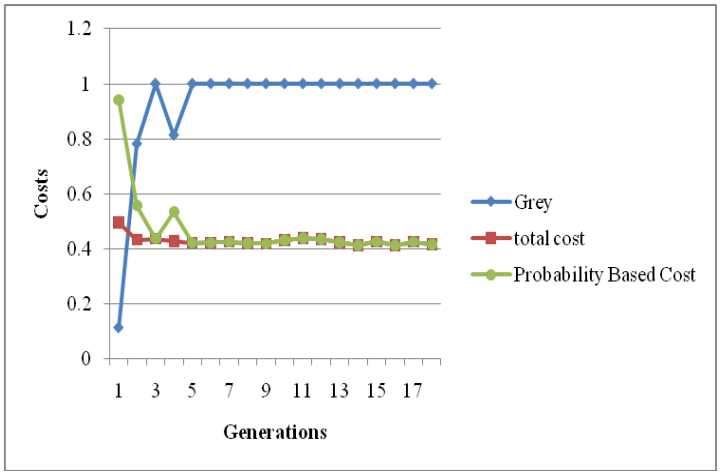


Figure 21: The various costs between generations

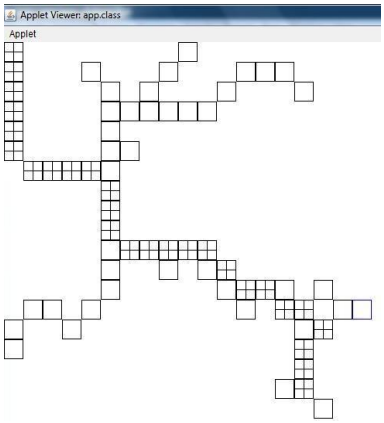


Figure 22(a): The passing of nodes between generation 1st to 2nd generation

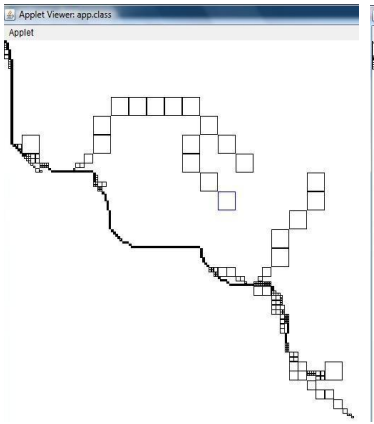


Figure 22(b): The passing of nodes between generation 3rd to 4th generation

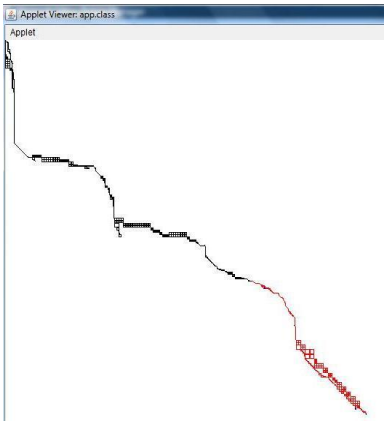


Figure 22(c): The passing of nodes at higher generations

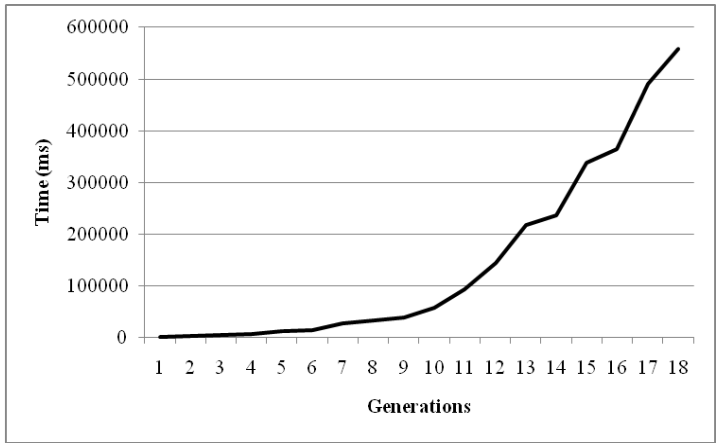


Figure 23: Time required by algorithm v/s generations

6.4 Scatter Test

The last test we applied to the algorithm was the scatter test. Here the obstacles were scattered all over the board. The map has been shown in figure 24. We saw that the robot was easily able to travel from the source to destination in a simple line. The map given was of dimensions 1000X1000 grids, which was initially broken down into chunks of 50X50 grids. The value of α was kept as 5 and β was kept as 10.

