IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 4, No 8, July 2010
ISSN (Online): 1694-0784
ISSN (Print): 1694-0814

Evolutionary Approach for Mobile Robot Path Planning in Complex environment

Amir Hosseinzadeh[1] and Habib Izadkhah[2]

[1] Department of Computer science, University of Daneshvaran
Tabriz, Iran

[2] Department of Computer science, University of Tabriz
Tabriz, Iran

Abstract

The shortest/optimal path planning in a static environment is essential for the efficient operation of a mobile robot. Recent advances in robotics and machine intelligence have led to the application of modern optimization method such as the genetic algorithm (GA), to solve the path-planning problem. In this paper, the problem of finding the optimal collision free path in complex environments for a mobile robot is solved using a hybrid neural network, Genetic Algorithm and local Search method. We constructed the neural network model of environmental and used this model to establish the relationship between a collision avoidance path and the output of the model. What is new in this work is a novel representation of solutions for evolutionary algorithms that is efficient, simple and also compatible with Hybrid algorithm. The new representation makes it possible to solve the problem with a small population and in a few generations. It also makes the genetic operator simple and allows using an efficient local search operator within the evolutionary algorithm. The performance of the proposed GA approach is tested on eight different environments consisting of polygonal obstacles with increasing complexity.

Keywords: *Genetic Algorithm, Robot Path Planning, Neural Network, Local Search.*

1. Introduction

The robot path planning problem is a very challenging problem in robotics. The main goal of this problem is to construct a collision-free path from a starting position to an end or destination position. The robot path planning problem is part of a larger class of problems pertaining to scheduling and routing, and is known to be NP-hard (NP-complete) [2].

Path planning is usually carried out offline and considers existing knowledge about environment [3]. The best path is defined to be the path with the lowest cost which assumes the shortest collision free path. There have been some efforts for solving this problem using evolutionary algorithms. One of the main challenges when using an evolutionary algorithm for solving a real problem is to representing the problem at hand using evolutionary algorithm fundamentals. Candidate solutions should be coded as chromosomes and well defined genetic operations as well as a suitable penalty function should be designed. As Ibrahim [4] pointed out, this problem can be broken in to several subtasks. A reliable navigation algorithm must be able to 1) Identify the current location of the robot, 2) Avoid any collisions, 3) Determine a path to the object. For this reason, mobile robot navigation problem is a challenging problem, and a number of studies have been attempted, resulting in a significant number of solutions. A number of algorithms have been proposed to address these three important issues. D Huh [7] approached the path finding problem by combining global path planning and local path planning. They used the Dijkstra algorithm for global path planning and the potential field method for local path planning. S. Lee and G. Kardaras, [5] used via points to find the optimum path. They developed a "smart" algorithm which could change the number of via points in response to a different level of complexity of the map upon which paths would be generated.

A heuristic optimization approach is recommended as shown by Hwang [6]. One of these approaches is the use of genetic algorithms. N. G. Bourbakis and L. Vlachavas [8] presented a path planning algorithm that uses a neural network and a skeletonization technique. N. Sadati and J. Taheri [25] presented a combination method consisting of a Hopfield Neural Net (NN) and a genetic algorithm (GA).

In this paper, we constructed the neural network model of environmental information in the workspace for a robot and used this model to establish the relationship between a collision avoidance path and the output of the model. Using this model, the proposed algorithm can know the environment, and search the global optimum simultaneously without knowledge of any passable region. Also a novel genetic representation of path planning problem and a suitable local search operation is proposed. The approach that is taken in this paper for coding is more similar to [10] except that allow to be a sub-path with specifiable shape between two points instead of a straight line and this shape is encoded in gene too. In addition, the chromosome length is fixed in contrast to [10]. This constrain simplifies the genetic operators [11].

Proposed algorithms use local search methods to find local optimums i.e. a point with the best fitness value among its neighbor points. The Proposed algorithm is faster and more accurate that a simple genetic algorithm for some reasons: first, local search methods can serve the genetic operators with solutions those are better in compare to randomly generated solutions. Moreover, genetic algorithms are not good hill-climbers and the combination of them with local search methods alleviates this problem [12].

