Neural, Parallel & Scientific Computations 15 (2007) 279-298

A Specialized Genetic Algorithm for Optimum Path Planning of Mobile Robots

Qing Li[1], Guangjun Liu[2], and Yixin Yin[1]
[1]School of Information Engineering, University of Science and Technology Beijing
Beijing, China 100083
[2] Department of Aerospace Engineering, Ryerson University
Toronto, Canada M5B 2K3

Abstract
A specialized genetic algorithm for optimum path planning of mobile robots is presented in this paper. The proposed algorithm consists of an obstacle avoidance scheme that is introduced to generate efficient initial population and domain heuristic knowledge based crossover, mutation, refinement and deletion operators that are tailored to fit path planning for mobile robots. Furthermore, a fuzzy logic control approach is also incorporated into the proposed genetic algorithm to self-adaptively adjust the probabilities of crossover and mutation. Simulations on both off-line and on-line planning with static obstacles are carried out, and the proposed method is studied in comparison with another recently reported method. The results have demonstrated superior performance of the proposed method such as rapid search speed and high search quality.

Keywords – path planning, genetic algorithm, mobile robot

1. INTRODUCTION

The path planning problem of mobile robots is often treated as finding an optimum collision-free path from a start node to a goal node in an environment with obstacles. Generally, certain optimization criteria (e.g., shortest distance, minimum time or lowest energy consumption) must be satisfied under constraints such as limited velocity and acceleration or minimum turning radius. In this work, we focus on the investigation of optimum path planning based on a shortest distance criterion.

Optimum path planning for mobile robots has been a hotspot research area for many years, and several optimization methods, such as global C-space method (Lozano-Perez,

Received February 15, 2007

1983), potential field method (Rimon & Doditschek, 1992) and artificial neural networks approach (Yang & Meng, 2000), have been developed to tackle this problem. Two improved algorithms based on graph theory and artificial intelligence have been previously investigated by some of the authors of this paper (Li et al., 2005). Although each of the methods has its own advantages compared to other methods, the common problems associated with all the known methods are inevitable, e.g., heavy computation load, local optima and lack of self-adaptability. In the last decade, genetic algorithms have been widely used as an alternative methodology to generate optimum path for mobile robots. Researchers have fulfilled extensive analysis and simulation studies, and some promising results have been reported (Sugihara & Smith, 1997; Xiao et al., 1997; Tu & Yang, 2003; Hu & Yang, 2004). Sugihara and Smith (1997) propose a genetic algorithm with a fixed-length binary encoding scheme, and standard genetic operators (crossover and mutation operators) are adopted for recombination process. The method is suitable for both path planning and trajectory planning. However, its encoding mode is inefficient, and the generated optimum paths are restricted to X-monotone or Y-monotone. An adaptive evolutionary planner for both on-line and off-line path planning of mobile robots is reported by Xiao et al. (1997). Paths are encoded in variable-length strings, and specialized genetic operators with heuristic knowledge are devised for evolution process. Moreover, the probabilities of each operator can be self-adjusted online. Unfortunately, in practice, the implementation of the method is difficult due to its high computation overhead. Tu et al. (2003) introduce a genetic algorithm with variable-length binary chromosomes, and no more specialized genetic operators are designed for path planning. This method is capable of generating collision-free paths in both static and dynamic environments, but is also time consuming with its complicated encoding strategy. A knowledge based genetic algorithm is investigated by Hu et al. (2004), which features a simple yet efficient path encoding method, and genetic operators are designed to incorporate domain knowledge. While improved effectiveness and efficiency of the algorithm have been demonstrated by simulation results in comparison with other genetic algorithm based approaches, there are two problems associated with this method. One is that the initial population contains many infeasible paths, which has a negative influence to the performance of the genetic algorithm. The other is that parameters such as probabilities of crossover and mutation of the genetic algorithm are not adjusted during the evolution process, which limits the search speed for the global optimum solutions.

A specialized genetic algorithm for optimum path planning of mobile robots with static and dynamic obstacles in a known environment is proposed in this paper. The main contributions are as follows. First, an obstacle avoidance algorithm is introduced to generate the initial population so that all the individuals in the first generation are feasible, which dramatically improves the path planning efficiency. Secondly, a number of genetic operators (three crossover operators, one mutation operator, one refinement operator and

one deletion operator) are specifically designed for path planning problems of mobile robots. Thirdly, a fuzzy logic control algorithm is proposed for controlling the probabilities of crossover and mutation in the proposed genetic algorithm. The changes of the average fitness value and standard deviation between two consecutive generations are selected as the input variables. Two adaptive scaling factors are introduced for normalizing the input variables, and new domain heuristic knowledge based rules are developed for adjusting the probabilities of crossover and mutation. Simulations in both off-line and on-line situations are conducted, and the results have shown the effectiveness of the proposed algorithm. Comparative simulation studies are also carried out, and the results have demonstrated that the proposed method provides faster search speed and higher search quality.

The paper is organized as follows. The obstacle avoidance algorithm is introduced in Section 2, and a number of genetic operators are designed in Section 3. In Section 4, the adjustment algorithm is investigated. The simulation studies and results are presented in Section 5, followed by the conclusions and future work in Section 6.

2. OBSTACLE AVOIDANCE ALGORITHM

2.1. Environment representation and encoding scheme

It is well known that the environment representation and encoding scheme are among the key issues of path planning for mobile robots. As proposed by Hu and Yang (2004), the environment is represented by orderly numbered grids, as shown in Fig. 1.