The path traced by the robot at various generations is given in figure 24(a), (b) and (c). The condition of the grid are given in figure 25(a), (b) and (c) respectively (only the cells are shown, the grayness is not shown in figure).

The total cost, fitness and grayness of the final path at various generations is given by figure 26. The change in cost is clearly visible across the generations

The cells that are passed from one generation to other are shown in figure 27(a), (b) and (c). Figure 28 shows the time required (in total) between the various generations.

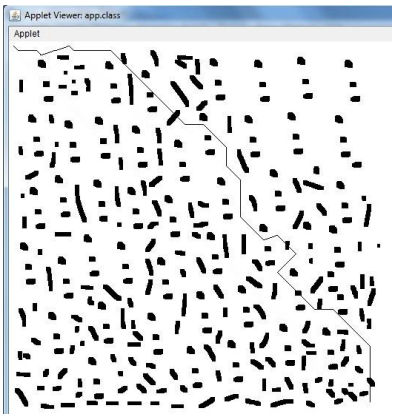


Figure 24(a): The path traced in scatter test at 1st generation

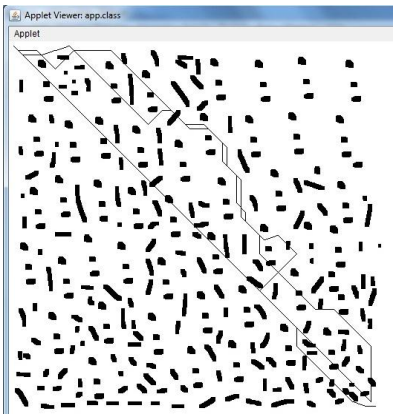


Figure 24(b): The path traced in scatter test at 3rd generation

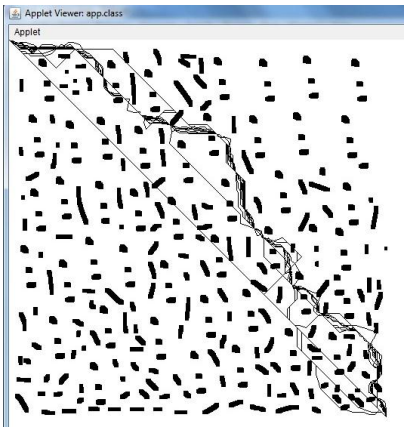


Figure 24(c): The path traced in scatter test at higher generations

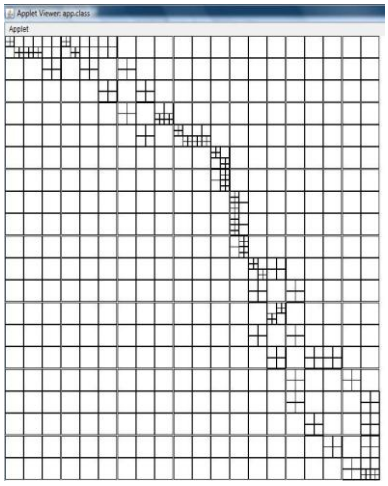


Figure 25(a): The condition of the grid at Generation 2

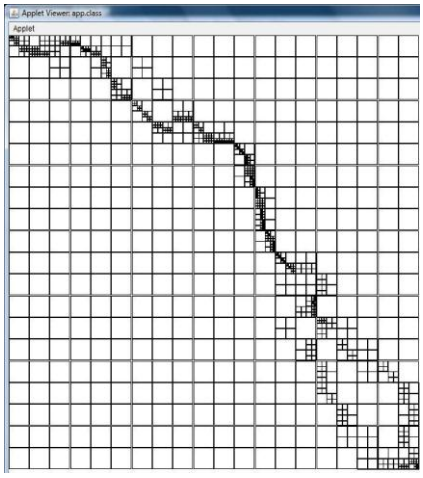


Figure 25(b): The condition of the grid at Generation 4

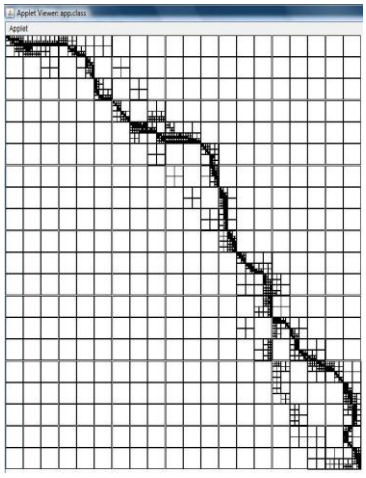


Figure 25(c): The condition of the grid at Higher Generation

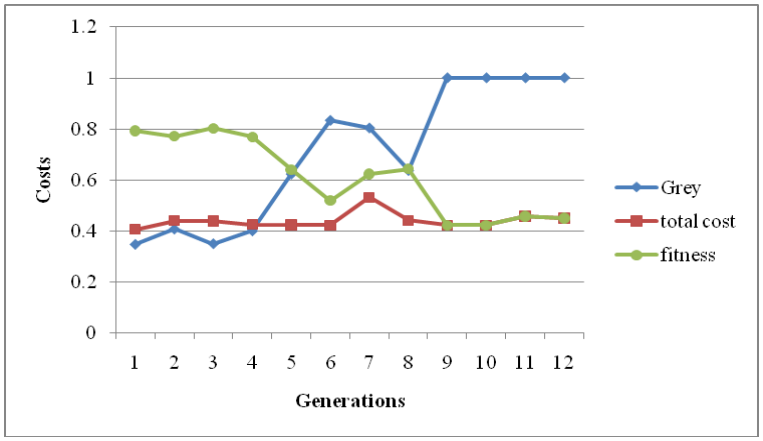


Figure 26: The various costs between generations

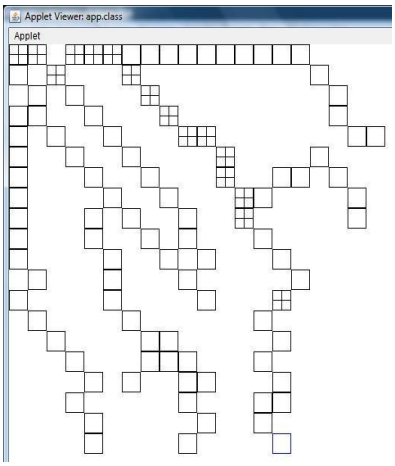