2. Genetic Algorithm

Genetic algorithms (GA) are global search and optimization techniques modeled from natural selection, genetic and evolution. The GA simulates this process through coding and special operators. The underlying principles of GAs were first published by [13]. Excellent reference on GAs and their applications is found in [14]. A genetic algorithm maintains a population of candidate solutions, where each candidate solution is usually coded as binary string called a chromosome. The best choice of coding has been shown to be a binary coding [13]. A set of chromosomes forms a population, which is evaluated and ranked by fitness evaluation function. The fitness evaluation function play a critical role in GAs because it provides information how good each candidate. The initial population is usually generated at random. The evolution from one generation to the next one involves mainly three steps: fitness evaluation, selection and reproduction [15]. **First**, the current population is evaluated using the fitness evolution function and then ranked based on their fitness. A new generation is created with the goal of improving the fitness. Simple GA uses three operators with probabilistic rules: reproduction, crossover and mutation. First selective reproduction is applied to the current population so that the string makes a number of copies proportional to their own fitness. This results in an intermediate population. **Second**, GA select "parents" from the current population with a bias that better chromosome are likely to be selected. This is accomplished by the fitness value or ranking of a chromosome. **Third**, GA reproduces "children" (new strings) from selected parents using crossover and/or mutation operators. Crossover is basically consists in a random exchange of bits between two strings of the intermediate population. Finally, the mutation operator alters randomly some bits of the new strings. This algorithm terminates when an acceptable solution is found, when convergence criterions met or when a predetermined limit number of iteration is reached. The main features of GAs are that they can explore the search space in parallel and don't need the function to be optimized to be differentiable or have any smooth properties. The precision of the solution obtained depends on the number of bits used to code a particular variable (length of chromosome).

3. Neural Networks (NN)

The concept of using Neural Networks for Robot Motion Planning (RMP) was first used in [16]. A novel biologically-inspired general neural network approach exists for real-time collision-free RMP in a dynamic environment [17]. This general model can be applied to point mobile robots, manipulator robots, car-like robots, and multi-robot systems. The state space of the NN is the configuration space of the robot, and the dynamically varying environment is represented by the dynamic activity landscape of the neural network. The target globally attracts the robot in whole state space, while the obstacles locally push the robot away to avoid collisions. The real-time robot motion is planned through the dynamic activity landscape of the neural network without explicitly searching over the free space or the collision-free paths, without explicitly optimizing any cost function, without any prior knowledge of the dynamic environment, without any learning process, and without any local collision checking procedures. Therefore, the model algorithm is computationally efficient [18]. In [19] an NN approach to path planning for two dimensional robot motion is developed. Also in [20] a neural network approach for the local navigation of a mobile robot via

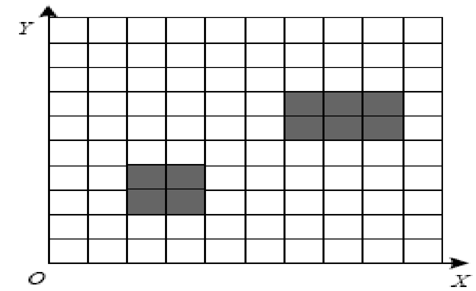
Perception maps is presented. In 1995, the collision identification between convex polyhedral using neural networks is implemented [21]. A cache-genetic-based modular fuzzy neural network is presented in [22] for robot path planning. Frontzek [23] constructed a flexible path planning method for real-time applications using A* method and Neural Radial Basis Function networks. An NN model is developed in [24] to real time MP and control of robot manipulators. RMP problem is solved in [25] using Hopfield neural networks in a fuzzified environment. Also in 2003, a Non-learning ANN approach to MP for the Pioneer robot is extended [26]. An NN approach is presented in [27] for dynamic task assignment of multi robots. Eventually, RL-ART2 NN-based mobile robot path planning is developed in 2007 in [28].

4. Environment Modeling by Neural Network

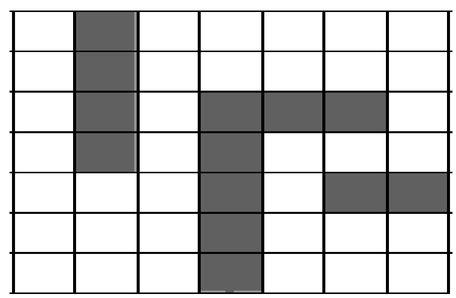
For constructed neural network model of environmental, we suppose that the robot moves in a limited two-dimensional space and the obstacles in the workspace can be described as convex polygons. Also static environment with different obstacles for motion of a mobile robot is considered as the problem of mobile robot path planning.

