Design of Robot Controller Based on Improved Genetic Algorithms

Di Wang[1, a], Wei Dong[1, b]

[1]*School of Southwest Minzu University, Chengdu 610041, China.*
[a]*794008048@qq.com*,[b]*brant.dong@163.com*

Keywords: Improved Genetic Algorithm; Robots Controller; Elite Individuals

Abstract: To simulate robots and their environment, an improved genetic algorithm is implemented in robots controller, and finally succeeding in passing a random maze. Introducing elite individuals into genetic algorithms achieves quadratic inspiration. The experimental results show that the improved algorithm has faster search speed, compared with traditional genetic algorithm**.**

1. Introduction

Genetic algorithm is one of the best tools for solving optimization problems of multi-objective functions [1]. By implementing genetic algorithm in real-world applications, it is possible to solve the problem that traditional computing methods can hardly solve [2]. Genetic algorithm is a random parallel search algorithm based on natural selection and genetic theory. It regards the set of all solutions to the problem as a population, and the quality of solutions is getting higher and higher by constant selection, crossover, and mutation [3].

This paper used the improved genetic algorithm to design the robot controller, and applied to the virtual robot in the virtual environment. It can continuously learn and eventually adapt to the environment successfully.

2. Improved Genetic Algorithm for Designing Robot Controller

The robot controller navigates the robot through the sensor's instructions. As shown in Figure 1.

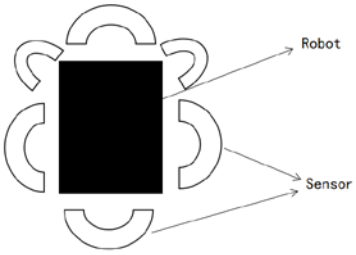

Figure 1 Robot and sensor

It is difficult to physical evaluate the robot controller shown in the Figure above, and the time required is very long. Therefore, the evaluation method for the robot controller in this paper is to

Published by CSP © 2018 the Authors
DOI: 10.23977/msmee.2018.72140

establish a simulation model of the physical robot and the environment (maze) so that the software rapidly evaluated each controller. [4, 5].

Each time the robot travels, the "wall" in the maze is fixed， and the six sensors will remain unchanged through the same position in the next time the robot travels. The problem we need to solve in this paper is how to make the right action based on the sensor's instructions when the robot is in a certain position.

2.1 Algorithm Structure

Based on the analysis of the design principle of robot controller, the algorithm flow in this paper showed in Figure 2.

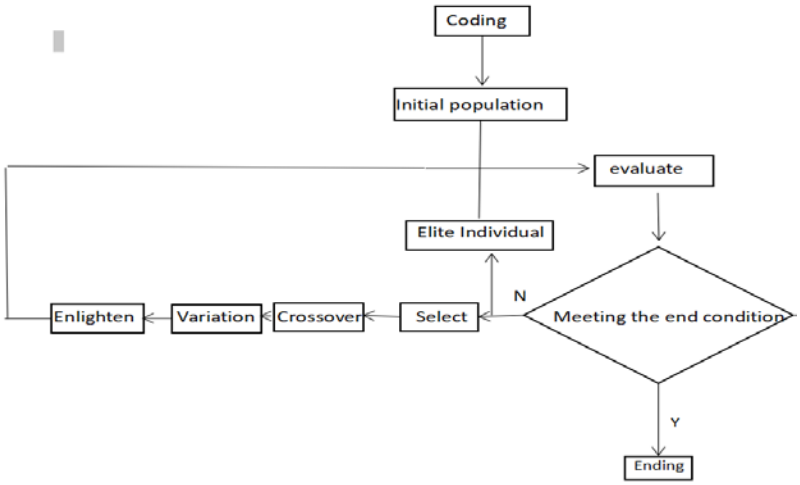

Figure 2 Process of improved genetic algorithm

2.2 Analysis of Algorithm Process

First, the four actions of the robot express in binary. "00" means nothing, "01" means move forward, "10" means turn left, and "11" means turn right. Since the maze is unknown, it is necessary to encode all possible instructions for the robot sensor. The instruction set of the robot controller represent in binary, and the sensor activated by "1" and "0" means that the sensor is not activated. The 64 binary instruction set converted to a decimal number corresponding to a digit from "0" to "63", so the sensor based instruction set encoded according to the "bit" action. The encoded chromosomes have "64" actions, each representing the robot controller's actions based on a sensor instruction. As shown in Figure 3.