Figure 27(a): The passing of nodes between generation 1st to 2nd generation

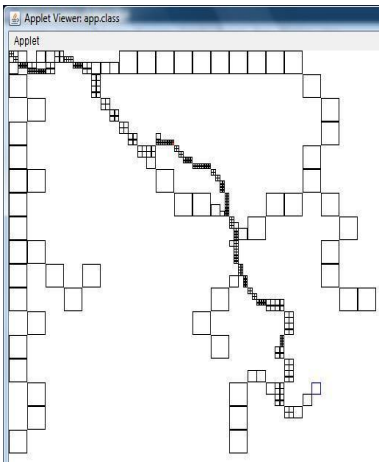


Figure 27(b): The passing of nodes between generation 3rd to 4th generation

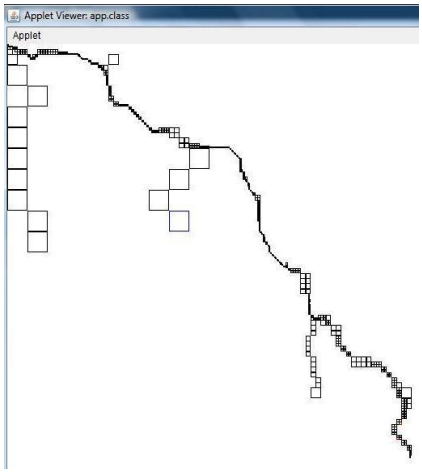


Figure 27(c): The passing of nodes between generation higher generations

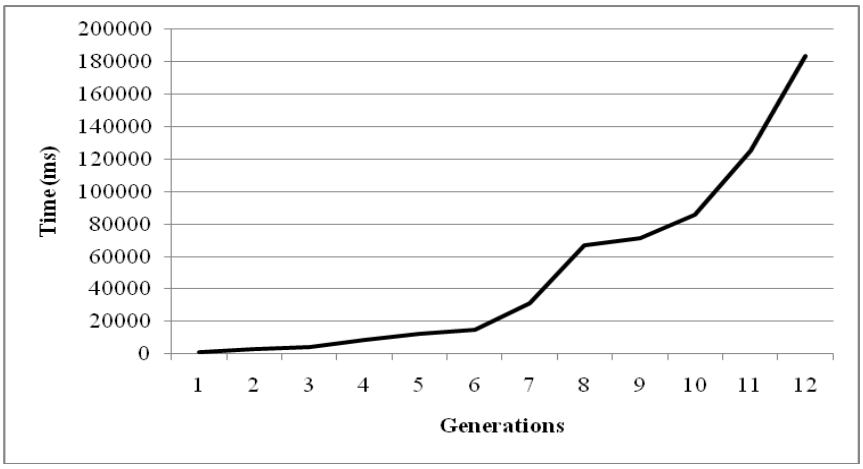


Figure 28: Time required by algorithm v/s generations

6.5 Effect of change in initial grid size

We stated that we start with map of size M x N grids, which is initially broken down into chunks of smaller sizes that make up the initial probabilistic map. In this section we analyze the effect of change in this parameter of the initial grid size (henceforth mentioned as size). The grid in the subsequent runs were decomposed to give finer grids. We tried to see the performance of the algorithm by running the algorithm for various sizes. It is clear that if the grid size becomes unity, the algorithm would get generalized to the conventional MNHS. If however the size becomes the size of the graph, the algorithm would generate highly un-probabilistic paths and would take large generations (not necessarily large time) to get high paths with high degree of certainty.

