AUTONOMOUS ROBOT NAVIGATION USING A GENETIC ALGORITHM WITH AN EFFICIENT GENOTYPE STRUCTURE

ADITIA HERMANU
Vico Indonesia
Project Engineering Dept.
Jakarta, Indonesia

KAVEH ASHENAYI
University of Tulsa
Dept. of Electrical Engineering
Tulsa, Oklahoma

THEODORE W. MANIKAS
University of Tulsa
Dept. of Electrical Engineering
Tulsa, Oklahoma

ROGER L. WAINWRIGHT
University of Tulsa
Dept. of Mathematical & Computer Sciences
Tulsa, Oklahoma

ABSTRACT

The goal for real-time mobile robots is to travel the shortest path in minimal time while avoiding obstacles in a navigation environment. Autonomous navigation allows robots to plan this path without the need for human intervention. The path-planning problem has been shown to be NP-hard, thus this problem is often solved using heuristic optimization methods such as genetic algorithms. An important part of the genetic algorithm solution is the structure of the genotype that represents paths in the navigation environment. The genotype must represent a valid path, but still be simple to process by the genetic algorithm in order to reduce computational requirements. Unfortunately, many contemporary genetic path-planning algorithms use complex structures that require a significant amount of processing, which can affect the real-time response of the robot. This paper describes the development of a genotype structure that contains only the essential information for path planning, which allows for more efficient processing. A genetic algorithm using this structure was tested on a variety of simulated navigation spaces and was found to produce valid, obstacle-free paths for most cases.

INTRODUCTION

Mobile robots are desirable for operations such as bomb disposal or hazardous material management, which would be potentially dangerous for humans. An important task for the robot is autonomous navigation, where the robot travels between a starting point and a target point without the need for human intervention. While basic information may be available to the robot about the navigation area boundaries, unknown obstacles may exist within the navigation area. A path between the starting and target points that avoids collisions with obstacles is said to be *feasible* - this is a path that lies within free space. Thus, robot navigation methods need to solve the *path-planning problem*, which is to generate a feasible path and optimize this path with respect to certain criteria. The work presented here is part of a larger project to build an autonomous path-planning robot. This research is motivated by earlier work in this field of interest (Geisler and Manikas, 2002) by the same research team. This paper presents the research and simulation results of a genetic algorithm based path-planning software.

GENETIC ALGORITHM TECHNIQUES FOR ROBOT PATH PLANNING

Robot path planning is part of a larger class of problems pertaining to scheduling and routing, and is known to be NP-hard. Thus, a heuristic optimization approach is recommended as shown by Hwang (Hwang and Ahuja, 1992). One of these approaches is the use of genetic algorithms. A genetic algorithm (GA) is an evolutionary problem solving method, where the solution to a problem evolves after a number of iterations (Mitchell, 1996). A genetic algorithm starts with a population of individuals (*chromosomes*). Each individual represents a possible solution for a given problem. For robot navigation, an individual may represent a path between the starting and target points. Each individual is assigned a *fitness value*, based on how well the individual meets the problem objectives. Using these fitness values, individuals are selected to be *parents*. These parents form new individuals, or *offspring*, via *crossover* and *mutation*. Parent selection, crossover and mutation operations continue for several iterations (*generations*) until the algorithm converges to an optimal or near-optimal solution.

Various genetic algorithm methods have been applied to the robot navigation problem. One approach is to combine fuzzy logic with genetic algorithms (Arsene and Zalzala, 1999; Kubota et al., 1999; Pratihar et al., 1999). In this approach, the genotype structure represents fuzzy rules that guide the robot navigation, so the genetic algorithm evolves the best set of rules. While this approach can produce a feasible path through an uncertain environment, the genotype structure becomes very complex, as it needs to represent a variety of fuzzy rules. Another approach is to use genotype structures that represent local distance and direction, as opposed to representing an entire path (Cazangi and Figueiredo, 2002; Di Gesu et al., 2000; Gallardo et al., 1998; Vadakkepat et al., 2000). While these are simple to process and allow for faster real-time performance, the local viewpoint of these methods may not allow the robot to reach its target. Some methods have relatively simple genotype structures that can represent feasible paths, but require complex decoders and fitness functions (Hocaoglu and Sanderson, 2001; Sugihara and Smith, 1997; Xiao et al., 1997). This can also affect real-time response. Our research has focused on improving the genetic algorithm performance by developing a more efficient genotype structure.

