Path planning for a mobile robot using genetic algorithms

Conference Paper · October 2004

DOI: 10.1109/ICEEC.2004.1374415 · Source: IEEE Xplore

2 authors:

Gihan Nagib
Fayoum University
19 PUBLICATIONS **192** CITATIONS
SEE PROFILE

Wahied Gharieb
Ain Shams University
35 PUBLICATIONS **291** CITATIONS
SEE PROFILE

PATH PLANNING FOR A MOBILE ROBOT USING GENETIC ALGORITHMS

Gihan NAGIB and W. GHARIEB***

** Electrical Engineering Dept., Faculty of Engineering*
Cairo University - Fayoum Branch, E-mail: gihannagib949@hotmail.com

*** Computer and Systems Engineering Dept., Faculty of Engineering*
Ain Shams University, E-mail: wahied@hotmail.com

Abstract *- This paper presents a new algorithm for global path planning to a goal for a mobile robot using Genetic Algorithm (GA). A genetic algorithm is used to find the optimal path for a mobile robot to move in a static environment expressed by a map with nodes and links. Locations of target and obstacles to find an optimal path are given in an environment that is a 2-D workplace. Each via point (landmark) in the net is a gene which is represented using binary code. The number of genes in one chromosome is function of the number of obstacles in the map. Therefore, we used a fixed length chromosome. The generated robot path is optimal in the sense of the shortest distance. The robot has a starting point and a target point under the assumption that the robot passes each point only once or not at all. The obtained results in simulation affirmed the potential of the proposed algorithm.*

I. INTRODUCTION

The path planning problem of a mobile robot can be stated as: given (starting location, goal location, 2-D map of workplace including static obstacles), plan a collision-free path between two specified points in satisfying an optimization criterion with constraints (most commonly: shortest path). The path planning problem is computationally very expensive. Although a great deal of research has been performed to further a solution to this problem, conventional approaches tend to be inflexible in response to:

- Different optimization goals and changes of goals
- Uncertainties in an environments and
- Different constraints on computational resources.

A review of the existing approaches for solving path-planning problem is provided in [1]. Many methods have been reported to generate an optimal path such: dynamic programming and distance transform methods. In the ***Dynamic programming*** method if the start point is Ps, goal point is Pg and sub-goal point is Pi, the path generation method is how to determine a sequence of sub-goals picking out the sub-goals from their set Pi (i=1,2,…,g-1). We must calculate all possible paths and select one has the smallest cost value. Very large computing power is required, especially in environments that have many sub-goals. In the ***distance transform*** method, the task of path planning is finding a path from the goal location back to the start location. The path planning procedure covers the environment with a uniform grid and the propagate distances through free space from the goal cell. The distance wave front flows around obstacles and eventually through all free space in the environment. For any starting point within the environment representing the initial position of the mobile robot, the shortest path to the goal is traced by walking downhill via the steepest descent path. However, ambiguity of optimal paths exists where there exist two or more cells (nodes) to choose with the same least distance transform. The above two methods require a very large computation power in case of environment that have large number of sub-goals and obstacles.

There are many approaches suggested by researchers to solve the path planning problems of mobile robots in presence of static and moving obstacles based on soft computing. Soft computing consists of fuzzy logic, neural networks and evolutionary Computation (genetic and evolutionary algorithms GA& EA). An invited paper which represents an application to soft computing has been reported in literature [2]. There have been many attempts to use fuzzy logic for motion planning of mobile robot [3-4]. Recently, it has been widespread using evolutionary computations which are robust, global and most straightforward to apply in situations where there is little or no priori knowledge about the problem to solve [5]-[9]. GA and EA require no derivative information or formal initial estimates of the solution, and because they are stochastic in nature, they are capable of searching the entire solution space with more likelihood of finding the global optimum. Previous research on the path planning can be classified as one of two approaches: model-based and sensor based. In the model based a priori models of known obstacles are used to generate a collision free-path. In contrast, the sensor-based approach aims to detect and avoid unknown obstacles (dynamic environment). In this paper, a new algorithm finding out an optimal path planning as model based is proposed.