00 11 10 11 01 00 11 01 00 11 ..... 00 11

Figure 3 One of chromosomes (64 groups).

In this paper, random methods are used to initialize the population, and each individual in the population will be randomly initialized, and each chromosome is composed of 128 genes randomly generated 1 or 0.

Every time the population evaluated, the most suitable individuals did not change to add directly to the next generation of population so that the optimal individual will not be lost in the generation of heredity. The evaluation population completed by walking through each individual and summing the fitness of each individual, and ranking each individual in the population based on the fitness degree in order to select "elite individuals".

The algorithm in this paper has two termination conditions. One is that the optimal path of the

robot in the maze terminated after fully explored. Another is the maximum number of generations (1000) that allows the algorithm to run is convenient for us to debug the algorithm. The two termination conditions are "or" relationships.

The "roulette wheel" method used to select parents to participate in crossover. See Figure 4. When the wheel turns and stops, a higher fitness experience is more likely to choose because they occupy more space on the wheel.

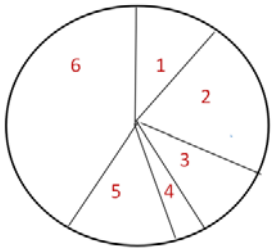

Figure 4 Roulette (population of only 6 individuals).

In this paper, we use the method of reverse realization roulette; we "mark" a position on the wheel, and then rotate the wheel to wait for "individual" to fall.

During the crossover process, individuals of the population exchange their genetic information to create a new individual, including the best part of their parental genes. This article quotes "elite" individuals to optimize the algorithm, always making the most suitable individual in the population unchanged and directly into the next population.

The crossover method used in this paper is a two-point crossover. As shown in Figure 5.

Patents1:	0	1	0	0	0	1	1	1
Patents2:	1	1	0	1	1	0	0	0
Offspring:	0	1	0	1	1	0	1	1

Figure 5 Intersection of two points.

In this paper, the mutation of "bit flipping" used to complete the variation of the individual. According to the initial value of the bit, "1" inverted to "0" and "0" inverted to "1". As shown in Figure 6.

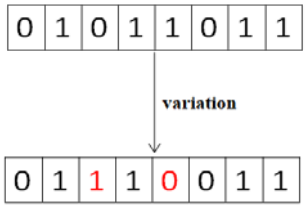

Figure 6 Variation

In the algorithm, update the mutation rate and the crossover rate to achieve the second heuristic. Starting with the higher ratio, the mutation rate and crossover rate gradually reduced as the algorithm carried out. The initial high mutation rate and crossover rate of the algorithm will cause the algorithm to find a large area of the search space.

3. Experimental analysis

Based on the above analysis, the algorithm designed and implemented in Java language.

3.1 Algorithm Pseudo Code

Input: maze, population number, mutation-Rate, crossover-Rate, number of elite individuals, cooling Rate.

```
// Create an algorithm and a maze instance:
1:Initialize Genetic Algorithm ( populationSize, mutationRate,crossoverRate, elitismCount, coolingRate);
2: Initialize Maze ;
// Initialize the population and run the robot:
3:population[generation]=initializePopulation(chromosomeLength);
4:Robot.run();
// Evaluation of population:
5:evaluatePopulation (population[generation], maze);
6:generation=1;
// Judge the termination condition if the conditional algorithm starts, otherwise the output result:
7:While isTerminationConditionMet()==false do;
// Obtain the best individual in this generation.
8:output the best individual in current Population;
//choose
9:parents=selectParents(population[generation]);
//cross
10:population[generation+1]=crossover(parents);
// variation
11:population[generation+1]=mutate(population[generation+1]);
// Evaluate and preserve fitness of individuals and populations, and run robots again.
12:evaluatePopulation(population[generation],maze);
13:Robot.run ;
14:generation++;
// Implement secondary inspiration
15:coolTemperature;
16:End loop;
```

Figure 7 Algorithm

3.2 Experimental Results and Analysis

Parameter Settings:

Population size: 150, crossover rate: 0.85, variation rate: 0.04, elite number: 2.0, cooling rate: 0.001, chromosome length: 128.