We suppose that the workspace for a robot is as shown in Fig.1, at Figure. (1a) example of an easy search space and (1b) example of an complex search space; and that the shadowed parts represent the obstacles. According to Zhu et al.(2002) [1], the environment can be described by the neural network shown in Fig.2 which can be written as Eq.(1).
where $C_i^{\,1}$, $C_i^{\,2}$: the outputs of the nodes of the top layer; T_I: the input of the nodes of the top layer; θ_T the threshold of the nodes of the top layer; O_{Mm}: the output of the mth node of the medium layer; I_{Mm}: the input of the mth node of the medium layer; θ_{Mm} : the threshold of the mth node of the medium layer; w_{xm}, w_{ym}: the weights from the input layer to the medium layer; $f(x)=1/(1+e^{\frac{X}{T}})$: the excitation function; $(X_i\ ,Y_i)$: a random point in the workspace.

$$\begin{cases} C_i^l = f(T_I), \\ T_I = \sum_{m=1}^{M} O_{Mm} + \theta_T\,, \\ O_{Mm} = f(I_{Mm}), \\ I_{Mm} = w_{xm}X_i + w_{ym}Y_i + \theta_{Mm} \end{cases} \qquad (1)$$



(1a) A example of an easy search



(1b) A example of an complex search

Fig.1 The workspace for a robot

The output of the neural network model of each point in the workspace is 0 or 1. C_{ik}=1 (k=1,2) implies that $(X_i\ ,Y_i)$ is in the kth obstacle region, otherwise the point is not in it. If the radius of the robot could be ignored, the robot can be regarded as a particle. If the neural network model of the point $(X_i\ ,Y_i)$ produces C_{ik} =1 (k=1,2) when the robot arrives, it collides with the kth obstacle in the workspace. On the contrary, it does not. Thus the collision avoidance path can be described as the path where the output of the neural network model of each $(X_i\ ,Y_i)$ is C_{ik} =0 (k=1,2). So the path planning algorithm can know the current environment information in real-time according to the output of the neural network model. In Fig.2, only two obstacles are included in the workspace, so the neural network model is simple. If there are more than two obstacles in it, the model is more complexity, as shown in Fig.3.

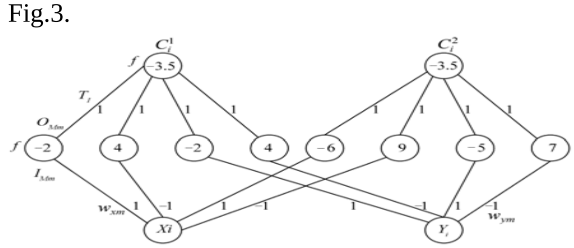


Fig.**2** The neural network for the environment (from[1])

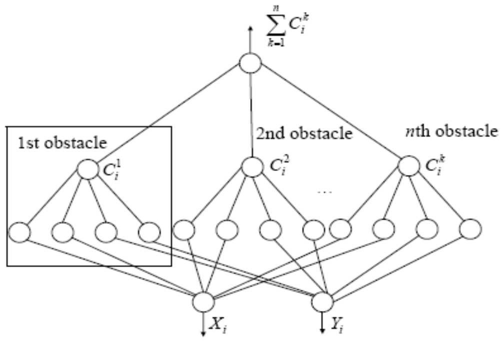


Fig.3 The neural network for the multi-obstacle environment

5. Proposed Algorithm

In this section, proposed algorithm for solving the path planning problem is described. We constructed the neural network model of environmental for a robot and used the output of the model to recognition collision avoidance path. Also a novel genetic representation of path planning problem and a suitable local search operation is proposed. The combination algorithms use local search methods to find local optimums i.e. a point with the best fitness value among its neighbor points. The Proposed algorithm is faster and more accurate that a simple genetic algorithm. The speed of genetic algorithm depends heavily on the encoding scheme of the chromosomes and on the genetic operators that work on these chromosomes [29],[30]. In order to speed up a GA, the chromosome's and gene's structures need to be as simple as possible. In addition, only a few, but very effective, reproduction operators should be applied on the chromosomes. Having that in mind, a novel encoding technique, based on value encoding [2], was developed. In value encoding, any type of number, character or object can be assigned to each gene. Most traditional methods that use value encoding do not use the information of a gene's position. However, this information allows the novel technique to more efficiently use the value-encoding scheme, which keeps the gene structure as simple as possible.

