UDK 004.896

V. Slavhorodskyi, S. Reznikov

STUDY OF THE GENETIC ALGORITHM CONFIGURATION FOR THE AUTONOMOUS ROBOT PATHFINDING PROBLEM

Annotation: The article is devoted to the development and improvement of the systems of rational routes of mobile robots that use a genetic algorithm. For the functioning of the mobile robot it is necessary to solve the problem of route formation. This task is global, contains many solutions and does not require absolute precision. To solve this problem, a genetic algorithm is suitable, it is a heuristic algorithm of global search. This algorithm operates with different genetic operators, such as crossover, mutation, selection, generation of populations and generations. These operators can be adapted to the need for route search. Thus, the genotype as a value that can operate on the algorithm can be represented as a route consisting of distance from each other waypoints. Then, with the help of genetic operators, routes are created and changed until a route is found, it should avoid various navigational hazards and satisfies the specified parameters. However, the search engine itself depends on the coefficients that determine the mode of operation of genetic operators. The effectiveness of this method of solving the navigation problem directly depends on selected factors that make solving the problem fast and reliable or completely effective the algorithm. Thus, before the direct application of the algorithm, it is necessary to identify the basic patterns between the coefficients used, as well as to determine their optimal values, which will be most effective for the algorithm.

Keywords: rational path planning, genetic algorithm, population size, crossover coefficient, mutation coefficient.

Introduction

Automated systems are inherently designed to reduce the number of errors made by a person navigating by assessing, predicting and assisting in decision-making at all stages of navigation [5]. One of the ways to implement such an automated system is a genetic algorithm (GA), which is a complex optimization algorithm based on evolutionary mechanisms in nature [2]. The method of adapting this algorithm in relation to navigation problems, as well as a description of its key features and terminology, are presented in the article [1]. The efficiency of this algorithm directly depends on the operated genetic parameters (genetic operators), which makes it impossible to assess the relevance of this method in isolation from the study of the effective impact of genetic operators to his job. Since the GA is identified with the evolutionary process in nature, a change in one or more of its operators inevitably affects the evolution that occurs in the algorithm cycle. The task of the algorithm in this study is to find the optimal route on the plane. Thus, the degree of influence of genetic operators on its work is determined by performance indicators, such as the length of the route, the time spent, the number of generations and the resources involved.

However, despite the numerous advantages of the genetic algorithm, its operation can hardly be called stable, given the influence of the random factor both in the generation of the initial population and in the application of crossover and mutation operators [1]. So, for example, in the article [4] it was revealed that the mutation operator in the composition of the GA can have a negative impact on the efficiency of the chosen method for solving the problem. Nevertheless, it is possible and necessary to make the operation of the algorithm more stable and reliable, as well as to increase its efficiency in relation to the task at hand, before implementing systems of this kind. In the article [1], the author emphasizes that the performance of the algorithm and its efficiency directly depend on its initial setting. Setting should be understood as the correct choice of coefficients for genetic operators in the algorithm, such as the crossover and mutation coefficients.

The purpose of this work: to clearly demonstrate the influence of various coefficients on the operation of the algorithm; identify the main trends in behaviour depending on various genetic operators and determine the degree of their influence; establish optimal intervals and combinations that can guarantee the efficient operation of the algorithm under the conditions of the task.

Description of the genetic algorithm and its parameters

For the purposes of this study test software was implemented using C++ programming language and QT - cross-platform library for user interface.

General scheme of genetic algorithm illustrated in Fig. 1. The first step is to create an initial population. A population (Y) of n chromosomes are initialized randomly code of this operation shown on Fig. 2. Each chromosome consists of gens. Gen represents a part of path. Gene stores path length and rotation angle using real numbers without encryption.

Class diagram illustrated at Fig. 3. shows relationship between Gen, Generation and Path classes. Each individual in a population is a solution to the problem you want to solve, in our case it`s path between points. This step could be parameterized with population size and number of genes in a chromosome.