We applied all the simulations using the third test case as described above. The graph size was 1000X1000. We studied the algorithm in two heads. The first head consisted of the higher order graph sizes. Here we used the sizes 500, 250, 125 and 100 and compared them with each other. The other head consisted of the sizes 50, 25, 10 and 5. The various metrics of the algorithm were compared between various sizes in these two heads.

We first plotted the probability (or grayness) at various times for the paths generated by these algorithms. The results are given in figure 29. The results clearly show that as the time increases, all the paths reach the grayness of 1. However the higher sizes usually take more time to reach the grayness of 1 as compared to the smaller sizes. This is because the uncertainty is very high as the sizes are high. The decomposition in larger sizes takes time till a point comes where the certainty is 1.

We also study the total cost of the algorithm at various algorithmic runs. The results are shown in figure 30. The various sizes at higher times converge to almost the same range. Please note that 30(a) and (b) are plotted against different data scales. No major conclusion can be drawn from this.

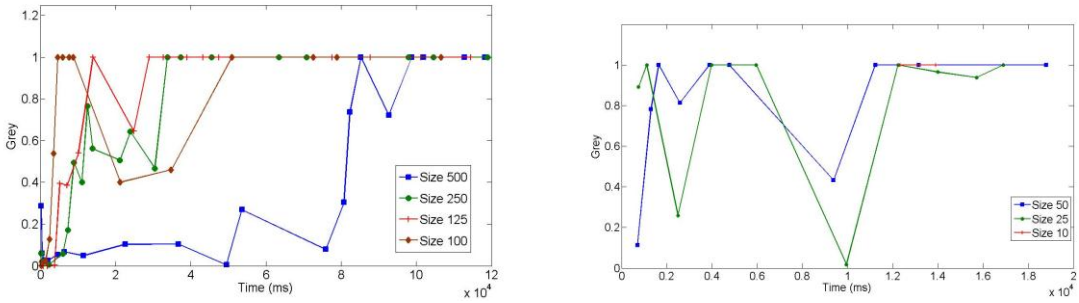


Figure 29: The grayness at various algorithmic runs v/s time for set of (a) higher initial grid sizes (b) lower initial grid sizes

