Robotic Path Planning using Multi Neuron Heuristic Search

Rahul Kala
MTech
Department of Information Technology, Indian Institute of Information Technology and Management Gwalior, Gwalior, India
Ph: +91-9993746487
rahulkalaiiitm@yahoo.co.in

Anupam Shukla
Associate Professor
Department of Information Technology, Indian Institute of Information Technology and Management Gwalior, Gwalior, India
Ph: +91-751-2449811
dranupamshukla@gmail.com

Ritu Tiwari
Assistant Professor
Department of Information Technology, Indian Institute of Information Technology and Management Gwalior, Gwalior, India
Ph: +91-751-2449822
rt_twr@yahoo.co.in

Citation: R. Kala, A. Shukla, R. Tiwari (2009), Robotic Path Planning using Multi Neuron Heuristic Search, *Proceedings of the ACM 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, Seoul, Korea, pp 1318-1323.
Final Version Available At: http://dl.acm.org/citation.cfm?id=1656167

ABSTRACT
Robotics is a highly multi-disciplinary field which attracts the attention of many researchers from diverse fields. One of the major studied problems in Robotics is the problem of Robotic Path Planning. The problem deals with finding of the path that can be used by the robot for navigation purpose without any collision. The output of this algorithm is then implemented for physically moving the real robot on the desired path. In this paper we have used Multi-Neuron Heuristic Search (MNHS) which is an advanced form of A* algorithm. The MNHS was earlier proposed by the authors for special cases where the heuristics changes sharply and it was shown to be a powerful algorithm in the same context. In this paper we apply the MNHS for the Robot Path Planning. The motivation is to make the problem robust against the uncertainties that might arise like the sudden discovery that the path being followed does not lead to the goal. The MNHS has better capabilities to solve maze-like maps where the uncertainty is extremely high. Another such area is when the robot enters into a highly chaotic area. Here it might be better to go with a path that is less chaotic or has lesser number of obstacles. We tested the algorithm for numerous test cases. In all the cases, the MNHS was able to solve the problem of path planning well.

Categories and Subject Descriptors
I.2.9 [**Robotics**]: Autonomous Vehicles, Workcell organization and planning.

General Terms
Algorithms

Keywords
Robotic Path Planning, Multi-Neuron Heuristic Search, Robotics, Robotic Navigation, Robotic Planning.

1. INTRODUCTION
Robotic Path Planning is one of the problems in the field of robotics that tries to find and optimize the path from the initial position to the final position. [1]. Besides optimization, it needs to be ensured that the robot moves without any collision in the entire path it follows from the source to the destination. This would mean that the algorithm avoids all obstacles and reaches the destination starting from the source in the least time possible. This is also referred to as the navigation plan of the robot. The problem is usually studied in two separate heads. These are path planning under static environment and path planning under dynamic environment. In static environment the condition of the robotic map is constant and does not change with respect to time due to absence of the moving obstacles. In dynamic environment path planning however, the map keeps changing with the passage of time. This is due to the presence of dynamic obstacles like other robots, vehicles, etc.

Path planning is one of the numerous algorithms used in the problem of robotics. The whole problem of robotics or intelligent robotics involves the simultaneous contribution of people from varied backgrounds and disciples. The sensors and sensor data, the building up of the map, the communication between robots or between robots and machine, visual processing, multi-robot coordination are some of the major fields in robotics that require participation from different people.

The paper proposes the use of Multi Neuron Heuristic Search (MNHS) for the problem. The algorithm is used to find out the most optimal path of the robot [2]. This path is the final path that is used for the purpose of robotic navigation. The algorithm returns the complete path if one exists between the source and the destination. If however, no path is possible between the source and the destination, the algorithm returns null.

The robot may be easily made to follow the path that the algorithm runs by any robotic controller. The controllers try to guide the robot in a step by step manner so that it follows the desired path and traverses the same in the lease possible time.