Increasing population size will decrease the number of generations needed to achieve sufficient result, but it will also increase time spent on creation each generation. We will consider the result as sufficient if it differs from the optimal path by 10% or less. Graph that illustrates dependency between population size and number of generations illustrated in Fig. 4a and dependency between population size and computation time in Fig. 4b.

Selection operation based on fitness values selects individuals from the current population, they become parents for the next population and will generate offspring.

Rank selection first sorts the population by fitness and then every chromosome receives fitness from this ranking. The worst will have fitness 1, second worst 2 etc. and the best will have fitness N (number of chromosomes in population). After this, all the chromosomes have a chance to be selected. The probability that a chromosome will be selected is then proportional to

 ISSN 1560-8956

its rank in this sorted list, rather than its fitness. But this method can lead to slower convergence, because the best chromosomes do not differ so much from other one [3].

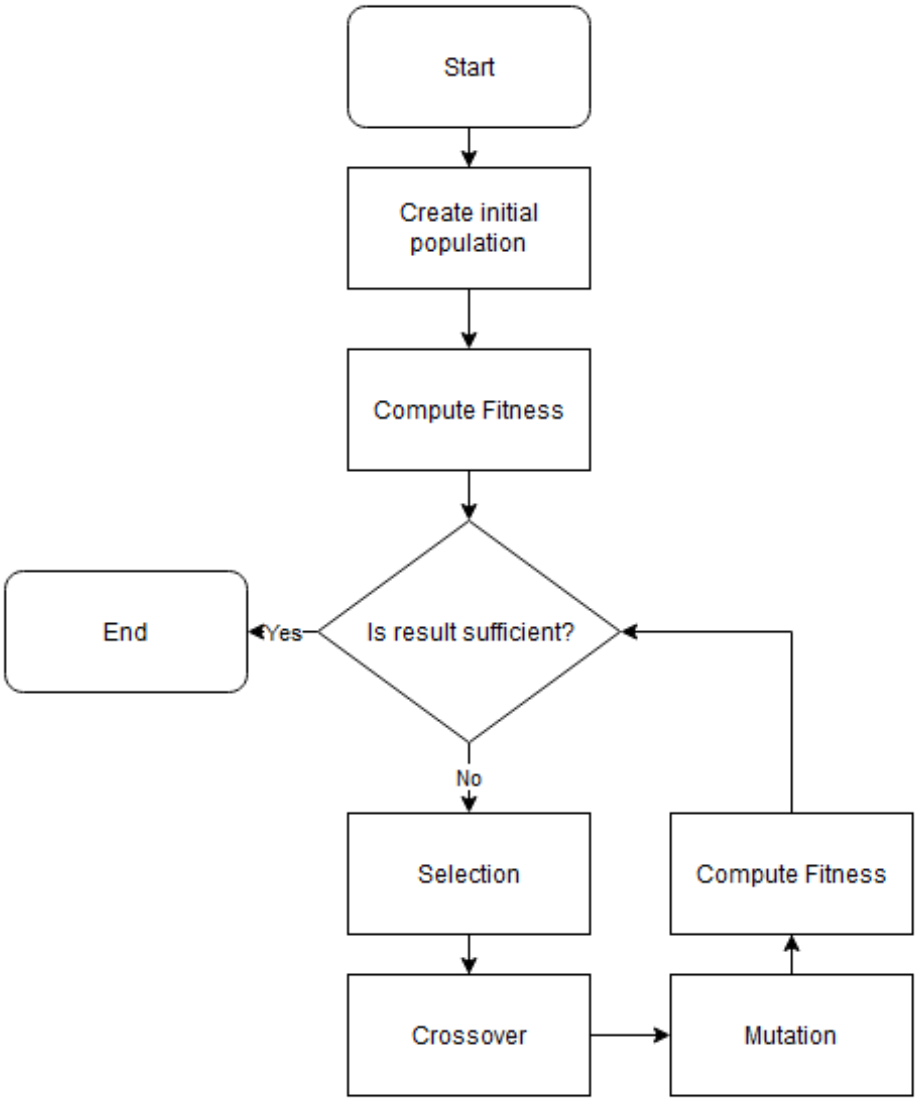

Figure 1. General scheme of genetic algorithm

```
Path GeneticAlgorithm::CreateInitialPopulation()
{
    Path path(m_gensLength);
    for (int gen_number = 0; gen_number < m_gensLength; gen_number++)
    {
        Gene gene;
        gene.m_path_length = (float)(GetRandomNumber() % MaxLength);
        gene.m_rotate_angle = (float)(GetRandomNumber() % GetAngleDiv());

        if (gen_number == 0) {...}
        else {...}
        path.m_gens_vector.push_back(gene);
    }
    return path;
}
```