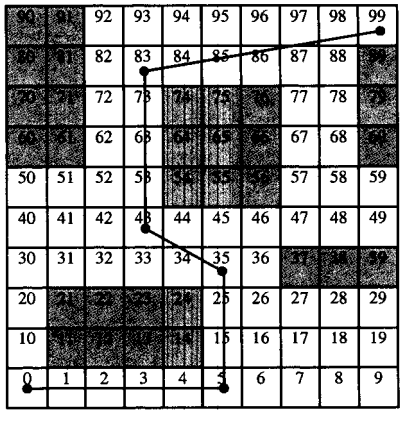


Fig. 1 **Environment representation and encoding scheme for path planning**

The blank grids stand for free areas where mobile robots can move freely and the shadow grids stand for obstacle areas, whose boundaries are formed by their actual boundaries plus a safety margin that is defined large enough so that the mobile robot moving in this environment can be treated as a point.

Therefore, a potential path is formed by line segments connecting the nodes falling on the grids with different numbers, and a path can be encoded as a sequence of grid numbers starting from the start node and ending at the goal node, with a number of intermediate nodes. For example, if node 0 is the start node and node 99 is the goal node, then the feasible path in Fig. 1 can be encoded as a string of numbers (0, 5, 35, 43, 83, 99).

2.2. Initialization of the population

2.2.1 Problems of random initialization

For all known genetic algorithms, the individuals of the first population are generated randomly. This is simple but will lead to large quantities of infeasible paths if used in path planning. Although some of these infeasible paths might become feasible solutions after certain genetic operations, they cause several problems. First, a penalty term is needed to append to the fitness function in order to distinguish feasible and infeasible paths, which will inevitably increase the computation load and decrease the search speed. Secondly, most of the genetic operations to the infeasible paths are meaningless. For instance, the crossover of two infeasible individuals can hardly generate a feasible one. Thirdly, some special genetic operators need to be designed for these infeasible individuals. For example, node-repair and line-repair operators (Hu & Yang, 2004) are specifically designed for repairing the infeasible paths.

2.2.2 The obstacle avoidance algorithm

An obstacle avoidance algorithm is proposed in this paper for generating the first population. The proposed algorithm consists of the following steps as illustrated in Fig. 2.

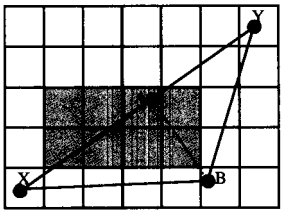


Fig. 2 **Diagram for the obstacle avoidance algorithm**

Step 1: Draw a line segment XY from the start node X to the goal node Y.

Step 2: If XY intersects an obstacle area O, continue; otherwise output XY as the shortest path.

Step 3: Select randomly one node A in the obstacle area O and on XY and then draw an orthogonal line (either right or left side) to XY from node A. If the orthogonal line intersects a free area F, select one node B in F randomly; otherwise stop.

Step 4: Connect nodes X and B, as well as nodes B and Y.

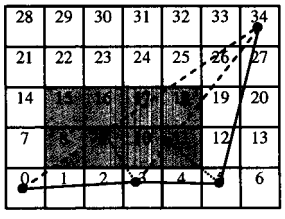
Step 5: Replace B with Y in line segment XB and B with X in line segment BY.

Step 6: Repeat steps 1 to 5 until none of the line segments from X to Y intersects any obstacle area.

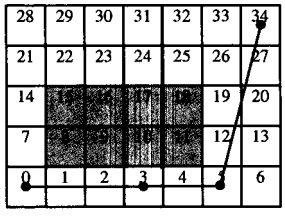
Step 7: A collision-free initial path is then generated and encoded by the series of grid numbers of the line segments from X to Y.

It should be noted that every orthogonal line from one obstacle areas should be drawn to the same side (right or left) as shown in Fig. 2.

Example: A 7×5 environment with one obstacle area, as shown in Fig. 3, is used to illustrate the generation process of initial paths. Here node 0 is the start node and node 34 is the goal node.



(a) Generation process



(b) The final path

Fig. 3 **Example of initial path generation with consideration of one obstacle area**

At first, a line segment between node 0 and node 34 is drawn, and it intersects with the obstacle area. Then node 9 is selected randomly and an orthogonal line to the right side from node 9 is produced. The orthogonal line intersects the free area at node 3. As the line segment between nodes 3 and 34 still intersects the obstacle area, node 3 is set as the start node and node 34 as the goal node, and the above process is repeated. After that, two line segments (one is between node 3 and node 5, the other is between node 5 and node 34) are generated, and they no longer intersect the obstacle area. Hence a collision-free initial path (0, 3, 5, 34) is produced, as shown in Fig. 3(b).

Different initial paths could be generated as there are various choices of intersection nodes in the obstacle area or free area and the side (right or left) of the orthogonal line.

If many obstacles exist in mobile robot environment, we can separate the whole generation process with multi-obstacles into several generation processes with one obstacle each so that we can apply the obstacle avoidance algorithm without any changes. Here is an example.

Fig. 4 is a 10×10 grid environment with five obstacles, with node 0 as the start node and node 99 as the goal node.