Maze object:

```
0, 0, 0, 0, 1, 0, 1, 3, 3, 2
1, 0, 1, 1, 1, 0, 1, 3, 1, 1
1, 0, 0, 1, 3, 3, 3, 3, 1, 0
3, 3, 3, 1 ,3, 1, 1, 0, 1, 1
3, 1, 3, 3, 3, 1, 1, 0, 0, 1
3, 3, 1, 1, 1, 1, 0, 1, 1, 0
1, 3, 0, 1, 3 ,3, 3, 3, 3, 1
0, 3, 1, 1, 3, 1, 0, 1, 3, 1
0, 3, 1, 1, 3, 1, 0, 1, 3, 3
1, 3, 3, 3, 3, 1, 1, 1, 1, 4
```

Figure 8 Maze

The maze object can be arbitrary, as long as it has a path that the robot will follow.In the maze of the experimental test, the correct position "3" has 34 places.

Multiple runs, randomly interception of three results, as shown in figure 9.(only the optimal individuals in the last two generations of evolution).

```
G 112 Best solution(31.0): 11 00 10 10 10 00 01 11 10 10 01 11 01 00 10 10 10 11 01 10 11 11 00 11 01 10 01 01 01 00 01 11
                           11 01 01 10 00 10 01 01 00 11 01 10 01 01 01 10 00 10 01 00 10 00 01 11 11 00 01 01 10 11 01 00
Stopped after 113 generations.
Best solution(34.0)        11 01 10 10 10 00 01 11 10 10 01 11 01 10 10 10 10 11 01 10 11 11 00 11 01 10 01 01 01 00 01 11
                           01 00 01 10 01 10 01 01 00 10 01 00 01 01 01 00 00 00 01 00 10 00 01 11 11 10 01 01 10 11 01 00
```

Figure 9 Quadratic heuristic genetic algorithm running result .

Set the cooling rate to 0 and randomly intercept three results, as shown in Figure 10.

```
G 808 Best solution(31.0):  10 11 11 01 10 10 01 00 10 10 01 01 01 11 10 10 01 01 01 01 11 11 11 11 01 01 01 10 01 01 01 10
                            11 01 01 01 00 10 01 10 10 10 01 11 01 01 01 10 00 10 01 10 01 01 01 11 10 11 10 10 01 10 01 11
Stopped after 809 generations.
Best solution(34.0)         10 11 01 01 10 10 01 00 11 10 01 01 01 10 10 10 01 01 01 01 11 11 11 11 01 01 01 00 01 01 01 10
                            00 01 01 01 01 01 01 10 10 00 01 11 01 01 01 10 00 10 01 10 01 11 01 10 10 11 10 10 01 10 01 11
```

Figure 10 The results of the genetic algorithm that failed to realize the secondary heuristic results .

Adjust the number of elite individuals and the cooling rate of two parameters, run the program several times, and take the average of the evolutionary algebra. The experimental comparison results show in table 1.

Table 1 Experimental comparison results.

Number Of Elites	Cooling Rate	Average Time
0	0	1000+
0	0.05	1000+
0	0.001	970
1	0	890
1	0.05	930
1	0.001	420
2	0	720
2	0.05	890
2	0.001	110

From table 1 can be seen, not to introduce elite individuals and unrealized secondary inspiration of traditional genetic algorithm need more evolution algebra to get the best individual and the algorithm may fall into local optimum and can't get the optimal solution. Thus, it can be seen that the introduction of elite individuals and the adoption of the second heuristic genetic algorithm will speed up the optimization and find the best solution in a faster and less time.

4. Conclusion

This paper uses the software to simulate the robot and its environment, thus saving time and avoiding the difficulty of artificial design. Using the improved genetic algorithm to design of robot controller, the introduction of the concept of "elite individual", and at the same time, we reduce two important parameters (cross-over rate and mutation rate)of each evolution to realize the secondary

heuristics. The experimental results show that the improved algorithm has faster searching speed. When the number of robot sensors increases, the coding task will become arduous, and the population size will become larger, which will result in more evolutionary algebra and longer running time, which should be improved in the future work.

Acknowledgments

This work was financially supported by the innovative project of the graduate student of the Southwest Minzu University in 2017 fund and the project number is CX2017SZ036.

References

[1] Shuguang Liu, Peiyan Fei and Zhimin Hou, Genetic algorithms in biological evolution and artificial intelligence (Study of natural dialectics.1999)
[2] David A. Grossman and Ophir Frieder, New algorithm and heuristic algorithms (Beijing People's post office.)
[3] Yingying Yu, Yan Chen, and TaoYing Li, Improved genetic algorithm for traveling salesman problem (Control and decision, 2014)
[4] Lee Jacobson and Burak Kanber, Java genetic algorithms programming (Beijing People's post office.2016)
[5] Ming Zhou, shuDong Sun and Yanwu Peng, Use genetic algorithm to plan mobile robot path.