Row-wise Structure

Previous research by our group used a row-wise navigation model (Geisler and Manikas, 2002). Given a navigation environment that is modeled by N rows, a path in that environment is represented by a genotype with N genes. Each gene position (locus) corresponds to a row index, while each gene value (allele) corresponds to a column index within that row. For example, assume that we have the chromosome {3,3,5,1,2,6}. This genotype represents a path that starts in row 1, column 3 (1,3) and ends at row 6, column 6 (6,6). The intermediate points on this path are (2,3), (3,5), (4,1), (5,2), respectively. Fig. 1 shows the navigation along this path in the world space, where point (1,1) is assumed to be at the top left corner.

The direction information of a chromosome represents the intermediate steps, or vertices, of a path. However, sending the robot on a straight line directly from the 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 will cause problems if any adjacent cells that the robot traverses from one row to the next have an obstacle. A better approach is to split the diagonal path segment into a horizontal segment and a vertical segment, which will allow the robot to circumvent obstacles. Therefore, we added a direction bit to the chromosome structure to indicate the first direction that the robot will turn to proceed to the next vertex.

Assume that we have a segment that starts in row 1, column 2 (denoted by (1,2)) and ends in row 2, column 5 (2,5). If the direction bit is 0 (Fig. 2a), then the path segment is split into a vertical segment from (1,2) to (2,2) and a horizontal segment from (2,2) to (2,5). However, if the direction bit is 1 (Fig. 2b), then the path segment is split into a horizontal segment from (1,2) to (1,5) and a vertical segment from (1,5) to (2,5).

The genetic algorithm using this genotype structure was tested on a set of simulated navigation environments. While this approach is simple, we discovered limitations with the row-wise model. The row-wise model assumes that each path-segment will start and end in consecutive rows. Unfortunately, this may not result in feasible paths for some world spaces.

Row and Column Structures

In order to address the limitations of our previous research involving only the row-wise genotype structure, we decided to incorporate more flexibility into the navigation model by allowing the orientation to be either row-wise or column-wise. For row-wise orientation, the robot is assumed to travel in consecutive rows from top to bottom, while for column-wise orientation, the robot is assumed to travel in consecutive columns from left to right. Thus, we modified the genotype structure by adding an orientation bit to each chromosome: 0 = column-wise, 1 = row-wise.

The row-wise orientation follows the same rules as our previous row-wise genotype structure. However, for a column-wise chromosome, each locus corresponds to a column index, while each allele corresponds to a row index. Fig. 3 shows the navigation for the column-wise chromosome {3,3,5,1,2,6}. This genotype represents a path that starts in row 3, column 1 (3,1), and travels along intermediate points (3,2), (5,3), (1,4), and (2,5) to reach its target point in row 6, column 6 (6,6).

For column-wise orientation, the direction bit is interpreted as follows: Assume that we have a column-wise path segment as shown in Fig. 4. The segment starts in row 2, column 1 (2,1) and ends in row 5, column 2 (5,2). If the direction bit is 1 (Fig. 4a), then the path segment is split into a vertical segment from (2,1) to (5,1), and a horizontal segment from (5,1) to (5,2). Conversely, if the direction bit is 0 (Fig. 4b), then the path segment is split into a horizontal segment from (2,1) to (2,2) and a vertical segment from (2,2) to (5,2).