The chromosome structure must have sufficient information about the entire path from the start point to the end-point in order to be able to represent it. The previous genotype encoding technique such as, T. Geisler [31] contained only two variables, Path-Location and Path-Direction. That encoding technique allowed only row-wise movements (In a row-based movement, the robot starts moving row by row from the start-point to the end-point.). Next, Aditia Hermanu [32] modified the genotype by introducing a new instruction flag for each path, called Path-Flag. This Flag instructs the next movement type for each step of the movement. Therefore, this genotype allowed the robot to plan either a row-wise or a column-wise movement according to the search space arrangements. But, neither of these two previous structures was able to combine both row-wise and column-wise paths while planning for a single path. This caused the robot to fail for complex environments that required the robot to move both row-wise and column-wise within those search spaces. Thus, the encoding that we have applied in this paper to address the path-planning problem consists of five variables: Path-Location, Path-Direction, Path-feasibility, Path-Flag, and Path-Switch. While the previous work required either a row-wise or column-wise movement, the new genotype is able to plan both row-wise and column-wise within a single search space. Hence, the path has more flexibility to switch between the two movements modes.

The remainder of this paper presents the novel gene structure and the GA operators.

5.1 Encoding Technique

A chromosome that uses the proposed gene structure with the five variables location, direction, feasibility, Flag, and Switch represents an entire path. The proposed **Encoding** code consists of a 1-bit flag that is of type Boolean (0 or 1) for each chromosome. The main responsibility of this bit is to tell the robot whether the next step of the movement is row-wise or column-wise (0 for row-wise or 1 for column-wise). In the previous proposed **Encoding** code, so far we do not have any information about whether a path is feasible; i.e., A step is *feasible* if all steps between the start-point and the end-point of that step do not contain any obstacles. To indicate the feasibility of each step, a Boolean variable is added to the chromosome structure that called Path- *feasibility*. This variable is not a part of the chromosome, since it is not assigned a random value during initialization. The information on a step's feasibility is later used to determine a path's overall fitness. Also The proposed encoding technique uses the information of a gene's position as well as the value stored at that position as an x- and a y-coordinate. These coordinates define the location of a cell within the row and column system. Thus, a gene's position within a chromosome corresponds to a row-number. The value, stored in a gene, in a variable called location, corresponds to a column-number. The chromosome described so far only represents vertices ('corner points' or 'intermediate steps') of a path. To send a robot on a straight line directly from a center of one vertex to the center of the next vertex would mean that the robot moves on a diagonal line across many adjacent cells. This could cause problems if not all adjacent cells that the robot is to traverse going from one cell to the next are free of obstacles, as shown in Figure 4. A better approach is to go to the side (horizontal) first, turn, and then goes down (vertical), or vice versa. To indicate the first direction the robot will turn to proceed to the next vertex, a variable called Path-Direction is added to the gene structure. Direction is a Boolean variable, which has either the value horizontal (true) or vertical (false).

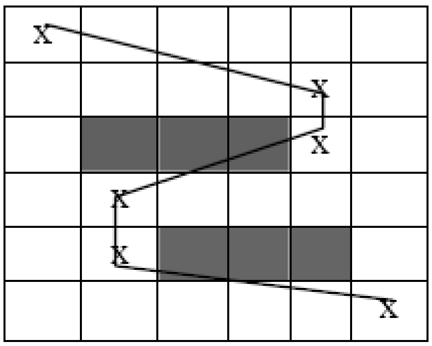


Fig 4. Problem with diagonal movement of the robot

None of the previous research was able to combine both row-wise and column-wise paths while planning for a single path. This caused the robot to fail for complex environments that Notice from the environment in Figure 1a, either row-wise or column-wise movement can address this problem since it does not have a complex obstacle arrangement. On the other hand, the environment in Figure 1b requires both row-wise and column-wise movement in order to be traversed. Therefore, it is considered a more complex search space compared to the search space shown in Figure 1b. Previous research was not able to address this type of environment.

In order to overcome this movement restriction, we added the Path-Switch variable, to the genotype. This variable enables the robot to switch back and forth between a row-wise and a column-wise movement in a single path.

5.2 Computing fitness

The fitness function is an important factor for the convergence and the stability of genetic algorithm. The collision avoidance and the shortest distance should be considered in path planning. The population of paths is evaluated during each reproduction cycle. The evaluation is based on the paths' fitness, which depends on how suitable the solution (path) is according to the problem. In preliminary evaluations, the values for the path length (F_{Length}), the number of turns ($F_{NumberOfTurns}$) and the number of infeasible steps ($F_{InfeasibleSteps}$) are determined for each path in the population.

