Fusion of Evolutionary Algorithms and Multi-Neuron Heuristic Search for Robotic Path Planning

Rahul Kala
Department of Information Technology
Indian Institute of Information Technology and Management Gwalior
Gwalior, Madhya Pradesh, India
rahulkalaiiitm@yahoo.co.in

Anupam Shukla
Department of Information Technology
Indian Institute of Information Technology and Management Gwalior
Gwalior, Madhya Pradesh, India
dranupamshukla@gmail.com

Ritu Tiwari
Department of Information Technology
Indian Institute of Information Technology and Management Gwalior
Gwalior, Madhya Pradesh, India
rt_twr@yahoo.co.in

Abstract—**The problem of path planning deserves a special mention in the field of robotics as it enables the intelligent systems used in autonomous robots to move the robot from one position to the other. Out of the various methods used for solving the problem of robot path planning, two of the common approaches include Multi-Neuron Heuristic Search (MNHS) algorithm and Evolutionary Algorithms (EA). The MNHS algorithm is an algorithm proposed earlier by the authors for solving uncertain search problems. The algorithm is slow but gives better optimal paths. On the other hand the EA gives results in finite time, but the optimality cannot be guaranteed. In this paper we propose to mix these two techniques to get the added benefits of both these algorithms. The MNHS improves the performance of the algorithm while the EA does the task of time optimization especially in case of complex graphs. The EA carries forward the task of selection of points in the robotic map. These points are checked for feasibility and then converted into a traversable graph. The same is used by MNHS to find the most optimal path from source to destination. In this way the algorithm finds out the best path without robotic collision.**

Keywords-robotic path planning; multi-neuron heuristic search; evolutionary algorithms; autonomous robotics; intelligent systems

I. 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 the 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 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, and multi-robot coordination are some of the major tasks in robotics that require participation from different people.

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 in the least 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.

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 done as it is possible that the bad heuristics may suddenly turn good and vice versa.

As the entire algorithm can be time consuming, there needs to be some mechanism to optimize the time. This is done with the help of evolutionary algorithms. Rather than giving the entire map to the MNHS for the purpose of path planning, only a small set of points chosen by the EA are given to the MNHS for the task of finding the smallest route. This greatly limits the search space of the MNHS and results in a big computational optimization. The location of the various points may be optimized by the EA as the algorithm runs and proceeds for convergence.

II. Related Research

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 until 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].

Also similar work exists in neural network [6, 11, 15]. Here neural network has been applied mainly on static data.

III. Multi Neuron Heuristic Search

The Multi-Neuron Heuristic Search (MNHS) algorithm can be taken as an improvement over the A* algorithm where the heuristic function exists, but is bound to change suddenly. The A* algorithm uses heuristic function in order to get the search closer and closer to the goal. But when heuristics change 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 is the case with other non-heuristic algorithms like breadth-first search.

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

- 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 squared 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 MNHS which 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 takes 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. 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. We have a list of heuristic costs each corresponding to node seen but waiting to be processed. We divide the cost range into equal α ranges. 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.