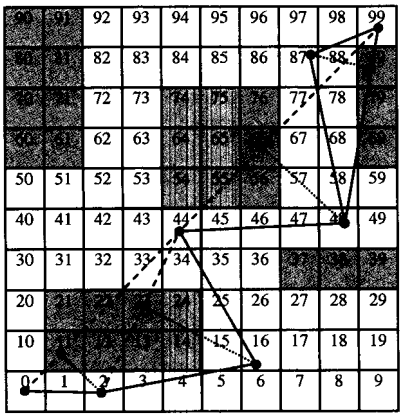


Fig. 4 **Example of initial path generation with consideration of multi obstacle areas**

At first, a line segment between node 0 and node 99 is drawn, and it intersects the free area at node 33 and node 44. Each node in the free area can be selected as the intermediate node, and suppose node 44 is selected here. Then we can use the obstacle avoidance algorithm to generate the first part of the initial path from the start node (node 0) to intermediate node (node 44), and the result is (0, 2, 16, 44). We can also use the above algorithm to produce the second part of the initial path from intermediate node (node 44) to the goal node (node 99) and the result is (44, 48, 87, 99). At last, the final initial path in multi-obstacle environment can be obtained via the connection of the separate parts of the initial path. In Fig. 4, the final initial path is (0, 2, 16, 44, 48, 87, 99). It is obvious that the third obstacle is avoided during the generation process from node 44 to node 99.

3. KNOWLEDGE BASED GENETIC OPERATORS

3.1. Crossover operators

Three knowledge-based one-point crossover operators are proposed in this paper to avoid the generation of infeasible paths and loops as in the one-point random crossover operation (Y. Hu & S. Yang, 2004). As described in the following, the proposed crossover operators are specifically designed for path planning of mobile robots.

3.1.1 Crossover at a common node

The proposed crossover operation at a common node consists of the following five main steps:

Step 1: Find all common nodes (the start node and the goal node are not included) between any two parent individuals.

Step 2: Select one common node randomly from those identified in Step 1 as the crossover point.

Step 3: If the contents before or after the crossover point of two parent individuals are the same, then cancel this crossover operation; otherwise, continue.

Step 4: Swap the contents after the crossover point of the two parent individuals.

Step 5: Check the offspring and if loops emerge, delete them, as proposed by Wu and Ruan (2004).

For example, suppose two parent individuals are V1(0, 30, 33, 47, 88, 99) and V2(0, 5, 35, 33, 83, 99). Obviously the two chromosomes have the common node 33. Select it as the crossover point, and the offspring individuals are V1'(0, 30, 33, 83, 99) and V2'(0, 5, 35, 33, 47, 88, 99).

3.1.2 Crossover at a potential node

A potential node appears in one individual (V1), but not in the other individual (V2). However, a line segment between two neighboring nodes in V2 passes the potential node. If such a potential node is identified, it is inserted into the individual V2. Then the potential node becomes a common node of the two individuals, and the crossover operation described in 3.1.1 can take place. For example, suppose two parent individuals are V1(0, 4, 15, 46, 47, 88, 99) and V2(0, 5, 35, 33, 83, 99). It can be seen that node 15 appears in V1, but not in V2. However, node 15 can be included in the line segment between two neighboring nodes (node 5 and node 35). Therefore, node 15 is a potential node. We insert node 15 into V2, and it is changed to (0, 5, 15, 35, 33, 83, 99). Although the genes of V2 are altered, the actual path is not modified. After that, we can take one-point crossover operation at the common node (node 15), and the offspring are V1'(0, 4, 15, 35, 33, 83, 99) and V2'(0, 5, 15, 46, 47, 88, 99).

3.1.3 Crossover at an interconnected node pair

An interconnected node pair refers to two different nodes (one in V1 and the other in V2), which can be connected without encountering obstacle. If the interconnected node pair is found between two parent individuals, it also can be treated as the common node, and the crossover operation in 3.1.1 can be taken. For example, suppose two parent individuals are V1(0, 4, 26, 46, 47, 88, 99) and V2(0, 5, 35, 33, 83, 99). Neither a common node nor a potential node exists between the two parent individuals. Node 46 (in V1) and node 35 (in V2) can be connected without encountering obstacle and can be

chosen as an interconnected node pair. After a one-point crossover operation similar to that for the common node case (now the interconnected node pair is treated as one node) is carried out, the offspring will be V1'(0, 4, 26, 46, 35, 33, 83, 99) and V2'(0, 5, 35, 46, 47, 88, 99).

3.2. Mutation operator

In order to increase the diversity of the population and avoid premature convergence, mutation operation is also essential in genetic algorithms. In conventional genetic algorithms, random mutation is the most common operation. However, random mutation operation can cause infeasible paths or loops easily in path planning. We propose a new mutation operator specifically for the path planning of mobile robots. The proposed mutation procedure consists of the following four steps.

Step 1: Select one node P (not the start node or the goal node) randomly from the mutation individual as the mutation gene.

Step 2: Define a set N that consists of all free nodes in a close vicinity to P (the set N can be stored in computer in advance).

Step 3: Choose one node Q randomly from the set N according to the forward direction of the path (the forward direction can be determined by coordinates' comparison of the start node and the goal node).

Step 4: If the node before (and after) P is connected with Q, then Q is applied to replace P in the new feasible path; otherwise repeat Step 3 until the desired node is found or the search process of set N is finished.