The path length can be optimized for minimum distance whose fitness function F_{Length} can be described by Eq.(2).

$$F_{length} = \sum_{i=0}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \qquad (2)$$

The number of turns value are set in relation to the entire population and therefore stored as fractional values from 0 to 1, where 1 indicates the optimal fitness value. The least the number of turns steps in the population corresponds to $F_{NumberOfTurns} = 1$ and the greatest number of turns steps corresponds to $F_{NumberOfTurns} = 0$.

Collision avoidance is essential to path planning and makes the mobile robot travel in the workspace safely. Collision avoidance Combined with the output of the neural network model, the fitness function of collision avoidance, $F_{InfeasibleSteps}$ can be depicted as:

$$F_{InfeasibileSteps} = \begin{cases} 1 & if \sum C_i^{\,k} = 0 \quad (k = 1,2,\ldots,n) \\ 0 & other \end{cases} \qquad (3)$$

Thus the collision avoidance path can be described as the path where the output of the neural network model of each (X_i ,Y_i) is C_{ik} =0 (k=1,2,..,n).

F_{Path} is the fitness value for the entire path whose fitness function fit2 can be described by Eq.(4).

$$F_{path} = F_{InFeasible\ Steps} . F_{numberofTurns} \qquad (4)$$

Thus the final fitness function is constructed as shown in Eq.(5). When the fitness function fit reaches the maximum, the optimal path is found. This not only makes computation simple but also overcomes the disadvantage of the instability from the summation of evaluation function weights.

$$Fit = \frac{F_{InFeasible\ Steps} . F_{numberofTurns}}{\sum_{i=0}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}} \qquad (5)$$

5.3 Crossover

The crossover operator has been modified to produce two offspring chromosomes with each crossover operation. During the operation of reproduction crossover is applied on the chosen parent chromosomes only within a certain probability, the crossover probability. In the chosen crossover operator, two parent chromosomes are combined applying a single-cross-point, value encoding crossover [2].This is achieved by using the gene information, which were not used to build offspring one, in order to build a second chromosome. Furthermore, the crossover operation acts only on the location and the direction parts of the chromosomes, as it does not make sense to have single point crossover operator acting on the two switching points.

5.4 Mutation

For mutation [2], almost every operation that changes the order of genes within a chromosome or that changes a gene's value (such as location or direction) is a valid mutation operator. The mutation operator has been designed according to the addressed path planning problem.

The mutation operator used here checks with the mutation probability for every single gene and decides whether it should be mutated or not. If a gene is to be mutated, a random number between 1 and the total number of rows or columns in the search space is assigned to location, and a random direction, either vertical or horizontal, is assigned to direction. Unlike the crossover operator that does not act on the switching points, mutation operator will affect these points.

The fitness of all affected genes (steps) is re-evaluated and stored in the variable feasibility immediately after the changes in location and direction are made. Each step's fitness is therefore always up to date.

Elitism was also used in order to keep the best individual (path) within a generation. If elitism is applied, the fittest chromosome path is copied to the offspring population without any changes.

5.5 Local Search Operator

Among various types of search methods that explore a limited neighborhood of a local optimum, called local search methods, which of them that use gradient information as well as value information are generally more efficient. But gradient information obtained through considerable amount of calculation that depends on the dimension of the search space. The dimension of the search space is equal to the number of genes in the problem at hand. Therefore the amount of calculation is reduced significantly when the proposed representation is used rather than the previous representations.

The gradient-based local search method used in this paper reduces the penalty of the chromosome through modifying the start and end points of genes based on the gradient information of the penalty function. This method is generally known as Gradient Ascent [33]. A typical path that is modified with this method is depicted in Figure. 5. It is seen that the path found using the local search has a shorter length, hence a lower penalty.

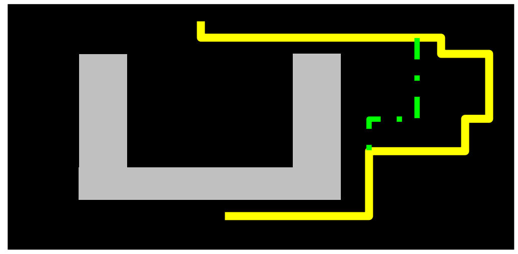


Fig. 5. Local Search operator. The path before the hill-climbing (solid) and after it (dashed).