Figure 2. Software implementation of creation initial population

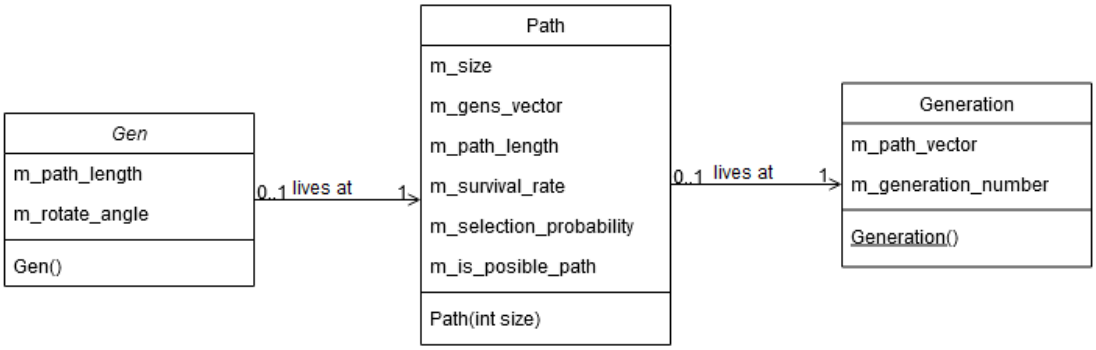

Figure 3. Dependency between Gen, Generation and Path classes

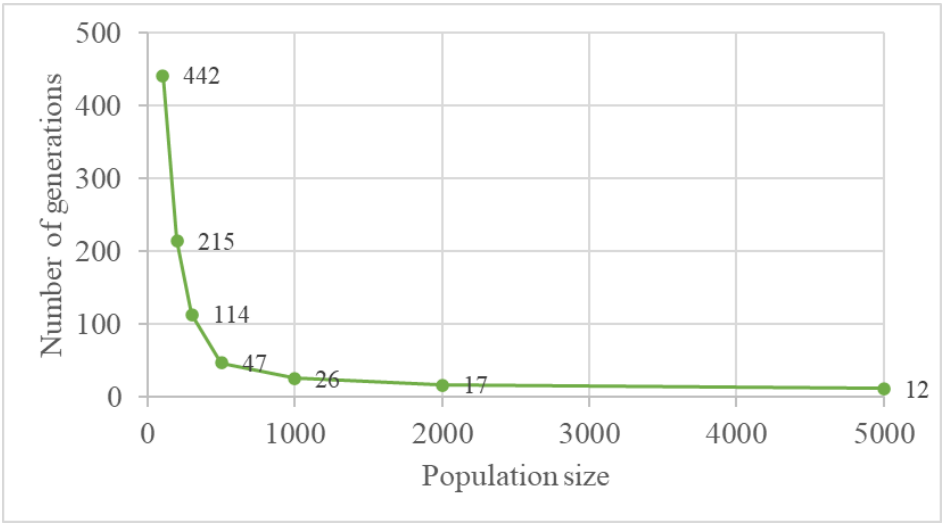

Figure 4a. Dependency between population size and number of generations

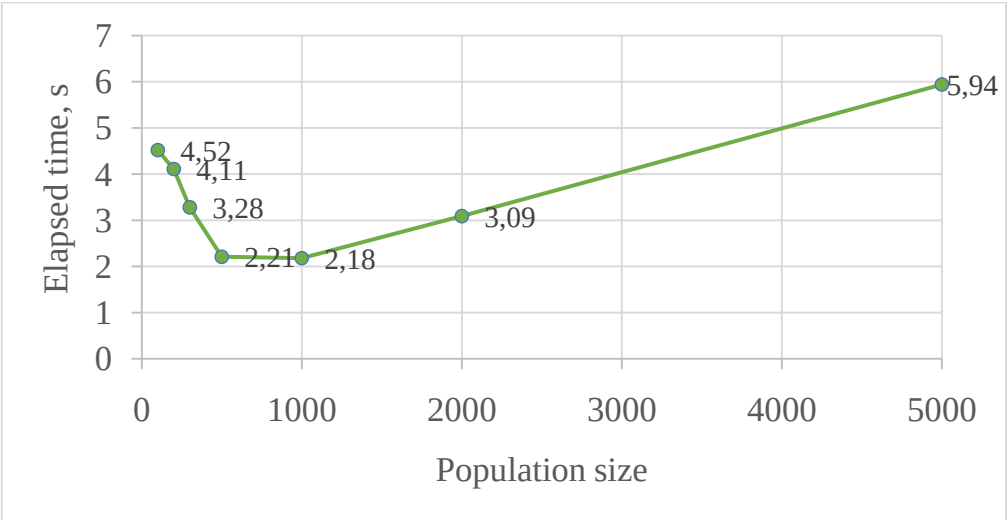

Figure 4b. Dependency between population size and computation time

In current implementation we used uniform crossover, which combines genes uniformly randomly from the two parents. Fig. 5 shows software implementation of uniform

 ISSN 1560-8956

crossover. Crossover coefficient *Kc* is the frequency of genes recombination and calculated according to the formula:

$$Kc = \frac{Pc}{P} * 100 \quad (1)$$

where *Pc* is the total number of recombinants in the population and *P* is population size.

The mutation is random, but with a given probability. The mutation coefficient *Km* sets the probability rate at which genes in a genotype will change. Fig. 5 also shows the software implementation of the mutation operator.

```
Path GeneticAlgorithm::UniformCrossover(Path first_parent, Path second_parent)
{
    Path newIndividual(m_gensLength);

    for (size_t i = 0; i < m_gensLength; i++)
    {
        if (GetRandomNumber(0,1) == 0)
            newIndividual.m_gens_vector.push_back(first_parent.m_gens_vector[i]);
        else
            newIndividual.m_gens_vector.push_back(second_parent.m_gens_vector[i]);

        if (GetRandomNumber(0, 100) <= PropobilityOfMutation)
        {
            newIndividual.m_gens_vector[i].m_path_length = (float)(rand() % MaxLength);