A. 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
Step 5:     extract the node $n_1$, $n_2$, $n_3$, $n_4$….. $n_\alpha$ 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
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)
Step 16:        Add n to closed
Step 17:        Remove n from open
```

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.

B. 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 is ready to provide a backup. The obstacles have not been shown in the figure.

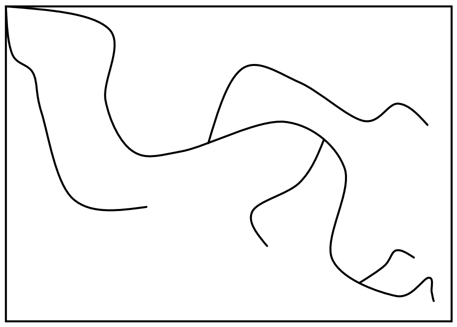

Figure 2: The MNHS path exploration

For this problem the historical cost is taken as the distance traversed from the source to the current position. The heuristic cost is the physical distance of the current position to the goal position. The total cost is the sum of both the costs.

IV. Evolutionary Algorithms

The primary work of EA is time optimization of the entire algorithm. It is known that the A* algorithm may be very time consuming when practically applied to complex maps [23]. The runtime of the A* algorithm depends a lot upon its search space. This is the number of nodes given to the algorithm. The MNHS being a modified version of A* follows similar trends where the total number of points in the map or the number of grids affect the complexity of the algorithm. The EA controls the algorithm optimality by limiting the number of points in the graph that the MNHS has to consider while optimizing the total path length. There are three major parts of this algorithm. Each of which is discussed in the subsequent sub-sections.

A. Individual Representation

One of the foremost tasks in this algorithm is a good individual representation. They may be scattered throughout the robotic map at various locations. Here we represent an individual by a collection of points (P_i) on the robotic map. We place a restriction here that the individual size is fixed to β. This means that the individual can have a maximum of β points in its collection. The complete individual hence becomes <P_0, P_1, P_2, P_3, …. P_β, $P_{\beta+1}$>. Here P_0 is the source and $P_{\beta+1}$ is the goal. This collection is given to the MNHS to work over the most optimal path out of this collection of points.

Each point is a collection of x and y coordinates and may be denoted by (x_i, y_i). The x axis that we take for this problem is the straight line joining the source and the goal. The y axis is perpendicular to the x-axis as given in Figure 3.

Let us suppose that the map is represented in the coordinate system X'-Y' given in Figure 3. Now any point needs to lie within the range of (0',0') to (m',n') so as to lie within the map. Here ' represents the use of X'-Y' coordinate system. This range needs to be converted into equivalent range in the X-Y coordinate system of the individual to generate valid points in the robotic map. This is done by a rotation of angle 'a' in the clockwise direction, where 'a' is the angle between the two coordinate systems given in Figure 3.

Another important characteristic of the individual representation is that the various points in the map are always sorted along the X axis. The final path is the path traversed by touching the various points in a straight line one after the other. Here the source is the first point and goal is the last point. Hence we assume that in the final path, the robot cannot move backwards.

The entire length of the chromosome is hence fixed to a maximum value of 2 x β. If β is set to 10 and each P_i denotes the point (x_i,y_i) then the genetic individual of this path would be represented as <x_1 y_1 x_2 y_2 x_3 y_3 x_4 y_4 x_5 y_5 x_6 y_6 x_7 y_7 x_8 y_8 x_9 y_9 x_{10} y_{10} >.

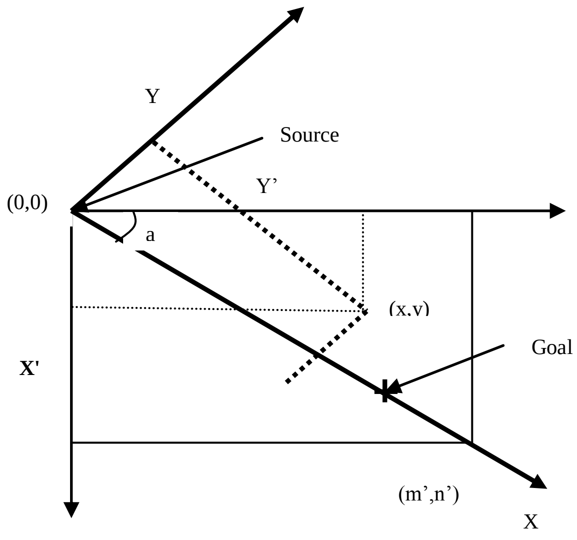

Figure 3: The coordinate system for individual representation and robotic map

B. Conversion to a Graph

So far we have a collection of β+1 points in the map. But the MNHS works on a graph. Hence we need to convert this collection of points into a graph so that the MNHS can work over the most optimal distance over this pool of points.

The graph that we give to the MNHS has the collection of β+1 points as the vertices. An edge exists between any vertex i to any vertex j if the robot can travel from i to j without any collision with obstacles. The weight of the edge is taken to be the physical distance between the vertices i and j. One of the major problems comes in the determination of feasibility of the path. To determine the feasibility of the path between any two vertices, we travel between the two vertices and check for the presence of obstacles. If an obstacle exists, the path is considered as infeasible and the traversal stops.

The adjacency matrix representation of graph has been used. This is a matrix whose every element a_{ij} denotes the existence of an edge between the vertex i and j. The element stores the weight of the weight if an edge exists, else it stores infinity. Sine this is an undirected graph, hence $a_{ij} = a_{ji}$. Also the diagonal elements are all kept as zeros.

This graph is given to the MNHS in the fitness function of the EA. The first node is specified as the source and the last one is specified as the destination of the EA.

C. Evolutionary Operators

The EA uses various operators to carry out the task of optimization. It uses a rank based fitness method and a stochastic selection technique. The two major operators used are crossover and mutation. Both these operators have been adapted as per the problem. Crossover is point based where the new individual gets half the points in the form of (x,y) from the first parent and the other half from the second parent. Scattered crossover is used for this purpose. Similarly mutation is point based where the mutation operator physically moves points represented by individuals on the map where the magnitude of movement depends upon the mutation rate. Elite is another operator used to pass on the best few individuals from one generation to the other. The explanation of the various operators can be found in [24].