For example, suppose V(0, 4, 26, 44, 48, 98, 99) is the mutation individual in the parent generation and node 44 is selected as the mutation gene. We can easily obtain the set N (45, 35, 34, 33, 43, 53) from Fig. 1. According to the forward direction (here is northeast), only node 45 meets the demand, and it is connected with node 26 and node 48. Then node 45 is chosen to substitute the node 44, and a new path V'(0, 4, 26, 45, 48, 98, 99) is produced in the offspring' generation.

The small-scale local search technique is introduced in Step 2, and heuristic knowledge is also applied in Step 3 to increase the search speed and improve the search quality. Step 4 guarantees the feasibility of the new path after mutation operation.

3.3. Refinement operator

In order to reduce the distance of paths, a refinement operator is developed based on the principle that the distance of hypotenuse in a right triangle must less than the sum of the two sides that form the right angle. The refinement operation consists of the following six steps.

Step 1: Calculate the angle of two successive neighboring line segments of one path using their coordinates.

Step 2: If two line segments meet at a right angle, find the cross point of them and denote it as node I.

Step 3: Determine a node H that is before but adjacent to node I in one line segment.

Step 4: Determine a node J that is adjacent to but after node I in the other line segment.

Step 5: Insert node H before node I and put node J after node I in the refinement path, and delete node I.

Step 6: Repeat Steps 2 to 5 until all the line segments which cross at right angles have been disposed.

An illustrative example is given as follows. Suppose V(0, 5, 35, 33, 83, 99) in Fig. 1 is one individual in a parent generation to be refined and its distance is 21.08. We can find that there are four line segments forming three right angles at nodes 5, 35 and 43. After three refinement operations, the offspring individual is V'(0, 4, 15, 25, 34, 43, 83, 99) and its distance is 19.31. The refinement operation indeed decreases the distance of the path.

3.4. Deletion operator

The deletion operator is also proposed for reducing the distance of paths. The main idea is that, if two non-conterminous nodes in a path can be connected without encountering obstacle, the intermediate nodes between them can be deleted from the path.

For example, suppose V(0, 5, 15, 35, 33, 43, 83, 99) is one individual in a parent generation and its distance is 21.08. We can find that nodes 5 and 35, as well as nodes 35 and 43, can be interconnected without any obstacles. Therefore node 15 and node 33 can be deleted from the individual. After the deletion operation, the offspring is V'(0, 5, 35, 43, 83, 99), and its distance is 20.32. Clearly the deletion operation reduces the distance of the path.

4. ADJUSTMENT ALGORITHM FOR CONTROLLING THE CROSSOVER AND MUTATION PROBABILITIES

The probabilities of crossover and mutation have great influence on performance (e.g., search speed and search quality) of genetic algorithms, and the correct setting of these parameters is not an easy task. If the crossover and mutation probabilities are fixed, it is difficult to achieve an efficient balance between exploration and exploitation, allowing the well-performing operators to produce more efficient offspring while reducing the chance for the poorly performing operators to destroy the potential individuals (Yun and Gen, 2003). As a result, the search speed will be decreased. Also, fixed crossover and mutation probabilities are more likely to result in premature convergence and local optima.

A fuzzy logic based adjustment algorithm for controlling the probabilities of crossover and mutation is proposed in this paper. The detailed procedures are presented as follows.

4.1. Selection of input and output variables

It is well known that the mean and variance can depict the characteristics of a group of data comprehensively. These two variables are also utilized to evaluate the individuals of each generation in the proposed genetic algorithm. The average fitness value of the individuals in one generation (e.g., the kth generation) is defined as:

$$f_{avg}(k) = \frac{1}{M}\sum_{i=1}^{M} f_i(k) \tag{1}$$

where M is the population size, and $f_i(k)$ $(i=1,2,\ldots,M)$ is the fitness value of each individual.

The standard deviation of the kth generation is used to measure the diversity of the individuals in that generation. It is defined as:

$$\sigma(k) = \sqrt{\frac{1}{M}\sum_{i=1}^{M}[f_i(k) - f_{avg}(k)]^2} \tag{2}$$

However, these two variables can not reveal the trend of the evolution process. Therefore, the changes of the average fitness value and standard deviation between two consecutive generations are selected as the input variables to the proposed fuzzy controller in this paper. They are defined as follows:

$$\Delta f_{avg}(k) = f_{avg}(k-1) - f_{avg}(k) \tag{3}$$

$$\Delta\sigma(k) = \sigma(k-1) - \sigma(k) \tag{4}$$

The outputs of the fuzzy logic controller are selected as the changes of the crossover and mutation probabilities in the $(k+1)$th generation: $\Delta p_c(k+1)$ and $\Delta p_m(k+1)$.

4.2. Normalization of the input variables

If a fixed scaling factor is used for normalization, the input values of fuzzy logic controller will be very small and the probabilities of crossover and mutation usually do not change in a late stage, which has a negative influence to the performance of the genetic algorithm. In this paper, two adaptive scaling factors $\lambda(k)$ and $\eta(k)$ are introduced to normalize the input variables $\Delta f_{avg}(k)$ and $\Delta\sigma(k)$, respectively. For the minimization problems, $\lambda(k)$ is defined as:

$$\lambda(k) = \frac{1}{\max\{\Delta f_{avg}(k-1), \Delta f_{avg}(k)\}} \tag{5}$$

From Equation (5), we can see that the scaling factor $\lambda(k)$ is not a fixed value, and it will be adaptively modified in the evolution process. It can also normalize the input variable $\Delta f_{avg}(k)$ into the range of [-1.0, 1.0].

