Path Generation for Mobile Robot using Genetic Algorithm

student member Daehee KANG (Univ. of Tokyo)
member Hideki HASHIMOTO (Univ. of Tokyo)
member Fumio HARASHIMA (Univ. of Tokyo)

The shortest/optimal path generation is essential for the efficient operation of mobile robot. This paper will present an algorithm for global path planning to a goal with a mobile robot in an known environment. The algorithm makes use of the modified quadtree data structure to model the environment and uses a genetic algorithm to generate a optimal path for the robot to move. Actually the genetic algorithm consists of two stages, the first stage(named a minor league) checks if a chromosome is reachable to goal position or not, and makes the individuals evolve. And, the only reachable chromosome and the best individuals of them are transferred to the second stage(called a major league) and then are evolved. Finally, the best chromosome of individuals in second stage is survived, so that the optimal/shortest path is generated. It is shown that our proposed method can find out a optimal path very quickly through simulation results.

Key words: Genetic Algorithm, Optimal Path Planning, Quadtree data structure

1. Introduction

In recent, Autonomous Guided Vehicles (AGV) that exist in industry to transport materials or to clean a warehouse are almost wire guided. Such vehicles are able only to follow the determined or desired paths that form an unchanging network. However, it may be costly and disruptive to make the wire, and it can be very difficult to modify them following a change of environment. These limitations have restricted commercial applications of guided vehicles. We have developed a wireless vehicle system. The vehicle potentially has the ability to model its environment, to estimate its self-locations, and to generate and modify automatically its paths using multiple sensors. In this paper, we will present a path generation approach under an assumption that the environment model has been established already, that is, the environment is known. Previous research on the path planning can be classified as following one of two approaches: model-based and sensor-based. In general, the model-based approach (6)considers obstacle avoidance globally. It uses prior models to describe known obstacles completely in order to generate a collision-free path. In contrast, the sensor-based approach (5)(9) aims to detect and avoid unknown obstacles. It is formulated as a local path planning problem in the sense that local detours around sensed obstacles are generated while the robot attempts to follow a globally planned path. In this paper, a global path planning as model-based is addressed, especially about the shortest path generation. Many methods have been reported to generate a optimal path. However, most of them require very large computing power in case of environments that have large number of subgoals and/or obstacles.

It is very important to generate the shortest path because of shortening the movement time and saving the energy of mobile robot. However, the most of actual mobile robot require an ability of real-time path planning rather than the shortest path generation, because the amount of the saving or the shortening in shortest path is not so much than one of the near shortest path. Therefore, in here, we propose a new algorithm finding out a near optimal/shortest path very quickly by Genetic Algorithm for elevating the real-time ability. A genetic algorithm(GA) is a hill-climbing search method.
that finds near-optimal solutions. The "Fitness" of each member of the GA population is computed by an evaluation function that measures how well the individual performs in the search domain. The best members of the population are propagated proportionately to their fitness, while members of poorly performing individuals are reduced or eliminated completely. The algorithm may find the optimal solution, but is not guaranteed to do so, therefore, we call it a near optimal/shortest path generation. At first, two previous methods are reviewed:

- Global path generation by dynamic programming
- Distance transform method

After introducing the previous works, we discuss how to establish an environment database in section 2. And, a global path planner using genetic algorithm is proposed in section 3. The simulation results and their discussion are presented in section 4, including the comparison of our proposed GA method and the previous method. Finally, a brief summary and conclusion is given in section 5.

2. Review of previous work

The problem of finding optimal paths for mobile robots within environments cluttered with obstacles has been studied with much interest. The most typically used methods of them are the Dynamic Programming method (or Dijkstra's algorithm) and the distance transforms.

(2.1) Dynamic programming method Dynamic programming is a mathematical optimization technique used for making a series of interrelated decisions. In contrast to other mathematical programming technique, there does not exist a standard mathematical formulation of the dynamic programming problem. The dynamic programming method for searching the shortest path is as follow:

Assume that a start point is $n_1$, a goal point is $n_m$, and subgoal points is $n_i$. The path generation method is how to determine a sequence of subgoals picking out the subgoals from their set $n_i (i = 2, 3, \cdots, m - 1)$. In the case, the cost function of the decided paths can be calculated as Eqs.(1).

$$F(n_1, n_2, \cdots, n_m) = f_k(n_1, n_2) + \cdots + f_k(n_{m-1}, n_m) \cdots \cdots$$

where $f_k(n_i, n_j)$ is a distance/cost function between subgoals $n_i$ with $n_j$. Now, define another cost function $g_k(n_i)$ as distance of path from start point to subgoal $n_i$:

$$g_k(n_{i+1}) = \min \{ g_k(n_i) + f_k(n_i, n_{i+1}) \} \quad \cdots (2)$$

$$g_k(n_1) = 0$$

In order to solve the Eqs.(2), we must calculate all of the possible path and select one that has the smallest cost value. Consequently, very large computing power is required, especially in environments that have very many subgoals.