6. Experiments and Their Results

In this section, simulation and experimental results are obtained by applying the proposed path planning method to navigate a mobile robot. After determining the optimum operators and parameters for the GA, path-planning simulations have been conducted on different sized search spaces and with different obstacle configurations.

The proposed path-planning GA was tested on eight different search spaces (SPSet01 ~ SPSet08) and the results were compared against those yielded by the Geisler and Hermanu's path-planning GA [32]. For each set of tests, the GAs were run 15 times and the average success rates were calculated and are shown in Table 1. Figure 8 shows an example search space with obstacle configuration. It is obvious this search space is very easy and can be navigated by previous methods [32] via a row–wise movement as well. Figure 10 shows the resulted paths generated by our new path-planning GA on different search spaces.

(a) The Proposed Genetic Algorithm (PGA) parameters:
Population Size = 30, Number of Generations = 150, Crossover probability = 0.8, Mutation probability = 0.05.

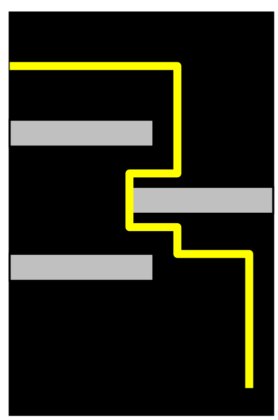
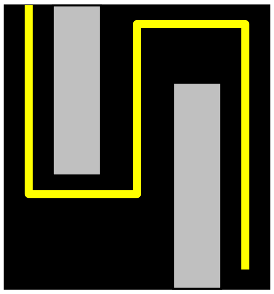


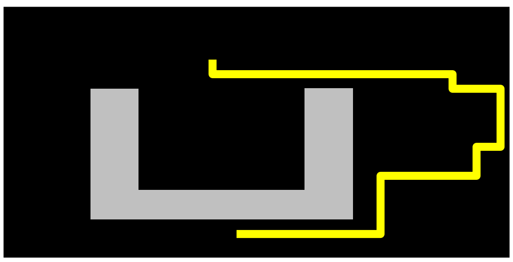
Fig. 8 an example search space with obstacle (SPSet01)

(b) Path(Chromosome) produced by proposed GA for Fig.8.

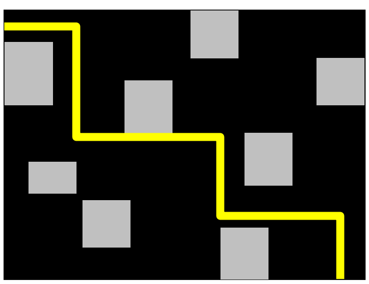
Row index	1	2	3	4	5	6	7	8	9
Location	0	6	4	4	5	7	9	9	9
Direction	1	0	1	0	1	1	1	1	1