The reminder of this paper is organized as follows: in section II a brief introduction to the genetic algorithm is presented. While, in section III, the problem formulation is given. In section IV, the proposed algorithm is introduced and in section V, simulation results are presented.

Conclusions and suggestions for future research are included in section VI.

II. 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 [10]. Excellent reference on GAs and their applications is found in [11]. 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 [10]. 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 [12]. **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).

III. PROBLEM FORMULATION

The problem is stated as follows. The mobile robot moves in a closed workspace (indoor area). This area is described by a 2-D map. The map includes a finite number of static obstacles and a set of via points (landmarks) to help the robot in avoiding obstacles. The objective is to plan a collision-free optimal path between the starting point and the goal point. That means; an objective criterion has to be optimized with a collision free constraint. If the 2-D map includes (n) via points, the space of all possible solutions is very huge. The C-space of a robot is the space of all configurations of the robot path. The path planning problem becomes equivalent to the motion planning of a point robot in a C-space. The collision free C-space refers to the locus of all points in the C-space that represent feasible and collision free configurations. That means, an optimal path has to be feasible. In general, there are three different optimization goals can be designed to find collision free optimal path:

1) Minimize distance traveled (shortest path)
2) Maintain a smooth trajectory and
3) Satisfy the clearance requirements (the robot should not approach the obstacles too closely)

Figure 1 illustrates the different situations:

- Path (1) is not feasible (obstacle collision)
- Path (2) is the shortest distance and feasible
- Path (3) is feasible and more safer (farthest from obstacles)
- Path (4) is feasible but not met the optimal criterion.

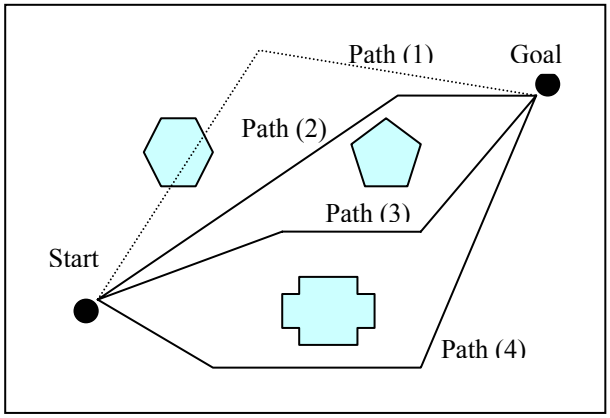

Figure 1
Path Planning Objectives

These objectives can be weighted with different factors relative to their importance in a static optimization problem. In this paper, we will be focused to find the collision free shortest path between starting point (P_s) and goal point (P_g) using a genetic algorithm.

IV. PROPOSED PATH PLANNING ALGORITHM

In this section, a genetic algorithm (GA) is used to find the optimal paths for a robot to move in a static environment expressed by a map with nodes (via points) and links. This application can be found in exhibitions, museums, and laboratories to transport materials. In the proposed algorithm, any path from the starting point to the goal is a solution. At the beginning a random populations of string is generated which represents admissible (feasible) or un-admissible (unfeasible)

solutions. Un-admissible solutions are strings that cannot reach the target. That means, the string solution would lead the robot to move in collision with obstacles. Admissible solutions are strings that can reach the target. In order to move the mobile robot toward the goal obstacles avoidance, it should found a best string that reaches the target in the shortest path. The un-admissible solution has lowest fitness (zero fitness). The X-Y map that considered in our paper is a square area of 15 m^2 as shown in figure 2.