GA ELEMENTS

The genetic algorithm parameters were experimentally determined. Crossover rate is 0.7, while mutation rate is 0.07. The population size is 50. Termination occurs if there is no improvement in the best path after 30 generations, or if a maximum of 800 paths have been generated. The fitness function weights are $w_l = w_t = 2$, $w_f = 3$. This penalizes infeasible steps, since we want our path to be obstacle free. 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.

Fitness Evaluation

Each chromosome represents a path in the world space. Recall that the goal for autonomous robot navigation is to determine the shortest feasible path and traverse this path in minimal time. A path is evaluated by examining its segments. The total path length is the sum of its segments. If a segment intersects an obstacle, then this is called

an infeasible step. Thus, a path is obstacle-free only if all of its steps are feasible. Minimizing path length will minimize travel time; however, so will minimizing the number of turns required to traverse this path.

Given these criteria, our fitness function must consider the following fitness factors: feasibility, length, and number of turns. Equation (1) shows how to calculate the feasibility factor f_f. S is the number of infeasible steps in the chromosome, while S_{max} and S_{min} are the maximum and minimum number of infeasible steps for the population. The length factor f_l is calculated in a similar manner, as shown in Eq. (2), where L is the total length of the path segments. The number of turns factor f_t is also calculated in a similar manner, as shown in Eq. (3), where T is the total number of turns. Putting this all together, we get the fitness function f shown in Eq. (4). w_f , w_L and w_T are weights for feasibility, length, and number of terms, respectively. After the fitness value for all chromosomes in the population have been computed, Rank Selection (Mitchell, 1996) is used to determine the parent chromosomes that will be used for reproduction.

$$f_f = 1 - \frac{S - S_{\min}}{S_{\max} - S_{\min}} \tag{1}$$

$$f_l = 1 - \frac{L - L_{\min}}{L_{\max} - L_{\min}} \tag{2}$$

$$f_t = 1 - \frac{T - T_{\min}}{T_{\max} - T_{\min}} \tag{3}$$

$$f = 100 w_f f_f \frac{\left(w_l f_l + w_t f_t\right)}{w_l + w_t} \tag{4}$$

Crossover and Mutation

In our GA, two parent chromosomes are combined applying a single-cross-point, value encoding crossover. The crossover operator produces two offspring chromosomes with each crossover operation. This is achieved by using the gene information, which was not used to build the first offspring, in order to build a second chromosome.

For mutation, almost every operation that changes the order of genes within a chromosome or that changes a gene's value is a valid mutation operator. A random number is generated for each gene. If this number is less than the mutation rate, then the gene is mutated. If a gene is to be mutated, another 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 1 or 0, is assigned to *direction* and *orientation.*

SIMULATION RESULTS AND CONCLUSION

The genetic algorithm was applied to seven sample spaces to simulate the navigation process. These spaces vary in size and in obstacle configuration. Fifteen trials were run for each space. Table 1 shows the success rate for each data set comparing Geisler's approach (Geisler and Manikas, 2002) to our new approach. For each test set, our new GA had more success navigating the search space. Figure 5 shows examples of successful runs for test sets SPSet05, SPSet06, and SPSet07.

Our modification to the genotype structure to allow more options in path planning improves the success of robot navigation. While the success rate was high, there were still trials where our new GA failed to find a feasible path. We are currently exploring methods to combine row-wise and column-wise representations in the same genotype structure. Hopefully, this will improve the success rate of the navigation algorithm.

REFERENCES

Arsene, C. T. C., and Zalzala, A. M. S., 1999, "Control of Autonomous Robots Using Fuzzy Logic Controllers Tuned by Genetic Algorithms." *Proceedings of the 1999 Congress on Evolutionary Computation (CEC99)*, pp. 428-435.

Cazangi, R. R., and Figueiredo, M., 2002, "Simultaneous Emergence of Conflicting Basic Behaviors and Their Coordination in an Evolutionary Autonomous Navigation System." *Proc. 2002 IEEE Conf. on Evol. Comp. (CEC '02)*, pp. 466-471.