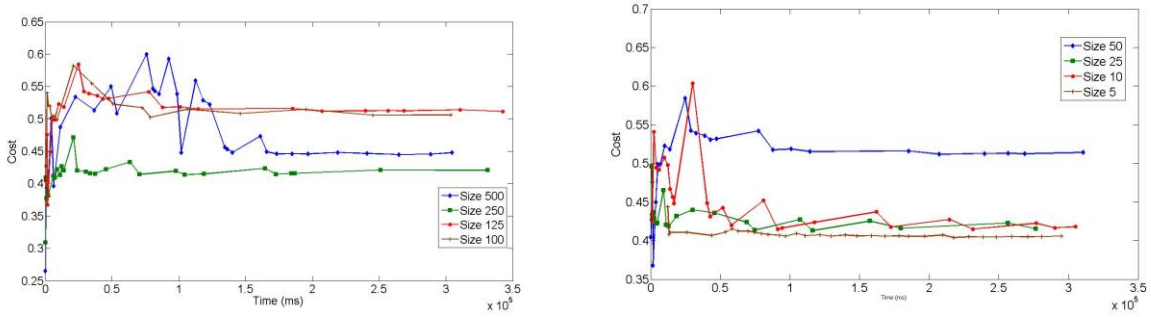


Figure 30: The cost at various algorithmic runs v/s time for (a) higher initial grid size set (b) lower initial grid size set

Based on the readings and analysis of both the presented studies, we may easily analyze the behavior of the probability based fitness of the algorithm at various sizes. The results at various times are given in figure 31.

The various generations that the algorithm completes in a specific amount of time is studied and shown in figure 32. It can be seen that the number of generations increase for smaller sizes.

It was also earlier shown by the authors [31] that the MNHS becomes A* algorithm when α is taken to be 1 it becomes a breadth first search when α is infinity. The same analogy is true in this case as well and the effect of changing α for various graphs can be hence easily visualized.

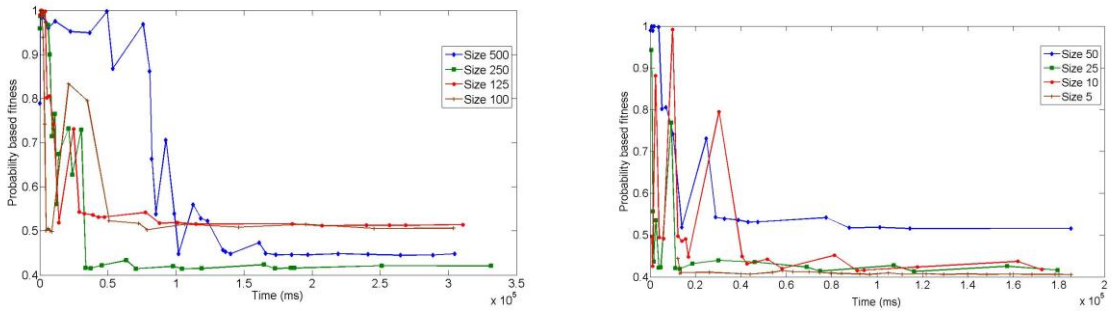


Figure 31: The probability based fitness at various algorithmic runs v/s time for (a) higher initial grid size set (b) lower initial grid size set

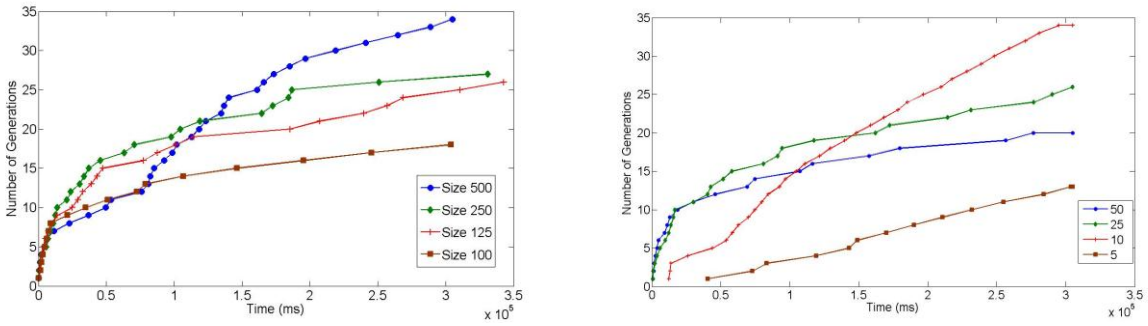


Figure 32: The number of generations at various algorithmic runs v/s time for set of (a) higher initial grid sizes (b) lower initial grid sizes