By the same token, the adaptive scaling factor $\eta(k)$ can be defined as:

$$\eta(k) = \frac{1}{\max\{\Delta\sigma(k-1), \Delta\sigma(k)\}} \tag{6}$$

4.3. Determination of the membership functions for input and output variables

The membership functions of the fuzzy input and output linguistic variables are illustrated in Fig.5. The only difference between the input variables and the output variables is their range. The input variables are normalized into the range [-1.0, 1.0], but the range of $\Delta p_c(k+1)$ is [-0.1, 0.1] and the range of $\Delta p_m(k+1)$ is [-0.01, 0.01].

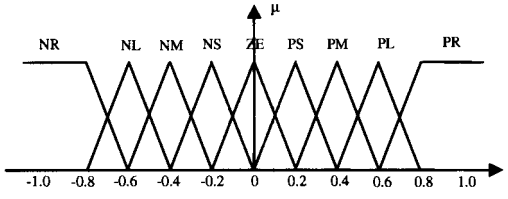


Fig.5 **Diagram of the membership functions**

The symbols in Fig. 5 are defined as: NR– Negative Larger, NL– Negative Large, NM– Negative Medium, NS– Negative Small, ZE– Zero, PS– Positive Small, PM– Positive Medium, PL– Positive Large, PR– Positive Larger.

4.4. Generation of fuzzy inference rules

Based on domain expert opinions, the fuzzy inference rules for controlling the probabilities of crossover and mutation can be summarized as follows (for the minimization problems):

(1) If f_{avg} is decreased, then increase p_c and p_m for the next generation; otherwise p_c and p_m should be decreased.

(2) If σ is increased, then increase p_c and decrease p_m for the next generation; otherwise p_c should be decreased and p_m should be increased.

(3) If the changes in f_{avg} and σ are very small, then increase p_c and p_m for the next generation rapidly.

The fuzzy decision tables for crossover and mutation probability from above rules are as shown in Table 1 and Table 2, respectively. In Table 1 and Table 2, the first row contains the language variables of $\Delta f_{avg}(k)$ and the first column represents the language variables of $\Delta\sigma(k)$.

Table 1 **Decision table for crossover probability**

	NR	NL	NM	NS	ZE	PS	PM	PL	PR
NR	ZE	PS	PS	PM	PM	PL	PL	PR	PR
NL	ZE	ZE	PS	PS	PM	PM	PL	PL	PR
NM	NS	ZE	ZE	PS	PS	PM	PM	PL	PL
NS	NS	NS	ZE	ZE	PS	PS	PM	PM	PL
ZE	NM	NS	NS	ZE	PM	PS	PS	PM	PM
PS	NM	NM	NS	NS	ZE	ZE	PS	PS	PM
PM	NL	NM	NM	NS	NS	ZE	ZE	PS	PS
PL	NL	NL	NM	NM	NS	NS	ZE	ZE	PS
PR	NR	NL	NL	NM	NM	NS	NS	ZE	ZE

Table 2 **Decision table for mutation probability**

	NR	NL	NM	NS	ZE	PS	PM	PL	PR
NR	NR	NL	NL	NM	NM	NS	NS	ZE	ZE
NL	NL	NL	NM	NM	NS	NS	ZE	ZE	PS
NM	NL	NM	NM	NS	NS	ZE	ZE	PS	PS
NS	NM	NM	NS	NS	ZE	ZE	PS	PS	PM
ZE	NM	NS	NS	ZE	PM	PS	PS	PM	PM
PS	NS	NS	ZE	ZE	PS	PS	PM	PM	PL
PM	NS	ZE	ZE	PS	PS	PM	PM	PL	PL
PL	ZE	ZE	PS	PS	PM	PM	PL	PL	PR
PR	ZE	PS	PS	PM	PM	PL	PL	PR	PR

4.5. Establishment of the look-up table

In order to increase the inference speed of fuzzy controller, look-up tables for crossover and mutation probabilities are set up, as shown in Tables 3 and 4.

Table 3 **Look-up table for crossover probability**

$\Delta p_c(k+1)$ ($\times 10^{-2}$)		$\Delta f_{avg}(k)$ ($\times 10^{-1}$)								
		-8	-6	-4	-2	0	2	4	6	8
$\Delta\sigma(k)$ ($\times 10^{-1}$)	-8	0	2	2	4	4	6	6	8	8
	-6	0	0	2	2	4	4	6	6	8
	-4	-2	0	0	2	2	4	4	6	6
	-2	-2	-2	0	0	2	2	4	4	6
	0	-4	-2	-2	0	4	2	2	4	4
	2	-4	-4	-2	-2	0	0	2	2	4
	4	-6	-4	-4	-2	-2	0	0	2	2
	6	-6	-6	-4	-4	-2	-2	0	0	2
	8	-8	-6	-6	-4	-4	-2	-2	0	0

Table 4 **Look-up table for mutation probability**

$\Delta p_m(k+1)$ ($\times 10^{-3}$)		$\Delta f_{avg}(k)$ ($\times 10^{-1}$)								
		-8	-6	-4	-2	0	2	4	6	8
$\Delta\sigma(k)$ ($\times 10^{-1}$)	-8	-8	-6	-6	-4	-4	-2	-2	0	0
	-6	-6	-6	-4	-4	-2	-2	0	0	2
	-4	-6	-4	-4	-2	-2	0	0	2	2
	-2	-4	-4	-2	-2	0	0	2	2	4
	0	-4	-2	-2	0	4	2	2	4	4
	2	-2	-2	0	0	2	2	4	4	6
	4	-2	0	0	2	2	4	4	6	6
	6	0	0	2	2	4	4	6	6	8
	8	0	2	2	4	4	6	6	8	8