The elementary model of cognition [5] includes three main cycles. Among these, the 'sensing-action' cycle is most common for mobile robots. This cycle inputs the location of the obstacles and subsequently generates the control commands for the motors to set them in motion. The second cycle passes through perception and planning states of cognition, while the third includes all possible states including sensing, acquisition, perception, planning and action [6]. Sensing here is done by ultrasonic sensors/camera or by both. There are many algorithms for construction of the robot's world map [7]. The term Planning of Navigation [8] refers to the generation of sequences of action in order to reach a given goal state from a predefined starting state.

The MNHS [21] is an advanced form of A* algorithm that was earlier proposed by the authors. The A* algorithm does not give good results in the absence of good heuristics. If the choice of heuristics is bad, then the algorithm would normally not perform well or would take a lot of time. The performance of the A* algorithm to a large extent is dependent on the heuristics.

It was also earlier shown by the authors that the A* algorithm gives good results when used in the problem of robotic path planning [22]. It gave the best results or the shortest results possible. These were found to be better than those obtained from the ANN or Evolutionary Algorithms [23]. However, the A* algorithm is known to be computationally expensive.

Suppose that A* algorithm is used for the solution of the problem on a maze-like situation. It would expand a lot many nodes, only to find that the entire region did not generate any path. Then it would start exploring the other regions which initially looked would provide bad solutions. This would waste a lot of time.

The motivation behind the use of MNHS is to keep backup paths ready and explore them also from time to time, so that if some region completely fails to give a correct solution, the other paths are already explored to a good extent. If the 2nd backup path also fails, then the 3rd backup path is also explored to a fair extent. This is possible due to the inherent nature of the MNHS that equally respects the various heuristic values from bad to good and expands all of them. This is because it is possible that the bad heuristics may suddenly turn good and vice versa.

The paper is organized as follows. In section 2 we discuss the motivation behind the problem. Section 3 talks about the MNHS algorithm. Section 4 deals with the application of MNHS in the problem of robotic path planning. In section 5 we discuss the results. Section 6 gives the conclusion.

2. RELATED WORKS

The problem of robot navigation control, due to its applicability, is of a great interest. We have already seen good research in various modules. A lot of work exists to model the entire problem [1 - 7]. There exist good algorithms to scan the environment and represent all the obstacles in form of a grid [3]. Also various algorithms have been proposed to plan the movement of the robot using various conditions.

The whole problem till now has been seen under separate heads of planning navigation control of static environment and planning navigation control of dynamic environment. If we come to static environment, many algorithms have been implemented and results verified [8, 10, 11, 19]. In planning dynamic environment the steps are a little different, as the environment continuously changes.

We also have various works of research in which people have tried to solve the navigation problem using genetic algorithm [8, 10, 11, 16]. The basic principles in all these have been to take a fixed solution length and find the solutions by using genetic operators. Also similar work exists in neural network [6, 11, 15]. Here neural network has been applied mainly on static data.

In this paper we have modified the A* algorithm to better handle uncertainness that happen to be natural whenever we work in practical scenarios. The work hence is advancement over the previous works that would give poor results in the presence of highly uncertain paths.

3. MULTI NEURON HEURISTIC SEARCH

This algorithm can be taken as a betterment over the A* algorithm where the heuristic function exists, but is bound to change suddenly. The heuristic and A* approach use the heuristic function in order to get the search closer and closer to the goal, but when it changes suddenly, the strategy is destroyed. Hence these algorithms suffer. A solution may be not to use the heuristics at all. But if the heuristic function is available, it is always better to use it rather than not to use it altogether as was the case with other algorithms.

The algorithm can be applied to the cases where he following problems occurs in heuristic function:

- The heuristic function keeps improving. As we reach near goal, it 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. Suppose the heuristic function of any point (x,y) on the maze denotes its distance from the goal. We can see that 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. The solution is to use the new algorithm which respects all the good, bad and moderate values of heuristics, so that no value suffers. Such an algorithm, due to its parallel nature will take huge benefits from modern concepts like multi-processor, grid computing etc.

The basic idea of this algorithm is the use of many neurons working one after the other. Each of these take care of high to low values of the heuristic functions. 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, some at low.

In all we take α neurons. We have a list of heuristic costs each corresponding to node seen but waiting to be processed. We divide the cost range into α ranges equally among them. 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 algorithm MNHS is given by the following pseudo code.