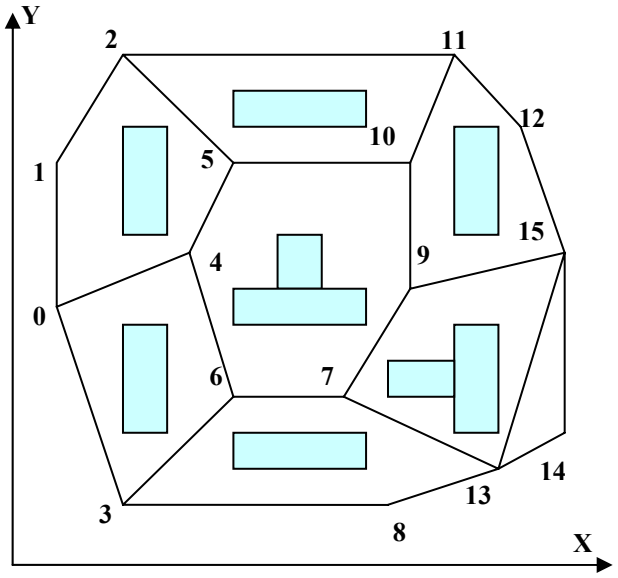

Figure 2
Workspace Map

The robot has a starting point and a goal point on the map under the assumption that in a path planning the robot passes each point only once or not at all. Each node has a number in Fig.2 and the nodes are used to encode a path as a string expressed by the order on numbers. For example, 0-4-5-10-9-15 is a path with P_s=0 and P_g=15. Using GA, via points are selected randomly at first, while adjacent numbers (points) must be connected with a link on the map. In this field, most researchers have used a variable length order-based string (chromosome). This chromosome is represented by a string bits (i.e. natural representation not binary). Hence special crossover and mutation operators must be adopted. During the crossover, a string that has an efficient fitness is randomly selected as a parent. If the second parent contains the common number in the first one both strings exchange the part of their strings following the common number, if not another string is selected as the second parent and the same procedure is followed. For example using the map in figure 2:
Parent1: 0-1-5-10-11-12-15
Parent2: 0-3-6-4-5-10-9-15
Each parent have a common point 10, the underlined parts of each string are exchanged, yielding: the two children are
Child1: 0-1-5-10-9-15
Child2: 0-3-6-4-5-10-11-12-15

After crossover children are checked to determine whether each string has repeated number. If so, the part of string between the repeated numbers is cut off. Some correction then required because it might be the case that the child is not admissible solution [11]. This approach makes the algorithm more complex. The important question is how to make the binary coding so possible and easy to be used in a fixed length chromosome definition. One of the more challenging aspects in our proposed method is encoding each string in a binary code with fixed length. A path of a mobile robot is encoded using binary numbers where each gene (via point) in a chromosome is encoded by 4 bits binary as in the table I.

Table I
Map Via Points

Point index	Point code	X-value (m)	Y-value (m)
0	0000	1	7
1	0001	1	11
2	0010	3	14
3	0011	3	1
4	0100	5	8
5	0101	6	11
6	0110	6	4
7	0111	8	4
8	1000	10	1
9	1001	10	7
10	1010	10	11
11	1011	11	14
12	1100	13	12
13	1101	12	2
14	1110	14	3
15	1111	14	8

The following subsections describe the important components of the proposed genetic algorithm.

A. Chromosomes and Initialization

A chromosome corresponds to possible solution of the optimization problem. Thus each chromosome represents a path which consists of straight line segments, as the sequence of nodes (points) with the first node indicating the starting point of the first segment, followed by intermediate nodes (via points) between segments, and the last node indicating the ending point of the last segment, which is the goal. The default maximum chromosome length is equal to the number of map points times the gene length (4-bit binary code). In our case, we have 16 points where each point is coded in 4 binary bits. That means; the chromosome length is equal to 16X4=64 bits. This result is straight forward but we can reduce efficiently the chromosome length to reduce the required computation power. From the map topology, we have observed that the optimal path to reach the goal point is related to the number of obstacles.