4.6. Updating the probabilities of crossover and mutation for the next generation

The probabilities of crossover and mutation for the next generation are updated using the following equations:

$$p_c(k+1) = p_c(k) + \Delta p_c(k+1) \tag{7}$$

$$p_m(k+1) = p_m(k) + \Delta p_m(k+1) \tag{8}$$

5. SIMULATION STUDIES

5.1. Off-line path planning

A U-shape static obstacle in an environment with 16×16 grids is used as the off-line simulation example, as shown in Fig. 6. "S" is the start node and "G" stands for the goal node.

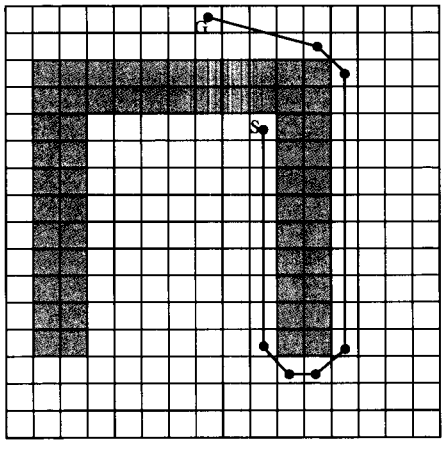


Fig. 6 **Optimum path planning in an environment with a U-shape obstacle**

The parameters of the genetic algorithm are set as follows: population size 50, maximum generation 50, initial crossover probability 0.5 and initial mutation probability

0.05. The fitness function is defined as

$$F(x_i) = \sum_{j=1}^{k-1} C_{j,j+1} \ (j = 1,2,\cdots,n) \tag{9}$$

where x_i represents the ith chromosome, $F(x_i)$ is the fitness value of the ith path, $C_{j,j+1}$ is the distance of the segment between node j and node $(j+1)$ in the simulation environment, k is total number of nodes in the ith chromosome and n is the population size. The best individuals will have the minimum fitness value. If the average fitness value does not change for 10 consecutive generations or the pre-defined maximum generation is reached, we consider the algorithm has converged.

At first, the obstacle avoidance algorithm is applied to generate the initial individuals. The tournament selection approach is adopted as the selection operator, and an elitism strategy is also used to preserve the best chromosome. Then crossover, mutation, refinement and deletion operations are carried out in order to obtain the optimum individuals. During the evolution process, the possibilities of crossover and mutation are adaptively adjusted using the algorithm in Section 4.

After 17 generations' iteration, the evolution process stopped, and the optimum path is generated as shown in Fig. 6, with a distance of 27.35.

The changes of parameters of each generation during the evolution process are listed in Table 5. "*Gen.*" stands for the number of evolution generations, and f_{avg} is the average fitness value, while f_{best} and f_{worst} are the fitness value of the best and worst individual of each generation separately. "*S.D.*" represents standard deviation of each generation, p_c is the crossover probability and p_m is the mutation probability.

Table 5 **The parameters of each generation during the evolution process**

Gen.	f_{avg}	f_{best}	f_{worst}	*S.D.*	p_c	p_m
1	36.43	30.01	40.00	3.29	0.50	0.100
2	33.87	29.69	36.82	2.73	0.50	0.100
3	31.64	28.83	35.89	2.49	0.54	0.108
4	31.76	28.51	34.96	2.08	0.56	0.112
5	30.03	28.06	31.72	1.35	0.52	0.122
6	30.42	28.06	32.30	1.70	0.58	0.122
7	28.66	27.89	29.89	0.67	0.58	0.118
8	29.32	27.35	31.31	1.27	0.60	0.126
9	29.13	27.35	30.96	1.24	0.60	0.120
10	28.27	27.35	29.30	0.72	0.60	0.124
11	27.58	27.35	27.92	0.23	0.62	0.134
12	27.61	27.35	28.83	0.88	0.64	0.140
13	27.61	27.35	27.92	0.44	0.62	0.138
14	27.60	27.35	27.92	0.31	0.60	0.140
15	27.63	27.35	28.83	0.58	0.60	0.146
16	27.62	27.35	27.92	0.66	0.56	0.144
17	27.57	27.35	27.92	0.45	0.56	0.146

5.2. On-line path planning

In Fig. 7, there are seven static obstacles in total (six of them are known, and one is unknown until it appears in the mobile robot's view, which is marked by its outline only). The optimum path (S, 23, 40, 199, 216, G) can be obtained in generation 62 when the proposed genetic algorithm is used for off-line path planning, as shown in Fig. 7, with a distance of 27.22.

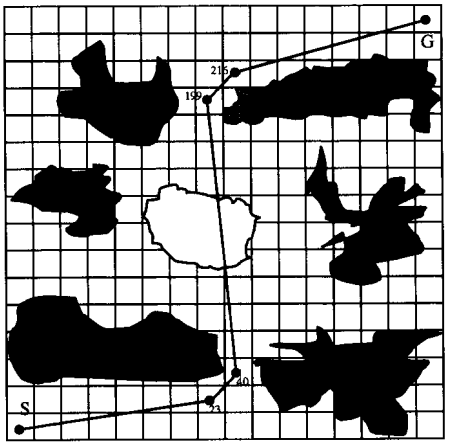


Fig. 7 **Environment with unknown obstacles**

When the mobile robot moves to gird 88 following the off-line optimum path, the originally unknown obstacle is in the robot's range of view and becomes a known one, as shown in Fig. 8.