MNHS(source, goal)

open ← empty priority queue
closed ← empty list
add a node n in open such that position(n) = source, previous(n) = null and f(n), g(n), h(n) are as calculated by respective formulas with priority f(n)
while open is not empty
extract the node n_1, n_2, n_3, n_4….. n_α from open with the priority of n1 as highest and the others equally distributed between other α-1 nodes.
if n_i = goal for i=1,2,3,4,5…..α then break
else
nodes ← nodes from the expanding of node ni
for each node m in nodes
if m is already in open list and is equally good or better then discard this move
if m is already in closed list and is equally good or better then discard this move
delete m from open and closed lists
make m as new node with parent n
calculate f(m), h(m), g(m)
Add node m to open with priority f(m)
Add n to closed
Remove n from open

Here g(n) denotes the historical cost function
h(n) denotes the heuristic cost
f(n) is the cost of the node (=f(n)+g(n))

The algorithm is similar to the A* algorithm. The only difference is that at each iteration, we take and process α nodes from the open list one after the other.

Consider the problem of solving a maze. The problem is that we have to move from the initial position to the final position in the maze without colliding from walls.

Refer Figure 1(a) for the problem input. Here 0 represents the region we cannot move (wall) and 1 represents the region we can move (path). Top left is the start point. Bottom right is the finish point. The heuristic function is taken as the square of the distance of the current point to the final point. The solution generated by the MNHS is given in Figure 1(b). The numbers in results show the order in which they were discovered. The number of bottom right corner is the number of nodes explored as shown in Figure 1(b).

4. MNHS IN 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 numerous 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 these algorithms is quite limited in nature. In the presence of complex maps they can hence cause problems. An example would be the maze like structure where a robot has to find its way out of the maze.

1	1	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0	1
1	1	1	1	1	1	1	1	0	1
1	0	0	0	0	0	0	1	0	1
1	1	1	1	1	1	0	1	0	1
1	0	0	0	0	1	0	1	0	1
1	1	1	1	0	1	0	1	0	1
1	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1

Figure 1(a): The maze solving problem input

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 1(b): The solution generated by MNHS

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 the rapid change in heuristics. This is what would come to rescue if by chance the robot reached quite near to the goal only to find that there is no way to reach it.

This concept is shown in figure 2. 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 this problem the historical cost is taken as the distance from the source to the current position. The heuristic cost is the distance of the current position to the goal position. The total cost is the sum of both the costs.

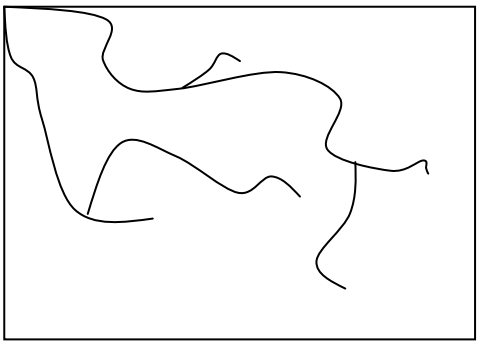

Figure 2: The concept of MNHS in path planning

The problem of path planning deals with the determination of a path which navigates the robot in such a way that no collision occurs. In order to solve the problem we assume that the input is already available in form of a map. The map is a representation of the robotic world that tells where the obstacles are found and which are the traversable regions. Here we assume that the map is available in form of grid of size MXN. Each of the cell of this grid contains 0 or 1. A 0 in such a grid signifies that the region has an obstacle present. Similarly a 1 signifies that the region is traversable and may be used for the purpose of travelling. The obstacles may span across multiple cells.

It is further assumed that the grid given as input is of considerable size. If the grid exceeds a certain threshold of size, it would become computationally impossible for the algorithm to find a result. Hence, we restrict the size of the map according to the computational capability and time constraints in whatever real life specific problem is being considered.

The algorithm would generate as its output a path that can be used by the robot for the navigation purposes. The path may be traversed using any robotic controller. This is for the execution of the steps given by the planning algorithm.