(2-2) Distance transforms The use of distance transforms to generate path-planning solutions for mobile robots was first presented by Jarvis and Byrne. This approach considers the task of path planning as finding paths from the goal location back to the start location. The path planning procedure covers the environment with a uniform grid and propagates distances through free space from the goal cell(cells are assumed to be 8 connected). 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 exits where there exist two or more cells to choose with the same least distance transform(refer to Fig. 1). In other words, if there is no downhill path, that is, if the start cell is on a plateau, then it can be concluded that there is no path from the start cell to the goal cell. Therefore, disadvantages of this method are the ambiguity and the inefficiency of a grid-based approach which leads to the costly computation of the distance transform through large areas of free space.

![Fig. 1. An example of distance transforms](image)

3. Establishing environment database

In order to solve the disadvantages of the distance transforms, we construct environment maps using a modified quadtree method. Quadtree is a hierarchical structure with grid-like structure to compromise between network/graph models and grid-based models. The grid-like structure of the quadtree simplifies path planning and allows sensor data to be readily included into the environment model. However, the quadtree...
overcomes the deficiencies of the grid-based models because the quadtree is adaptive to the clutter of the environment. The quadtree fragments into the smallest possible grid cells only along the boundaries of obstacles. Kambhampati and Davis comment on the savings of a quadtree representation compared to a grid-based representation. They showed that the number of leaf nodes in a quadtree is proportional to the sum of perimeters of the obstacles in the environment. For applying easily quadtree model to genetic algorithm, we modify slightly the quadtree method.

At first, we extract empty regions (to be sub-goals) which are not occupied by obstacles (Fig. 2-(a)), and establish a graph to represent a relationship of start position and goal point, such as Fig. 2-(b), after finding out neighboring subgoals in the quadtree. And then a database is constructed with a linked list, such as Fig. 3, using information of the graph. The database consists of two parts mainly:

- information part of the cell's self
- neighboring information

The information of the cell's self are its name, its located position, and a flag which is used to check the cell’s state. And, the neighboring information are a total number of neighboring subgoals/cells and their name and address/pointers. The smallest quadrant cell area in the quadtree is sufficiently small to fully accommodate the mobile robot.

4. Genetic algorithm for optimal path

Genetic Algorithms are general purpose, parallel search procedures that are based upon genetic and evolutionary principles. A genetic algorithm works by repeatedly modifying a population of artificial structures through the application of genetic operators. GAs are typically black-box methods that use fitness information exclusively, that is, they are not require gradient information or other internal knowledge of the problem. The goal in optimization is to find the best possible solution or solutions of a problem, with respect to one or more criteria. In order to use a genetic algorithm, we must first choose a suitable structure for representing those solutions. A genetic algorithm’s data structure consists of one or more chromosomes (usually one). A chromosome is typically a string of bits, so the term string is often used instead. Each chromosome (string) is a set of a number called genes. Genes occur at various positions or loci of the chromosome, and take on values called alleles. The biological term genotype refers to the overall genetic makeup of an individual, and in our case, a gene is a bit, a locus is its position in the string, an allele is its value (natural number), and length of a chromosome is variable.

This section firstly defines a gene and a fitness, and presents a genetic operator.