            newIndividual.m_gens_vector[i].m_rotate_angle = (float)(rand() % GetAngleDiv());
        }
    }
    return newIndividual;
}
```

Figure 5. Software implementation of uniform crossover operator with mutation

Results

Based on the averaged data of 600 trials with different algorithm configurations, the trends in the behaviour of the genetic algorithm were plotted depending on the crossover coefficients Kc (Fig. 6a) and the mutation Km (Fig. 6b).

Based on the graph on Fig. 6a, we can conclude the following:

- with the growth of Kc, the efficiency of the algorithm increases;
- in the range from 1 to 10% in terms of the crossover coefficient, the highest rate of change of the function is achieved - 56.2 generations per unit of the crossover coefficient, which indicates a high degree of influence that this coefficient has on the operation of the algorithm;
- in the range from 1 to 30%, the algorithm achieves the worst efficiency parameters relative to the remaining section of the graph (627.2 generations versus 188.0), which confirms the inoperability of the algorithm at low crossover coefficients, which, in turn, significantly complicate the evolutionary process;
- the function reaches a minimum at a value of Kc = 70%, which indicates the efficiency of the algorithm at high values of the crossover coefficient;
- in the range from 70 to 100%, the function begins to increase, which indicates a decrease in the efficiency of the algorithm at excessively high values of the crossover coefficient;
- in the range from 70 to 90%, the algorithm achieves the best efficiency parameters (138.5 generations) with random mutation rates.