There are 2 major issues in the use of MNHS in the problem of path planning. These are the problem representation in the form of a graph, fixing the value of parameter α. We discuss both of them one by one.

4.1 Graphical Representation of Problem

The MNHS is a graph searching algorithm. It is hence necessary to first formulate the problem in the form of a graph. Every grid of the map corresponds to a vertex of the graph. A graph vertex may hence be denoted by (i,j) where i and j refer to the position of the vertex with respect to the 2 coordinate axis. A vertex exists in the graph only if it is accessible. In other words if some position is occupied by some obstacle in the robot map, it is not stated as a valid graphical vertex.

The edges in this graph are in the form of valid robot moves. An edge exists between two vertices V_1 and V_2 only if the robot can make a direct move between the two vertices. The weight of the vertex is the total length of the path in between V_1 and V_2.

Suppose that a robot is presently located at location V(i,j). In our model the robot can only move to the vertices A(i-1,j), B(i-1,j+1), C(i,j+1), D(i+1,j+1), E(i+1,j), F(i+1,j-1), G(i,j-1), H(i-1,j-1), I(i-1,j+2), J(i+1,j+2), K(i+2,j+1), L(i+2,j-1), M(i+1,j-2), N(i-1,j-2), O(i-2,j-1) and P(i-2,j+1) as shown in Figure 3. The weights of the edges are 1, √2 or √5. It may be noted that other vertices in the figure are not connected directly as they may easily be visualized as a combination of 2 or more moves totaling to same weight.

	I (i-1,j+2)		J (i+1,j+2)	
P (i-2,j+1)	B (i-1,j+1)	C (i,j+1)	D (i+1,j+1)	K (i+2,j+1)
	A (i-1,j)	V (i,j)	E (i+1,j)	
O (i-2,j-1)	H (i-1,j-1)	G (i,j-1)	F (i+1,j-1)	L (i+2,j-1)
	N (i-1,j-2)		M (i+1,j-2)	

Figure 3: The edges between the vertices in problem graph

Hence any vertex can be connected to a maximum of 8 vertices around it, subjected to its existence and presence of obstacles. Here it may also be observed that by converting the problem into this map we have made the robotic moves discrete. Now the robot has a limited number of choices of basic moves that it may make. These choices play a big role in deciding the time and memory complexity of the MNHS. A higher number of choices may mean a larger complexity. For the same reason we cannot allow the robot to make all kinds of moves in the continuous domain.

4.2 MNHS Uncertainty Parameter α

The MNHS has an extra parameter that helps us control the algorithm. It was earlier shown by the authors that if α is 1, the algorithm behaves like a conventional A* algorithm and if α is infinite, the algorithm behaves like the Breadth First Search [21].

In the problem of path planning this factor should be fixed according to the uncertainty of the map. If the heuristics may fluctuate very rapidly, this factor must be high. This would be the situation in case of a maze-like map with large number of complex obstacles. In most simple graphs, this factor may be kept low.

5. RESULTS

In order to test the algorithm, we developed a simulation engine of our own. The engine was made keeping in mind the practical applicability of the algorithm on the robot. The simulation engine took as input the map. This was given in the form of an image. The algorithm then executed the algorithm to compute the path. The path was shown using JAVA Applets.

We applied various tests to the algorithm in order to ensure that the algorithm behaves well in each and every condition. These tests are discussed in the next sections.

5.1 Case I: No Obstacle Condition

Initially we did not place any obstacle in the path from the source to destination. The algorithm was made to run on a completely blank map. We observed that the algorithm traced the path from the source to the destination following a straight line path. This was the shortest path possible. The results of the algorithm are shown in Figure 4(a). The robot was supposed to move from the top left corner to the bottom left corner. The size of the grid was 100X100. The value of α was fixed to be 5.