(4.1) Determination of a Gene

A subgoal has usually a lot of neighboring subgoals, refer to Fig. 4. We make a numbering system that the neighboring subgoals get a natural number increasing sequentially in clockwise direction which is started from a point “S” in Fig 4 which is indicating the prior subgoal. In here, define a Gene as a natural number \( g_i \), and a genotype as its string. With a value of the Gene \( g_i \), we select one of the neighboring as its next subgoal. That is, we calculate a remainder \( R \) by Eqs. (3), and choose the \( R \)’th path of subgoal as the next one.
Here, \( n_t \) is a total number of the neighboring subgoals. And, this operation is performed repeatedly. Therefore, a genotype can represent a global path from start point to goal position as a set of sequential numbers. For an example, in case that a genotype is \([3,4,1]\), sequence of subgoals is \(\text{start} \rightarrow p6 \rightarrow p5 \rightarrow \text{goal} \) in Fig.2-(b), therefore, its phenotype becomes to be \([p6,p5,\text{goal}]\). But it is a case of simple environment, our environments are not so simple usually. Also, we don’t know if the genotype is long enough to arrive at goal after starting from the start point or not. So, we can not decide the length of genotype at initial state. In order to solve the problem, we must define a chromosome as a ring type which length is variable(Fig.7). The chromosome is duplicated repeatedly to the genotype until arriving at goal position or arriving at subgoal with no other path. In other words, the genotype is generated by repeating a chromosome until meeting goal point (named feasible individual) or meeting dummy subgoal (named unfeasible individual) in Fig.2. At next generation the length of a genotype and a chromosome are regulated to be long enough to be able to express the path sequence as only one turn of the ring-type chromosome. So that, the length of genotype and chromosome are always regulated that the sequence from start point to goal can be represented as only one repetition of chromosome. Our genetic algorithm is composed of three operators (refer to Fig.6):

- Calculation of Fitness
- Selection
- Reproduction

(4.2) Definition of Fitness To optimize a structure using a GA, one must be able to assign some measure of quality to each structure in the search space. The fitness function is responsible for this task. Objective function in optimization problem often acts as a fitness function. In genetic algorithm, the fitness function rates individuals in the population: better individuals have better chances for survival and reproduction. Hence it is essential to define a fitness function which characterizes the problem in a perfect way. However, dealing with problems to include constrains, it is very careful to define the fitness because feasible and unfeasible individuals exist in coincidence: very often a population contains unfeasible individuals. Finding a proper evaluation measures for feasible and unfeasible individuals is of great importance, that is, it directly influences the outcome (success or failure) of the algorithm\(^{(2)(1)}\)

Our problem searching for the shortest path can be written as following.

\[
\text{optimize } f(\hat{G}), \quad \hat{G} = (g_1, g_2, \ldots, g_n) \in \mathbb{R}^n
\]

\[
\text{constraint : arrive at goal point} \quad (4)
\]

where \( \hat{G} \in \mathcal{F} \subseteq S \). The set \( S \subseteq \mathbb{R}^n \) defines the search space and the set \( \mathcal{F} \subseteq S \) defines a feasible space. Our search space consists of two disjoint subsets of feasible and unfeasible subspaces, \( \mathcal{F} \) and \( \mathcal{U} \), respectively (refer to Fig.5). We can deal with only feasible space because we are searching for a feasible optimal. However, some of unfeasible individuals may have the better information than feasible individuals. For an example (see Fig.5), at some stage of the evolution process, a population may contain some feasible and unfeasible individuals, while the optimum solution is marked by 'X'. One of the unfeasible individuals such as \( g_m \) in Fig.5 may be better than all of feasible individuals. Therefore, we need to define two Fitness functions as cost function:

\[
\text{Fit}_f \quad \text{Genotype) =} \quad \frac{1}{1 + \sum Length(Gene_i(x,y) - Gene_{i-1}(x,y))} \quad \text{(5)}
\]

\[
\text{Fit}_u \quad \text{Genotype) =} \quad \frac{1}{1 + Length(good(x,y) - Gene_n(x,y))} \quad \text{(6)}
\]

Here, \( \text{Fit}_f \) is a fitness for feasible individuals and \( \text{Fit}_u \) is for unfeasible individuals. Genotype = \{Gene_1, Gene_2, \ldots, Gene_n\} is a string/set of natural number. At initial stage, genotypes may not be able to generate paths to goal point, it means that the most of the initially generated path may not finish at goal point but arrive at cells with no other path (named dummy point). For solving this problem, our genetic algorithm consists of two stages (refer to Fig.6), the first stage(named a minor league) checks if a genotype is reachable to goal position or not, and makes the individuals evolved by Eqs.(6). And, the only reachable individuals of them and the best individual of them are
transferred to the second stage (called a major league) and then are evolved in the major league using Eqs. (5). Finally, the best one of individuals in second stage is survived, so that one near optimal/shortest path is generated.