V. Relation Between EA and MNHS

We have already stated that the EA does path optimality and the MNHS carries forward the task of path optimality. In this section we explicitly state this relation that also speaks about the algorithm optimality as compared to the other simple and hybrid approaches.

The MNHS algorithm fails to work in the presence of large number of nodes due to the large time complexity. Since it is not an iterative algorithm, we cannot break its run to get the path. This makes it impossible to use MNHS or other heuristic algorithms in most real life scenarios. As a result we need an algorithm that can reduce the dimensionality and select the best nodes for the MNHS to perform in finite time. The MNHS can easily work over a reduced set of points to give the most optimal paths.

The EA is an iterative manner of solving problems. One of the main disadvantages of the algorithm is that it only tries to generate paths based on the fitness of the paths of the previous generations. This makes the algorithm make slow convergence in path optimality. The solutions come early but are not optimal. Hence this algorithm needs the assistance of some heuristics that enable it to form good paths and figure out good and bad points that make up a path.

Hence the combination of both these algorithms solves the twin problem of path optimality and time. It may be easily seen from the algorithm that the mutual contribution of the two algorithms is controlled by the factor β. If β is very small, the resultant algorithm would be dominated by EA. The MNHS would have very little choice between the nodes selection. On the other hand if β is very large, the algorithm would be primarily MNHS in nature. The placement of nodes would lose importance as compared to the path formulation between them.

VI. Results

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. Here we assume that the map is available in form of grid of size MXN. Each of the cells 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. The black regions here signify the presence of obstacles.

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.

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. In all the cases the map was of size 1000X1000 and the robot was supposed to move from the top left corner to the bottom right corner. The value of α of MNHS was fixed as 2. The EA parameters consisted of 125 individuals, 100 generations, 2 as the elite count, 0.78 crossover rate and 0.06 mutation rate. The individual size β was kept as 5. All simulations were made on a 256 MB RAM and 585 MHz processor. All runs took less than 50 seconds with a very high convergence of the path length.

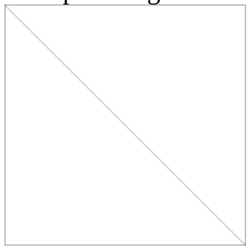

Figure 4: The path generated by MNHS for no obstacle.

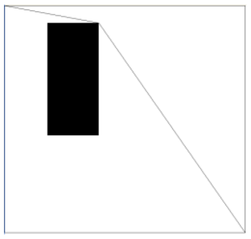

Figure 5: The run for single obstacle in case II

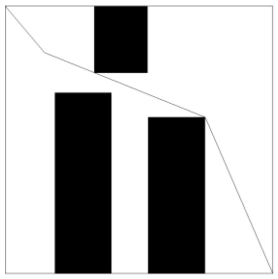

Figure 6: The run for complex obstacles in case III

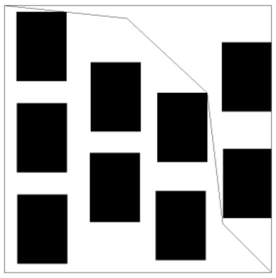

Figure 7: The second run for complex obstacles in case III

Initially we did not place any obstacle in the path from the source to destination. We observed that the algorithm traced the path from the source to the destination following a straight line path. The results of the algorithm are shown in Figure 4. 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. The results of the algorithm are shown in Figure 5. 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. The results of the algorithm are shown in Figure 6 and 7.

VII. Conclusions

In this paper we proposed the use of MNHS and EA 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 of optimality of path which is a very important parameter for the algorithm. At the same time the optimality of the total time of execution was provided by the use of EA. The EA gave limited points in the entire map to the MNHS. Hence using the two algorithms, we were able to optimize both the time and the total path length. This hence proved to be a very important algorithm that could solve much of the problems present in the earlier techniques of robotic path planning.

A beautiful relation exists between the A* algorithm as well as the MNHS. Suppose we want more of path optimality. A very natural choice would be to make the MNHS dominant in the entire algorithm. Similarly say we want more of time optimality. This can be achieved by making the EA more dominant in the algorithm. Hence a controlling measure must exist between the two algorithms. This controlling measure is provided by the use of the EA size constant β. Suppose the value of β is very large. Now the MNHS would be dominant and vice versa.

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 and β was kept constant in the cases presented. The determination and setting of the most optimal values needs to be studied in the future. The algorithm may further be developed for real time dynamic obstacles as well which would help in practical implementation of the algorithm. This may be done by the inclusion of any behavioral robotic controller like he neuro fuzzy controllers.

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

[9] Hem Fkezza-Buet, FrBd6ric Alexandre “Modeling prefrontal functions for robot navigation”

[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

[12] Zhang Huan-cheng, Zhu Miao-liang, “Self-organized architecture for outdoor mobile robot navigation”, Journal of Zhejiang University SCIENCE

[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 and Se-Young Oh, “Hybrid Control for Autonomous Mobile Robot Navigation Using Neural Network Based Behavior Modules and Environment Classification”, 2003 Kluwer Academic Publishers, Manufactured in The Netherlands

[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”

[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

[24] Mitchell, M; “An Introduction to Genetic Algorithms”, 1996, Cambridge, MA: MIT Press.