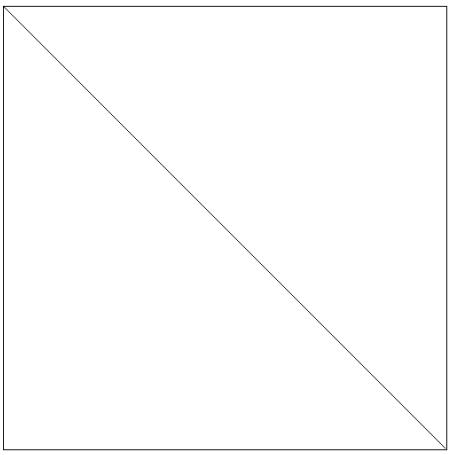

Figure 4(a): The result of MNHS for Case I

5.2 Case II: Single Obstacle

The second case we considered was of a single obstacle in the path from the source to destination. The robot easily avoided the obstacle and marched towards the goal position. This was also the shortest path possible. The results of the algorithm are shown in Figure 4(b). The robot was supposed to move from the top left corner to the bottom left corner. The size of the grid was 100X100. The value of α was fixed to be 5.

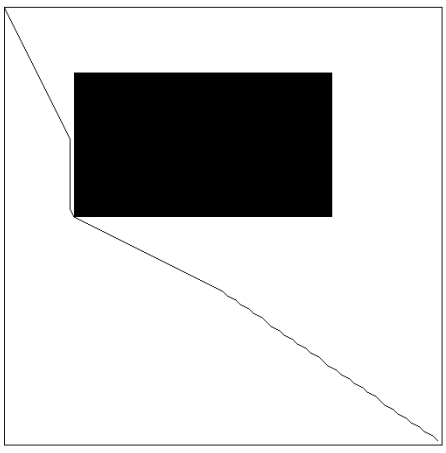

Figure 4(b): The result of MNHS for Case II

5.3 Case III: Complex Obstacles

The last case we presented before the algorithm was to test its ability to handle complex inputs. Various complex obstacles were placed in the path of the robot from the source to destination. The robot again easily avoided the obstacle and marched towards the goal position. This was also the shortest path possible. The results of the algorithm are shown in Figure 4(c) and 4(d). The robot was supposed to move from the top left corner to the bottom left corner. The size of the grid was 100X100. The value of α was fixed to be 5.

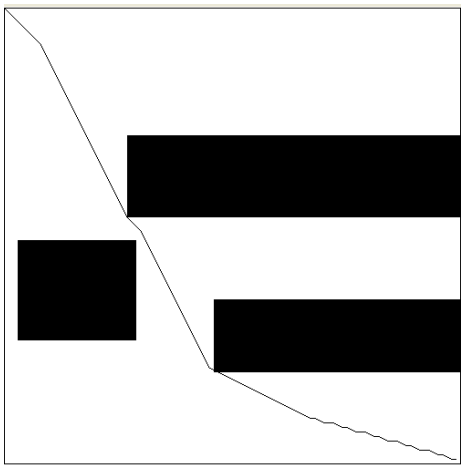

Figure 4(c): The result of MNHS for Case III, 1st map

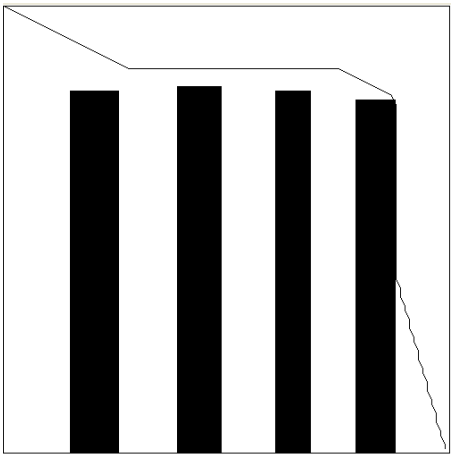

Figure 4(d): The result of MNHS for Case III, 2nd map

6. CONCLUSIONS

In this paper we proposed the use of MNHS to solve the problem of robotic path planning. We saw that we were able to solve the problem in almost all given scenarios well in time. The MNHS proved to be a great algorithm for the purpose.

The algorithm further needs to be used in practical life scenarios which are more complex than the cases presented here. Also the value of α was kept constant in the cases presented. The determination and setting of the most optimal value of α needs to be studied in the future.