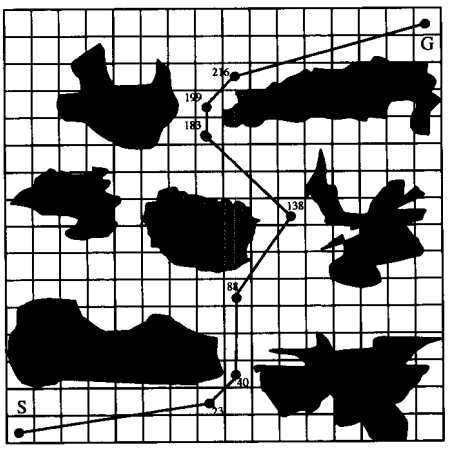


Fig. 8 **Environment for on-line path planning**

The information of obstacles will then be updated. It is clear that the original best solution is no longer feasible. So the genetic algorithm has to start a new evolution process to identify the best path from node 88 to the goal node. Certainly, the obstacle avoidance algorithm will be applied first to escape the newly known obstacle. After 55 generations, a new optimum path (S, 23, 40, 88, 138, 183, 199, 216, G) is generated, as shown in Fig. 8 and its total distance is 31.07.

Obviously it is desirable to sense unknown obstacles as early as possible, in order to stay close to the truly optimum path and to gain time for updating the path planning.

5.3. Comparative study

The proposed specialized genetic algorithm (PSGA) is compared with a recently reported method of similar type proposed by Hu and Yang (2004) (denoted as KBGA). The off-line path planning in Fig. 7 and on-line path planning in Fig. 8 are used for the comparative studies. Two indices are used to evaluate the performances of the two algorithms. One is distance of the optimum paths which represents the search quality of the 20 runs' evolution. The other is execution time of the evolution process, which exhibits the search speed. Here the number of generations as proposed by Hu and Yang (2004) is not adopted for measuring the search speed as every genetic algorithm has its own computation time for each generation, depending on its encoding scheme, initialization method, recombination operations and adjustment strategy for crossover and mutation probabilities. Therefore we consider that the execution time of the evolution process is more reasonable than the number of generations for evaluating the search speed of various algorithms. Table 6 lists the average distance and standard deviation (*S.D.*) results of the PSGA and KBGA for the above simulation environments with both off-line and on-line planning.

All the simulation programs are executed on an Acer notebook (AMD Turion 64, 512MB DDR) and programmed in MATLAB.

Table 6 **Comparison results of two algorithms**

Performance / Algorithm		Distance of the optimum paths		Execution time of the evolution process (s)	
		Average	*S.D.*	Average	*S.D.*
Off-line	KBGA	28.15	0.63	2.44	0.68
	PSGA	27.35	0.39	1.38	0.36
On-line	KBGA	32.13	0.86	2.25	0.65
	PSGA	31.27	0.75	1.22	0.37

From the results listed in the above table, we can see that (1) shorter average distance and lower standard deviation are achieved with the PSGA compared to those of KBGA; and (2) PSGA also requires less computation time for getting the optimal solutions than that required by KBGA.

With the PSGA, faster search speed is achieved by using both the obstacle avoidance algorithm and fuzzy logic based adjustment approach. The former ensures the feasibility of all individuals in the first generation, which results in computation time saving for calculating the fitness function in the proposed genetic algorithms. The latter guarantees more appropriate probabilities for crossover and mutation operators online, which leads to more balanced exploration and exploitation in the evolution process. Although the obstacle avoidance algorithm and fuzzy logic based adjustment add some computation load, they dramatically decrease the total number of evolution generations required. As a result, the execution time of PSGA is less than that of KBGA.

The individuals generated by the obstacle avoidance algorithm produce better descendants than that of randomly initialized. The refinement and deletion operators have reduced the distance of the paths efficiently. The fuzzy logic adjustment algorithm prevents premature convergence of the PSGA to local optima. Therefore, more optimum paths can be located by PSGA compared with KBGA.

In order to evaluate the effectiveness of the adjustment algorithm for crossover and mutation probabilities, the comparative simulations are carried out in two different situations. In the first case, the crossover and mutation probabilities are adjusted while they are fixed in the second case. The crossover probability is set to 0.8 and the mutation probability is equal to 0.4, which is the best value selected after running the proposed genetic algorithm for 100 times with various fixed crossover and mutation probabilities from 0.1 to 1.0.

The off-line path planning in Fig. 8 is selected for the comparative studies, and the results are illustrated in Figs. 9 and 10. In order to show the statistic characteristics of the two different cases, 20 runs were executed.