If the number of static obstacles is equal m, the shortest path consists at most of (m+2) points or (m+1) linear segments. This relation assumes that the obstacle

shape is not complicated and can be considered as mass point. For example, if there is no any obstacle (m = 0), the shortest path consists of a one linear segment from the start to goal point; (number of points =2). In figure 1 the number of static obstacles m=3; the shortest path consists of 3 point (start, via, goal) which is less than 5 (m+2). In figure 2, m =7; the shortest path from points 0 to point 15 is {0-4-6-7-9-15} (6 points) which is less than 9 (m+2). Thus the following equation will be used to define the chromosome length.

$$C_n = m+2 \quad (1)$$

Where m number of static obstacles
C_n number of genes in the chromosome

For the given map in figure 2, the chromosome length = (7+2) X 4 =36 binary bits. The initial population of chromosomes can be randomly generated such that each chromosome has a random m genes (via points), while the start and goal points are fixed in the population.

B. Evaluation

The choice of a fitness function is usually very specific to the problem under condition. The evaluation function of a chromosome measures the objective cost function. The distance of a path indicated by the chromosome is used to calculate its fitness. Since the fitness should increase as the distance decreases. Thus, the fitness function (F) of a feasible path is evaluated as:

$$F = \begin{cases} \dfrac{1}{\sum_{i=1}^{m+1} d(P_i, P_{i+1})} & ; \text{ Feasible path} \\ 0 & ; \text{ Infeasible path} \end{cases} \quad (2)$$

Where $d(p_i, p_{i+1})$ is the segment length between a point P_i and next point P_{i+1} in the chromosome of length m+2 points.

Table II
Linked Points

Point index	Linked points
0	1, 3, 4
1	0, 2
2	1, 5, 11
3	0, 6, 8
4	0, 5, 6
5	2, 4, 10
6	3, 4, 7
7	6, 9, 13
8	3, 13
9	7, 10, 15
10	5, 9, 11
11	2, 10, 12
12	11, 15
13	7, 8, 14 , 15
14	13, 15
15	9, 12, 13, 14

The segment length can be calculated using the given X-Y coordinate values in table I as follows:

$$d(P_{i+1}, P_i) = \sqrt{(X_{i+1} - X_i)^2 + (Y_{i+1} - Y_i)^2} \quad (3)$$

If the path is not feasible, its fitness is equal to zero. Our algorithm can trace the path points to detect if it is feasible or not using table II. The linked points are the admissible next points to the current one in the solution in order to avoid obstacles.

C. Operators

In our algorithm, we use the common two genetic operators: ***crossover*** and ***mutation***. Crossover recombines two 'parent' paths to produce two 'children' new paths in the next generation. We have used two points crossover. Both parent paths are divided randomly into three parts respectively and recombined. The middle part of the first path between crossover bit positions and the middle part of the second path are exchanged to produce the new children. The crossover bit positions are selected randomly along the chromosome length between the positions 5 and 32. These limits are chosen in order to keep the start and goal points without change during the crossover process. For example, given two parents (36 bits) as following, they will be crossover randomly at bit positions 15 and 23:

000000000111111***10000000***0011000001111
000000000000000***01111111***0000000001111

The new two children are:

000000000111111***01111111***0011000001111
000000000000000***10000000***0000000001111

The mutation process is also applied to flip randomly a bit position in the chromosome (between bit position 5 and 32). The pseudo-code of the proposed algorithm is given by:

BEGIN
  Initialize *the starting and goal points*
  Generate *randomly the initial population using m via points in each chromosome*
  While NOT (*convergence condition****) DO***
    Evaluate *the fitness for each chromosome in current population using equation (2)*
    Rank *the population using the fitness values*
    Eliminate *the lowest fitness chromosome*
    Duplicate *the highest fitness chromosome*
    Apply *randomly crossover process between current parents using the given probability, while keeping the start and end points without change in the population*
    Apply *the mutation process with the given probability*
    Generate *the new population*
  END
  Output the best individual found
END

V. Simulation Results

The above algorithm is tested via simulation using MATLAB. The simulation environment has a very powerful string manipulation commands that help to convert easily the numeric variables into strings and vice versa. Consequently, the bit crossover and mutation were so easily to be implemented.