7. REFERENCES

[1] Hutchinson, S. A. and Kak, A. C., "Planning sensing strategies in a robot work cell with Multi-sensor capabilities," IEEE Trans. On Robotics and Automation, vol.5, no.6, 1989.

[2] Rich, E. and Knight, K., Artificial Intelligence, McGraw-Hill, New York, pp. 29-98, 1991.

[3] Takahashi, O. and Schilling, R. J., "Motion planning in a plane using generalized voronoi diagrams," IEEE Trans. on Robotics and Automation, vol.5, no.2, 1989.

[4] Borenstain, J., Everett, H. R., and Feng, L., Navigating "Mobile Robots: Systems and Techniques", A. K. Peters, Wellesley, 1996

[5] Matlin, W. Margaret, Cognition, Hault Sounders, printed and circulated by Prism books, India, 1996.

[6] Konar, A. and Pal, S., "Modeling cognition with fuzzy neural nets" In Fuzzy Systems Theory: Techniques and Applications, Leondes, C. T., Ed., Academic Press, New York, 1999.

[7] Pagac, D., Nebot, E. M. and Durrant. W., H., "An evidential approach to map building for autonomous robots," IEEE Trans. On Robotics and Automation, vol.14, no.2, pp. 623-629, Aug. 1998.

[8] V. Ayala-Ramirez, A. Perez-Garcia, E J. Montecillo-Puente, R.E. Sanchez-Yanez, "Path planning using genetic algorithms for mini-robotic tasks", 2004 IEEE International Conference on Systems, Man and Cybernetics, Vol 4, pp 3746- 3750

[9] H Fkezza-Buet, F Alexandre "Modeling prefrontal functions for robot navigation", IEEE International Joint Conference on Neural Networks, 1999. IJCNN '99, vol 1, pp 252-257

[10] Theodore W. Manikas, Kaveh Ashenayi, and Roger L. Wainwright, "Genetic Algorithms for Autonomous Robot Navigation", IEEE Instrumentation & Measurement Magazine December 2007

[11] Du Xin, Chen Hua-hua, Gu Wei-kang, "Neural network and genetic algorithm based global path planning in a static environment", Journal of Zhejiang University Science, 2005 Vol. 6A No. 6, pp 549-554

[12] Zhang Huan-cheng, Zhu Miao-liang, "Self-organized architecture for outdoor mobile robot navigation", Journal of Zhejiang University Science, 2005 Vol. 6A No. 6 p.583-590

[13] Peter Corke, Ron Peterson, Daniela Rus, "Networked Robots: Flying Robot Navigation using a Sensor Net", April 18, 2003

[14] Cory Quammen, "Evolutionary learning in mobile robot navigation", The ACM Student Magazine

[15] Yong-Kyun Na & Se-Young Oh, "Hybrid Control for Autonomous Mobile Robot Navigation Using Neural Network Based Behavior Modules and Environment Classification", Autonomous Robots, Vol 15, Issue 2, pp 193-206

[16] Seyyed Ehsan Mahmoudi, Ali Akhavan Bitaghsir, Behjat Forouzandeh and Ali Reza Marandi, "A New Genetic Method for Mobile Robot Navigation", 10th IEEE International Conference on Methods and Models in Automation and Robotics, 30 August - 2 September 2004, Miedzyzdroje, Poland

[17] Torvald Ersson and Xiaoming Hu, "Path Planning and Navigation of Mobile Robots in Unknown Environments", CiteSeerX - Scientific Literature Digital Library and Search Engine, doi=10.1.1.111.1668, 2008

[18] László Kiss, Annamária R. Várkonyi-Kóczy, "A Universal Vision-based Navigation System for Autonomous Indoor Robots"

[19] Sven Behnke, "Local Multiresolution Path Planning", Preliminary version in Proc. of 7th RoboCup Int. Symposium, Padua, Italy, 2003

[20] S. Veera Ragavan, and V. Ganapathy, "A Unified Framework for a Robust Conflict-Free Robot Navigation", Proceedings of World Academy of Science, Engineering and Technology, Volume 21 January 2007 ISSN 1307-6884

[21] 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

[22] 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

[23] 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, pp 705-713