7 Comparative Analyses

In this section we compare the proposed algorithm with other commonly used approaches in the robotic literature. As most of the algorithms differ in regard to modeling scenarios, we restrict our discussion to theoretical concepts while comparing the advantages and disadvantages of the proposed approach over other algorithms. These may be easily verified by experimentation over the different modeling scenarios that the individual algorithms assume. The first class includes the potential methods where every point on the map is given some potential and accordingly the move is made. These algorithms are known to be computationally very expensive and require a large setup time. The same is the problem with the use of A* algorithm [32]. The proposed algorithm uses the concept of multi-resolution path planning to overcome the computational cost and make planning iterative in nature. If the map resolution is small or time available is infinitely large, the conventional potential or A* approaches would give better performance both in terms of path optimality as well as time. But these algorithms would fail severely in case any of these assumptions does not hold. This holds even in case of models proposed by Pozna et al. who optimized the working of the approach [28]. The second class includes the Fuzzy Logic [30] and Artificial Neural Networks [16]. These are behavioral approaches as they model the robotic behavior to react to obstacles and are used for real time robot path planning. The path generated obeys the non-holonomic constraints. However these approaches have two problems. First, they do not regard optimization of path at all. They make the robot travel as per the path

visibility. Second, they completely fail in complex or maze-like map where there may be multiple ways but only one of them is correct.

One of the advantages of the algorithm was its iterative nature which is an in-built phenomenon in the use of Genetic Algorithms. We hence compare our algorithm to the genetic approaches which have now assumed a sophisticated nature [1, 15, 22, 37]. Besides iterative nature, these approaches have an added advantage of flexibility of path which is not the case with the proposed approach in which the robot can only make a discrete set of moves from every cell [16]. As a result the overall path optimality is better in case of use of these approaches. However, they work only for simpler maps. As the map complexity increases and the maps become more maze-like, these algorithms fail to perform as none of their solutions are feasible. This is due to the complex and massive size of the Genetic search space. This is not the case with our algorithm which can handle very complex graphs as well. This was shown by the scalability test in section 6.

The major work in the paper was into the direction of Multi-Resolution Path Planning. This calls for the comparison of the proposed algorithm over other hierarchical planning algorithms. Kambhampati and Davis used the multi-resolution path planning for solving the same problem in hierarchical mode with the use of A* algorithm [17]. They used two levels of planning on a coarser and a finer scale. Their approach ensured the feasibility of path while working over the coarser stage. Our algorithm, however, does not guarantee the feasibility of the path. It instead maintains backup paths by the use of MNHS algorithm. The additional lookup used by Kambhampati and Davis made their algorithm computationally much expensive as compared to ours. Their algorithm was further not iterative in nature and compromised with path optimality for computational speedup. Unless the main path generated at coarser level (as per their algorithm) or at initial generations (as per our algorithm) meets with infeasibility, our algorithm would be better in terms of time and iterative nature. This would be the case with most of the practical maps, as we saw in section 6. Even if the path is regarded as in-feasible, with no feasible paths in the vicinity, our backup strategy would closely equate their approach with smaller computations.

Hwang et al. further worked over the optimization of the problem and proposed the mesh simplification of the map [12]. Their approach was an optimal representation but suffered from the same drawbacks as was the case with Kambhampati and Davis. Urdylis et al. also used a hierarchical path planning based on a probabilistic map at multi-levels [36]. However their algorithm heuristically selected a level and operated at the same level. This was the final path. It may be easily argued that their algorithm did not have any advantages of iterative nature. The proposed algorithm may be regarded as advancement over their algorithm in terms of iterative nature in two heads. Firstly, the passage of information between levels that ensures the algorithm does not start all over from scratch every time level is change; Secondly, provision of simultaneously having multiple cells of the map at multiple levels.

8 Conclusions

In this paper we had proposed a method to solve the problem of path planning in static environment using a hierarchical approach involving MNHS and probability based fitness. 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 time required, number of generations etc. depended upon the problem being considered.