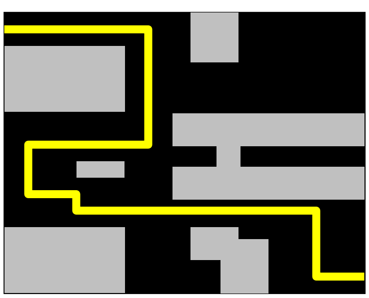
SPSet02



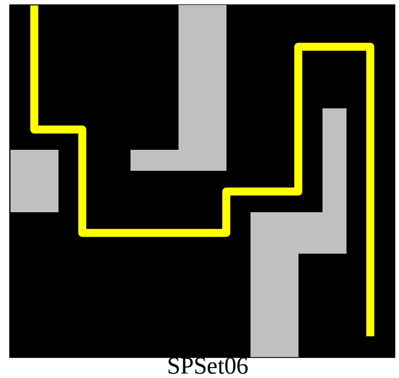
SPSet03



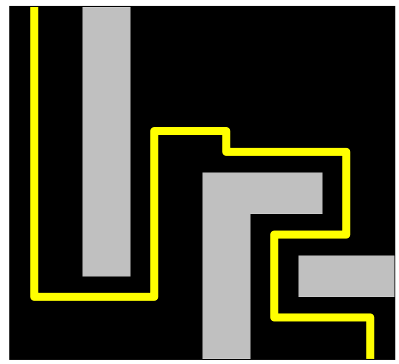
SPSet04



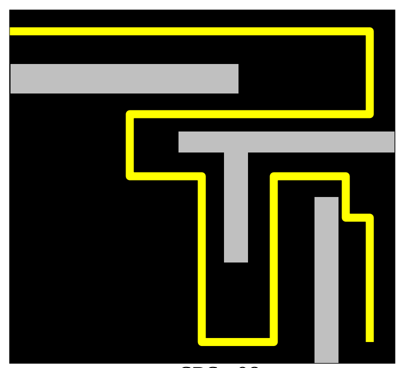
SPSet05



SPSet06



SPSet07



SPSet08

Search spaces 07 and 08 were the two that required both row-wise and column-wise movement of the robot in order for the robot to get around the obstacles. As shown in Table 1, only the genotype developed in this research can address these search spaces as previous genotypes failed to handle these environments. This is because the new genotype allows the robot to switch back and forth from row-wise to column-wise movement and vice versa, which means a free movement of the robot in any direction.

TABLE 1.
SUCCESS RATES COMPARISON BETWEEN THE PROPOSED GA AND THE PREVIOUS METHODS

Search Space	Success Rate (%)		
	Geisler GA	Hermanu GA	Proposed Algorithm
SPSet01	100	93.3	100
SPSet02	0.00	86.7	100
SPSet03	100	100	100
SPSet04	53	80	100
SPSet05	0.00	100	100
SPSet06	0.00	20	100
SPSet07	0.00	0.00	89.4
SPSet08	0.00	86.7	83.2

7. CONCLUSIONS

In this paper, a novel representation for the path planning problem that was suitable for evolutionary algorithm especially hybrid algorithm was proposed. A local search operator for tuning the start and end points of sub-paths was also proposed.

The experimental results illustrate that in the path planning problem, the path found in a few generations with a relatively small population of chromosomes. The results also demonstrate that the solution found using a hybrid algorithm is more optimal than that found by a simple genetic algorithm. Optimization of the shape of sub-paths using an appropriate local search method is our future step.

REFERENCES

[1] Zhu, Y., Chang, J., Wang, S., 2002.A New Path-planning Algorithm for Mobile Robot Based on Neural Network. IEEE Region Tenth Conference on Computers, Communications, Control and Power Engineering. Beijing, p.1570-1573.

[2] Obitko, M., “Genetic Algorithms”, Internet publication, 1998, http://cs.felk.cvut.cz/~xobitko/ga/main.html

[3] Gerke M.: Genetic Path Planning for Mobile Robots. In Proc. of the American Control Conference (1999).

[4] M. Youself Ibrahim and Allwyn Fernandes, “Study on Mobile Robot Navigation Techniques,” IEEE ICIT, Tunisia, December 8-10, 2004.

[5] S. Lee and G. Kararas, “Collision-Free Path Planning with Neural Networks,” 1997 IEEE Interational Conference on Robotics and Automation.

[6] Hwang, Y.K., Ahuja, N., “Gross Motion Planning – A Survey”, ACM Computing Surveys, volume 24, issue 3, pp. 219-291, September 1992.

[7] D. Huh, J, Park, U. Huh, H. Kim, “Path Planning and Navigation for Autonomus Mobile Robot,” IECON 02 IEEE annual conference.

[8] N.G. Bourbakis, D. Goladman, R.Fematt, I. Vlachavas, L.H. Tsoukalas. “Path Planning in a 2-D Known Space using Neural Networks and Skeletonization,” Man and Cybernetics IEEE, 1997.

[9] N. Sadati and J. Taheri. “Genetic Algorithm in Robot Path Planning Problem in Crisp and Fuzzified Environments,” IEEE ICIT02, Bangkok, Thailand

[10] Hu Y., yang S.X.: Knowledge Based Genetic Algorithm for Path Planning of a Mobile Robot. In Proc. of the 2004 IEEE Intern. Conference on Robotics & Automation (2004) 4350-4355.

[11] Sugihara, K., Smith J.: Genetic Algorithms for Adaptive Motion Planning of an Autonomous Mobile Robot. In Proc. of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (1997) 138-146.

[12] Land, M. W. S.: Evolutionary Algorithms with Local Search for Combinatorial Optimization. Ph. D. Thesis, University of California, San Diego, 1998.

[13] J. H. Holland, "Adaptation in Natural and Artificial Systems", Ann Arbor, MI, The University of Michigan Press, 1975.

[14] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison- Wesley Publishing Company, 1989.