Di Gesu, V., Lenzitti, B., Lo Bosco, G., and Tegolo, D., 2000, "A distributed architecture for autonomous navigation of robots." *Proceedings Fifth IEEE International Workshop on Computer Architectures for Machine Perception, 2000.*, pp. 190 - 194.

Gallardo, D., Colomina, O., Florez, F., and Rizo, R., 1998, "A Genetic Algorithm for Robust Motion Planning." *11th Int. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pp. 115-122.

Geisler, T., and Manikas, T. W., 2002, "Autonomous Robot Navigation System Using a Novel Value Encoded Genetic Algorithm." *45th IEEE Int. Midwest Symp. on Circuits and Systems*, Tulsa, OK, pp. 45-48.

Hocaoglu, C., and Sanderson, A. C. 2001, "Planning Multiple Paths with Evolutionary Speciation." *IEEE Trans. Evolutionary Computation*, Vol. 5, pp. 169-191.

Hwang, Y. K., and Ahuja, N. 1992, "Gross Motion Planning - A Survey." *ACM Computing Surveys*, Vol. 24, pp. 219-291.

Kubota, N., Morioka, T., Kojima, F., and Fukuda, T., 1999, "Perception-Based Genetic Algorithm for a Mobile Robot with Fuzzy Controllers." *Proceedings of the 1999 Congress on Evolutionary Computation (CEC99)*, pp. 397-404.

Mitchell, M., 1996, *An Introduction to Genetic Algorithms*, MIT Press.

Pratihar, D. K., Deb, K., and Ghosh, A., 1999, "Fuzzy-Genetic Algorithms and Mobile Robot Navigation Among Static Obstacles." *Proceedings of the 1999 Congress on Evolutionary Computation (CEC99)*, pp. 327-334.

Sugihara, K., and Smith, J., 1997, "Genetic Algorithms for Adaptive Motion Planning of an Autonomous Mobile Robot." *Proc. 1997 IEEE Int. Symp. Computational Intelligence in Robotics and Automation (CIRA '97)*, pp. 138-143.

Vadakkepat, P., Tan, K. C., and Ming-Liang, W., 2000, "Evolutionary Artificial Potential Fields and Their Application in Real Time Robot Path Planning." *Proc. 2000 Congress on Evolutionary Computation (CEC00)*, pp. 256-263.

Xiao, J., Michalewicz, Z., Zhang, L., and Trojanowski, K. 1997, "Adaptive Evolutionary Planner/Navigator for Mobile Robots." *IEEE Trans. Evolutionary Computation*, Vol. 1, pp. 18-28.

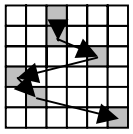

Figure 1. Row-wise navigation in world space.

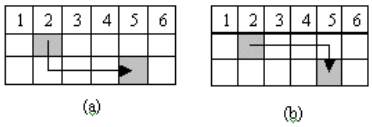

Figure 2. Navigation direction: (a) vertical first, (b) horizontal first.

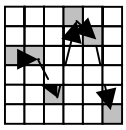

Figure 3. Column-wise navigation in world space.

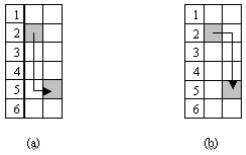

Figure 4. Column-wise direction: (a) vertical, (b) horizontal.

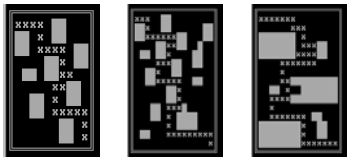

Figure 5. Navigation path examples for test sets (a) SPSet05, (b) SPSet06, (c) SPSet07.

Table 1. Comparison of methods.

Test Set	Success Rate (%)	
	Geisler's GA	Our new GA
SPSet01	0	87
SPSet02	47	100
SPSet03	73	87
SPSet04	87	100
SPSet05	53	80
SPSet06	0	93
SPSet07	0	100