We first saw the working of the algorithm in a plain environment without any collisions. We saw that the algorithm was able to easily march towards the goal. Even though the algorithm worked correctly, yet the time required was larger in this case as compared to the standard A* algorithm. This was due to the computational overheads. Both the A* as well as our algorithm had marched in a straight line from the source to the destination and hence reached the goal easily. We also saw that our algorithm tried to expand nodes that were not in the most optimal path. This was done to ensure that we have a solution even if the main path ultimately fails. In this case such an approach added computational overheads and hence was unlikely.

However, when we introduced an obstacle, the effect of such an approach was clear. In this run, the algorithm initially generated a solution that was later found to be in the path of the obstacle. As a result the algorithm had to search for a new path that was already expanded to some extent and hence saved expansion time. We also studied the tendency of the algorithm to expand sparse nodes that were distant apart. In this case also, we saw that the algorithm cost converged very quickly as the algorithm was able to find almost the correct path relatively easily.

We again tried to test the power of the algorithm to react to high number of obstacles. This is very practical in nature. We saw that again the algorithm was able to find the correct path in a few generations. Then the process of decomposition went on to further break down the path and give finer paths. Earlier the algorithm had given paths that were prone to collisions. However the algorithm soon found out the path that had no obstacles.

The last test that we applied was of scattered inputs. Here the algorithm was made to run over a large number of distant points. The algorithm in this case also found the correct result. However, unlike the earlier cases, the algorithm took more number of generations to finally come to a path where the probability of collision by obstacle was zero. Also the algorithm in this case had to expand a larger number of paths between obstacles to get the final desired path.

Any practical scenario is a uniform mixture of all the 4 cases that we have considered. Since the algorithm could solve each of the 4 cases, we can be assured that this algorithm would give a timely response in any situation. Another important aspect associated with the algorithm is that it can be terminated at any point of time. We saw that the improvement in path was rapid at the start and then the improvement was slower. Hence, even if we terminate the algorithm after a few iterations, we would still get good solutions that would most probably be collision free and of shortest possible length possible. Hence using this algorithm we were able to make modified A* algorithm that was computationally much less expensive and solved the problems presented. The algorithm hence works better than the classical A* algorithm.

We have studied this algorithm for static problems. It needs to be generalized to dynamic problems as well. Also the relation of the parameters α, β and the initial graph size need to be studied so that the algorithm may be optimized for various kinds of problems. The algorithm again needs to be validated by using a physical robot and the physical paths and obstacles. All this needs to be done in future.

References

[1] 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

[2] 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

[3] Carpin, Stefano & Pagello, Enrico, "An experimental study of distributed robot coordination", *Robotics and Autonomous Systems*, Volume 57 , Issue 2, pp 129-133, February 2009

[4] 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

[5] 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

[6] 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

[7] Ge, Shuzhi Sam & Lewis, Frank L., "Autonomous Mobile Robot", *Taylor and Francis*, 2006

[8] Goel,Ashok K, "Multistrategy Adaptive Path Planning", *IEEE Special Feature*, pp57-65, December 1994

[9] 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

[10] Holland, J.H., "Adaptation in natural and artificial systems", *Ann Arbor: University of Michigan Press*, 1975.

[11] 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

[12] 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

[13] 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

[14] 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
[15] Juidette, H. & Youlal, H., " Fuzzy dynamic path planning using genetic algorithms", *IEEE Electronics Letters*, 17th February 2000 Vol. 36 No. 4, pp 374-376
[16] 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
[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] 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
[19] 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
[20] 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
[21] Lima, Pedro U. & Custodio, Luis M., "Multi-Robot Systems", *Innovations in Robot Mobility and Control*, Springer 2005, pp 1-64
[22] 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
[23] Lozano-Pérez, T. & Wesley, M. A., "An algorithm for planning collision-free paths among polyhedral obstacles", *Communications of the ACM*, pp 560–570, 1979.
[24] 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
[25] 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
[26] 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
[27] 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
[28] 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
[29] 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
[30] 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
[31] Shukla, Anupam & Kala, Rahul; "Multi Neuron Heuristic Search", *International Journal of Computer Science and Network Security*, Vol. 8, No. 6, pp 344-350, June 2008
[32] 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
[33] 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.
[34] Thrun, Sebastian, "Learning metric-topological maps for indoor mobile robot navigation", *Artificial Intelligence*, Volume 99, Issue 1, February 1998, Pages 21-71
[35] 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

[36] 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
[37] 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
[38] 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