[15] Y. John and R. Langari, "Fuzzy Logic: Intelligence, Control, and Information", Prentice- Hall Inc., Ch (17), pp.469-470, 1999.

[16] Zacksenhouse, M.; DeFigueiredo, R. J. P. and Johnson, D. H.,"A neural network architecture for cue-based motion planning". Proc. IEEE Int. Conf. on Decision and Control,(1988) pp. 324 – 327.

[17] Zelinsky, A. “Using path transforms to guide the search for find path in 2D”, Int. J. Rob. Res, 1994,(13)4, pp. 315-325.

[18] Glasius, R.; Komoda, A. and Gielen, S., C., A., M., “Neural network dynamics for path planning and obstacle avoidance”. (1995) Proc. IEEE Neural Networks, pp. 125-133.

[19] Kozakiewicz, C., Ejiri, M. "Neural network approach to path planning for two dimensional robot motion". Proc. IEEE/RSJ (IROS '91) vol. 2 (1991) pp. 818 – 823.

[20] Santos, V.; Goncalves, J. G. M. and Vaz, F.; "Perception maps for the local navigation of a mobile robot: a neural network approach" Proc. IEEE Int. Conf. on Rob. and Aut., vol. 3 (1994) pp. 2193-2198.

[21] Jing Y., “Collision identification between convex polyhedra using neural networks”, IEEE Trans. on Neural Networks, (1995), Vol. (6) No. 6, pp. 1411-1419.

[22] Kun H. W.; Chin H. C. and Jiann D. L.; "A cache-genetic-based modular fuzzy neural network for robot path planning", Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics,Vol4, (1996) pp. 3089 - 3094.

[23] Frontzek, T.; Goerke, N.; Eckmiller, R.; "Flexible path planning for realtime applications using A*-method and neural RBF-networks",Proc. IEEEICRA,(1998) pp.1417-1422.

[24] Xianyi Y. and Meng, M.; "A neural network approach to real-time motion planning and control of robot manipulators" Proc. IEEE/ SMC, vol. 4,(1999) pp. 674-679.
[25] Sadati, N. and Taheri, J.; "Solving robot motion planning problem using Hopfield neural network in a fuzzified environment" Proc. IEEE/FUZZ.,Vol.2, (2002) pp.1144-1149.
[26] Erickson, D.; "Non-learning artificial neural network approach to motion planning for the Pioneer robot". IEEE/IROS Vol. 1,(2003) pp. 112-117.
[27] Zhu, A. and Yang, S. X.;"A Neural Network Approach to Dynamic Task Assignment of Multirobots" IEEE Trans. Neural Networks, Vol. 17, No. 5,(2006) pp. 1278-1287.
[28] Jian, F.; MinRui, F. and ShiWei M., “RL-ART2 Neural Network Based Mobile Robot Path Planning”, Proc. ISDA (2006), Vol. 2, pp. 581-585.
[29] Thomas Bäck, Ulrich Hammel, “Evolutionary Computation: Comments on the History and Current State”, IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1,April 1997.
[30] Jing Xiao, Lixin Zhang, “Adaptive Evolutionary Planner/Navigator for Mobile Robots”, IEEE Transactions on Evolutionary Computation, Vol. 1 No. 1, April 1997.
[31] Geisler, T. and Manikas, T., “Autonomous Robot Navigation System Using a Novel Value Encoded Genetic Algorithm”, Proceeding of IEEE Midwest Symposium on Circuits and Systems, Tulsa, OK, 2002.
[32] Hermanu, A., “Genetic Algorithm with Modified Novel Value Encoding Technique for Autonomous Robot Navigation”, Master’s Thesis, The University of Tulsa, OK, 2002.
[33] Principe J. C., Euliano N. R., Lefebvre W. C.: Neural and Adaptive Systems. Jone Wiley & Sons, Inc., 2000.

Habib Izadkhah received the B.Sc. degree in Computer Science from University of PayamNour (Iran) in 2005, the M.Sc. degree in software engineering from Shabestar Islamic Azad university in 2008. He is currently a Ph.D. student in the Faculty of Mathematical Sciences at University of Tabriz, Iran. His current researches focus on view based software engineering and reverse engineering.

Amir Hosseinzadeh received the B.Sc. degree in Computer Science from Shabestar Islamic Azad university (Iran) in 2005, the M.Sc. degree in software engineering from Shabestar Islamic Azad university in 2008. His current researches focus on view based software engineering and reverse engineering.