Simulation example

Given the start point is 0 and the goal point is 15. The map in figure 2 is simulated to the path generator. The initial population consists of 6 chromosomes 36 bits of each. The probability of crossover was chosen as 100% and the mutation rate equal to 0.005. The best results are obtained using two points of crossover. The simulation result after 50 generation is given by:

000001000110011001100111100111111111
000001000110011011110111100111111111
000001000110011001100111100111111111
000001000110011001100111100111111111
000001000110011001100111100111111111
000001000110011001100111100111111111

The above result of binary individuals corresponds to the population of paths:

{0-4-6-6-6-7-9-15-15}
{0-4-6-6-15-7-9-15-15}
{0-4-6-6-6-7-9-15-15}
{0-4-6-6-6-7-9-15-15}
{0-4-6-6-6-7-9-15-15}
{0-4-6-6-6-7-9-15-15}

The second chromosome is slightly different due to mutation process. The part of repeated numbers is cut off and the final result has been converged to the optimal/shortest individual {0-4-6-7-9-15} with the length of 17.975 m. The program is tested with other start points and goal points and its convergence was guaranteed to obtain the optimal path in each case.

VI. CONCLUSIONS

A simple genetic algorithm with fixed chromosome length is developed for a mobile robot to obtain a shortest path in 2-D static environment with m obstacles. The developed algorithm uses an efficient coding scheme with fixed length binary string (solution). The chromosome length depends on the number of static obstacles. Its length is fixed to (m+2) genes. This algorithm requires a less computation power than other recent developed algorithms. Variable chromosome length could be required in dynamic environments. In a forthcoming paper, we will use a multi-objective criterion to obtain a shortest path that more smooth and safer in both static and dynamic environment.

VII. REFERENCES

[1] J. C. Latombe, "Robot Motion Planning", Norwell, MA: Kluwer, 1991.

[2] Patricia Melin and Oscar Castillo, "Soft Computing for Intelligent Control of NonLinear Dynamical Systems" Invited paper, Int. Journal of Computational Congnition, Vol. 2, No.1, PP.45-78, March 2004.

[3] H. Surmann, J. Huser, J. Wehking, "Path Planning for Fuzzy Controlled Autonomous Mobile Robot", Proc. 5-th int. conf. on Fuzzy Systems, Vol.3, pp.1660-1665, New Orleans, Louisiana, Sept. 1996.

[4] K. H. Wu, C. H. Chen ,J. D. Lee, "Genetic-Based Adaptive Fuzzy Controller for Robot Path Planning" Proc. 5-th int. conference on Fuzzy Systems, Vol. 3,pp.1687-1692,New Orleans, Louisiana, Sept. 1996.

[5] D. Kang, H. Hashimoto and F. Harashima, "Path Generation for Mobile Robot Navigation using Genetic Algorithm", Proc. Of IEEE IECON, First Int. Conf. on Industrial Electronics, Control and Instrumentation, Vol. 1, pp. 167-172, 1995.

[6] J. Xiao, Z. Michalewicz, L. Zhang and K. Trojanowski, "Adaptive Evolutionary Planner/ Navigator for Mobile Robots", IEEE Trans. on Evolutionary Computation, Vol. 1, No. 1, April, 1997.

[7] C. Hocaoglu, A.C. Sanderson, "Planning Multiple Paths with Evolutionary Speciation", IEEE Trans. on Evolutionary Computation, Vol.5, No. 3, JUNE 2001.

[8] J. A. Sauter, R. Mattnews, H. V. Parunak and S. Brueckner, "Evolving Adaptive Pheromone Path Planning Mechanisms", Proc. of the first Int. Conf. on Autonomous Agents and Multi-Agent Systems AAMAS, 2002, Italy.

[9] J.TU, S.X. Yang, "Genetic Algorithm Based Path Planning for a Mobile Robot", Int. Conf. on Robotics & Automation, pp. 1221-1226, Taipei, Taiwan, Sep 14-19. 2003

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

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

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