The relationship of minor league and major league and their process are shown at Fig. 6. Consequently, the fitness function of Eqs. (5) is for a genotype reaching to the goal position and Eqs. (6) is a fitness function for a genotype that don’t arrive to goal position.

(4.3) Selection One tournament method is used as selection scheme in major league. The population numbers are divided into subgroups, and a genotype with the best fitness among the subgroups is selected for reproduction. The size of subgroups is four. And, in minor league, fitness-proportionate selection (named roulette-wheel selection) is applied. The individuals \( i \) are selected with a probability of selection \( p_s(i) \), according to the ratio of its fitness to the overall population fitness:

\[
p_s(i) = \frac{\text{Fit}_v(i)}{\sum_{j=1}^{n} \text{Fit}_v(j)}
\]

It is necessary for the best element of the population to survive from generation to generation. Hence, elitist selection is adopted, which copies the best element from the current population to next population if the best individual has not been transferred through the normal process of selection, crossover, and mutation.

(4.4) Reproduction After finishing the above selection stage, we go to reproduction stage and operate crossover and mutation. First, the newly selected genotypes are paired together randomly with some probability. Second, two integer positions "n" and "m" along every pair of genotypes is selected uniformly. Finally, based on a probability of crossover, the paired strings are exchanged over the range between position "n" and "m" as Fig. 7. Although the crossover operation is a randomized event, when combined with selection, it becomes an effective means of exchanging information and combining portions of good quality solutions. And then, we operate mutation such as Fig. 8. The mutation is simply an occasional random change of a genotype position, based on probability of mutation. Our mutation changes gene of a position to random number. This mutation operator helps in avoiding the possibility of mistaking a local minimum for a global minimum. The selection scheme, the crossover and the mutation are applied at minor and major league, respectively, but only with the different probabilities.

Finally, we terminate this search in meeting three stopping criteria:

- Stop after a period time which make sure that the best genotype string is always retained.
- Stop when the average fitness of Eqs. (5) is close to the best fitness.
- Maximum Number of Generation.

5. Simulation Results and discussion

We simulated our path generation method under an environment of Fig. 9. The environment is indoor with area of \((26m \times 26m)\), and the minimum size of cell for
quadtree method is (0.5m x 0.5m). 284 cells were extracted as result of quadtree method. For genetic algorithm, we set empirically the parameters as followings:

- Population size (number of genotype per generation): 40
- Tournament size: 4
- Probability of mutate for stage1: 0.1
- Probability of mutate for stage2: 0.0012
- Probability of crossover for stage1: 0.6
- Probability of crossover for stage2: 0.1

Fig. 11 and Fig. 10 show the average fitness and the best fitness in major league, respectively. The result paths are shown in a curve of Fig. 9. Its best fitness of a dotted line path in Fig. 9 is reached at 1710's generation, and its value is 0.000194, so that length of the best path is 46.37m. And Fig. 13 shows the change of populations in both leagues. The number of total population is constant. That is, if the number of population in major is 'n', minor league has populations of \( \text{populationsize}(40) - n \). Its ratio depends upon the probabilities of mutate and crossover in each stage. If the changing probabilities in minor is high while the probabilities in major is low, the number of populations in major becomes to be large because it is almost impossible for the populations in major to transfer to minor league. However, the speed of convergence becomes to be very high while it may be unable to search for global minimum point. By the way, if the probabilities of the mutate and the crossover in major league is high, the population size of major league is to be small. And the probability to converge to global minimum becomes to be high while the speed of convergence is low. So, we have to trade off the two probabilities. Fig. 12 is also the average fitness in minor league. The average fitness in unfeasible space shows the more violent change than in feasible space. Our simulation environment has at least possible paths of \( 2^{284} \) by simple calculation because it is possible to assume that each cell has at least two selectable paths. Therefore, it is impossible to select the optimal path because the general methods must check lengths of all path, \( 2^{284} \), and select one of the shortest. For another example, consider the case of TSP (Traveling Salesman Problem) similar with our problem searching the shortest path. When salesman must visit \( N \) villages, the number of possible paths that have to be found out for determining the shortest path is like as Eqs. (8).

\[
\text{Number of Paths} = \sum_{k=1}^{N-1} (N-1)C_k = \sum_{k=1}^{N-1} \frac{(N-1)!}{k!(N-1-k)!} \quad (8)
\]
Hence, assuming that \( N \) is an odd number, \( N - 1 = 2m \), it is required to find out \( (2m)!/(m!m!) \) paths at least. In case of \( N = 21 \), the number to search for paths becomes to be 923780, so that we can not find out the shortest path in the case of large number. They require very large computing power. Our method, however, searched for the path at only 1710 generations. Being compared with the general approaches, our method showed that the problem of optimal path generation could be solved very fast.