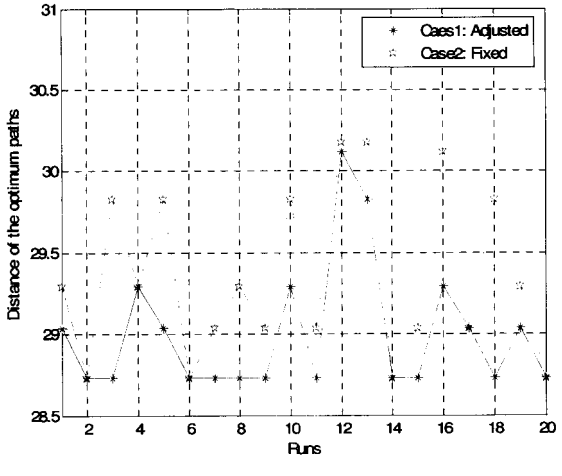


Fig. 9 **Distance of the optimum paths**

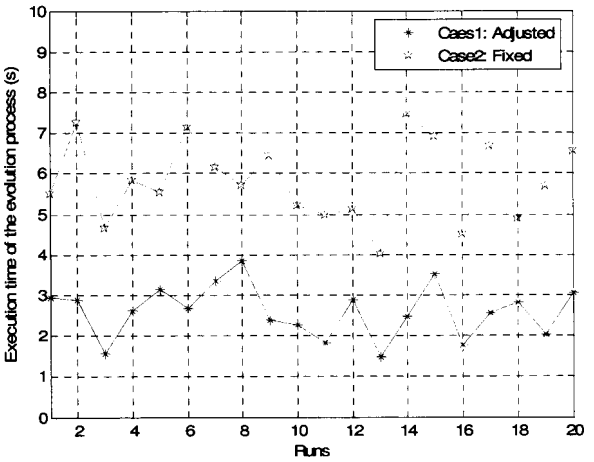


Fig. 10 **Execution time of the evolution process**

By comparing the results of case 1 and case 2 in Figs. 9 and 10, we can see that the adjustment algorithm for controlling the crossover and mutation probabilities has the following two functions:

(1) It can enhance the search speed significantly because the execution time in case 2 (with fixed probabilities) are deteriorated dramatically compared with case 1.

(2) It can improve the search quality. The distance of the optimum paths in case 1 is much shorter than that in case 2.

6. CONCLUSION AND FUTURE WORK

In this paper, a specialized genetic algorithm for off-line and on-line path planning of mobile robots is proposed. In order to increase the efficiency of the evolution process, an obstacle avoidance algorithm is investigated to generate the initial population so that all individuals in the first generation are feasible. Some specialized genetic operators including three crossover operators, one mutation operator, one refinement operator and one deletion operator are tailored to the optimum path planning for mobile robots. In order to improve the performance of proposed genetic algorithm, a fuzzy control strategy is also developed to self-adaptively adjust the probabilities of crossover and mutation. The changes of the average fitness value and standard deviation between two consecutive generations are used as the input variables, while the changes of the crossover and mutation probabilities are the output variables. Two adaptive scaling factors are introduced for normalizing the input variables and new domain heuristic knowledge based rules are integrated to adjust the probabilities of crossover and mutation. The

simulation studies on both off-line and on-line planning with static obstacles have shown the effectiveness of the proposed method. The comparative studies with a recently reported method are carried out and the simulation results demonstrate that the proposed algorithm has provided faster search speed and better search quality.

There are at least two tasks to be performed in the near future:

(1) In simulations, more complicated and generic environments with moving obstacles and uncertainty are to be applied to test the generality of the proposed algorithm; and

(2) The proposed genetic algorithm is to be used in actual systems to verify its feasibility.

ACKNOWLEDGMENT

This work is supported in part by NSFC (National Natural Science Foundation of China) Grant #60374032 and #60604002, CSC (China Scholarship Council) and in part by CRC (Canada Research Chair) program.

REFERENCES

1. Lozano-Perez, T. (1983). Spatial planning: A configuration approach. IEEE Trans. on Computers, v. C-32, 108-120.
2. Rimon, E., & Doditschek, D. E. (1992). Exact robot navigation using artificial potential fields. IEEE Trans. on Robotics and Automation, v. 8, 501-518.
3. Yang, S., & Meng, M. (2000). An efficient neural network approach to dynamic robot motion planning. Neural Networks, v. 13, 143-148.
4. Li Q., Song, D., Zhang, S., Li, Z., Liu, J., & Wang, Z. (2005). Two improved optimum path planning algorithms. Journal of University of Science and Technology Beijing, v. 27, 360-370.
5. Sugihara, K., & Smith, J. (1997). Genetic algorithms for adaptive motion planning of an autonomous mobile robot. Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation, 138-143.
6. Xiao, J., Michalewicz, Z., Zhang, L., & Trojanowski, K. (1997). Adaptive evolutionary planner/navigator for mobile robots. IEEE Trans. on Evolutionary Computation, v. 1, 18-28.
7. Tu, J., & Yang, S. (2003). Genetic algorithm based path planning for a mobile robot. Proceedings of the 2003 IEEE International Conference on Robotics and Automation, 1221-1226.
8. Hu, Y., & Yang, S. (2004). A knowledge based genetic algorithm for path planning of a mobile robot. Proceedings of the 2004 IEEE International Conference on Robotics and Automation, 4350-4355.

9. Wu, W., & Ruan, Q. (2004). A gene-constrained genetic algorithm for solving shortest path problem. Proceedings of the 7th International Conference on Signal Processing, 2510-2513.
10. Yun, Y., & Gen, M. (2003). Performance analysis of adaptive genetic algorithm with fuzzy logic and heuristics. Fuzzy Optimization and Decision Making, v. 2, 161-175.