(5.1) Comparison with the GA method and Dijkstra's algorithm

It is said that the fastest of the Dynamic Programming methods for finding out the shortest path is Dijkstra's algorithm\((11)\). In this section, we compare our proposed GA method with the Dijkstra's algorithm under environments of various types. At first, we specify the environments with their complexity which is defined as following:

\[
Diff = \log \prod_i (N_i^{\text{en}} - 1), \quad i = 1, 2, \ldots, n, \ldots \ldots \ldots \ldots (9)
\]

where \( N_i^{\text{en}} \) is number of neighbors at \( i \)’th subgoal and \( n \) is a total number of subgoals in an environment. In here, we need to consider the meaning of the defined complexity, Eqs.(9), through example of environments. In Fig. 14, each subgoal has two neighboring subgoals, so that the complexity of the environment is "0". Of course, in this case, it is not necessary to consider a path planning because the number of possible paths is only one. The complexity also is to be large with the number of the possible paths. Fig. 15 is a simplified environment of a big super-market, its complexity is about "150". In case of departments to need a serving mobile robot, their complexity will be increased. Consequently, our \( Diff \) defined by Eqs.(9) can represent an environment complexity as a number. With the complexity value, let classify various environments, and then compare our GA method with Dijkstra's algorithm.

The proposed GA method and the Dijkstra's algorithm are tested changing the complexity from "15" to "190", and the results show in Fig. 16. Fig. 16 shows that the GA method becomes to be faster than the Dijkstra's beyond the complexity "50".

6. Conclusion

In this paper we presented a novel genetic algorithm to optimize or generate a path, and proposed and simulated it. Our method found out a near optimal/shortest path very quickly than conventional methods. And through simulation, we found that our method searches for the near-optimal path even case that the optimal/minimal path is not found. Future works will develop the faster GA path planner and extend the global path planner to real-time path planner.
References

(7) David E. Goldberg, " Genetic Algorithm", Addison-Wesley, 1989
(15) 姜 大熊, 植本 秀紀, 原島 文雄, " マルチセンサ情報に基づく移動ロボットの自己位置推定とナビゲーション", ロボティクス・メカトロニクス講演会 95, 1995 年, 6 月

Dahee Kang (Member) was born in 1960. He received B.E. degree in electronic engineering from Yonsei University, Seoul, Korea in 1983 and M.E. degree in electrical and electronic engineering from Korea Advanced Institute of Science and Technology (KAIST), Seoul, Korea in 1985. He is currently a doctoral student in University of Tokyo, Japan, and he is working for Daewoo Heavy Industries Ltd., Korea. His research interests are in mobile robot and its intelligent control. He is a member of the IEE of Japan and the Robotics Society of Japan.

Hideki Hashimoto (Member) was born in 1957. He is an associate professor at the Institute of Industrial Science, University of Tokyo. He received B.E., M.E., and Ph.D. degrees at Electrical Engineering from University of Tokyo in 1981, 1984 and 1987, respectively. He is currently working on control engineering and robotics. His main interest is in intelligent control system. He is a member of the IEE of Japan, the Society of Instrument and Control Engineers of Japan (SICE), the Robotics Society of Japan, and IEEE.

Fumio HARASHIMA (Member) was born in 1940. He has received B.S., M.E., and Ph.D. degrees at Electrical Engineering from University of Tokyo in 1962, 1964 and 1967, respectively. He has been a Professor at Institute of Industrial Science, University of Tokyo since 1980. His research interests are in power electronics, control and robotics. He has been active in various academic societies such as Institute of Electrical Engineers of Japan, Instrument and Control Engineers of Japan(SICE), Robotics Society of Japan and IEEE. He had served as President of IEEE Industrial Electronics Society and 1990 IEEE Secretary. He has received a number of awards including 1978 SICE Best Paper Award, 1983 IEEE of Japan Best Paper Award, 1984 IEEE Anthony J. Hornfeck Award and 1988 IEEE Eugene Mittelmann Award. He is also a fellow in IEEE and SICE.