The graph Fig. 6b shows the trend of the test results averaged over the crossover coefficient with a change in the mutation coefficient. So, we can conclude the following:

- with the growth of Km, the efficiency of the algorithm increases;
- in the range from 0 to 30% in terms of the mutation rate, the highest rate of function change is achieved - 14.1 generations per unit of the mutation rate, which is significantly inferior to the rate of change of the function from the crossing coefficient in the same section of the graph (24.6 generations), which indicates a lower the influence of the mutation rate on the operation of the algorithm;
- in the initial section of the graph, the function tends to reduce the number of generations, reaching its minimum at Km = 60%, after which it rapidly increases; this is especially noticeable in the area from 80 to 90% (9.7 generations per coefficient unit), which indicates a decrease in the efficiency of the algorithm at high mutation rates, which introduce a random nature into the evolutionary process;
- in the range from 50 to 70%, the algorithm achieves the best performance indicators (178.5 generations) with random crossover coefficients.

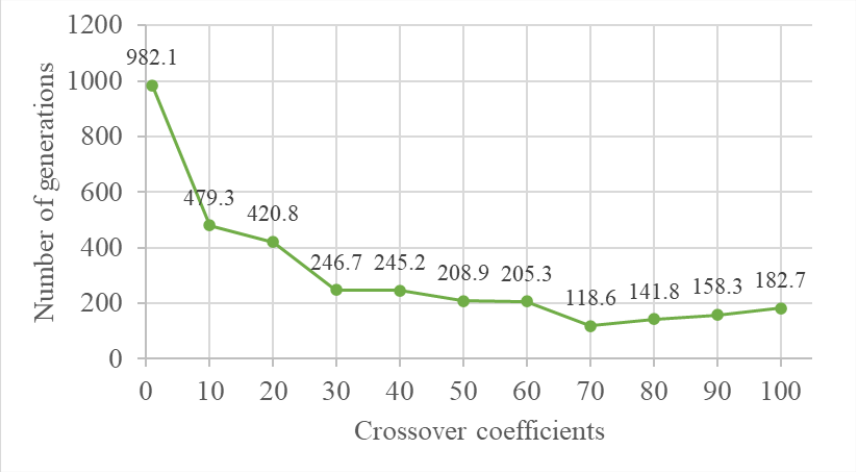

Figure 6a. Dependency between number of generations and crossover coefficients

Figure 6b. Dependency between number of generations and mutations

ISSN 1560-8956

Conclusions

On the example of solving the problem of finding the optimal route for mobile robots, it is clearly demonstrated the role of the main genetic parameters and the importance of the correct choice of coefficients. In the course of the study, it was possible to achieve the maximum efficiency of the algorithm in the problem posed by experimental enumeration of all possible coefficients, due to which the following results are obtained:

1) even with the correct implementation, the algorithm may turn out to be absolutely inoperable with thoughtlessly chosen coefficients;

2) the introduction of crossover and mutation operators has a positive effect on the operation of the algorithm;

3) the optimal intervals of values were identified that guarantee the efficient operation of the algorithm, taking into account the mutual influence: Kc = 70 ~ 100% and Km = 60 ~ 70%.

4) optimal combinations were identified that provide the best results: K60/50, K100/50, K70/60, K90/60, K80/70, K100/70

REFERENCES

1. Fedorenko K. V. Poisk optimal’nogo marshruta s primeneniem geneticheskikh algoritmov. //Materialy VII Mezhvuzovskoi nauchno-prakticheskoi konferentsii aspirantov, studentov i kursantov, SPb., 2017, p. 344–347.

2. Goldberg D. E., Holland J. H. Genetic algorithms and machine learning. //Machine learning vol.3, №2-3, 1988, p. 95–99

3. Haldurai L. A Study on Genetic Algorithm and its Applications. // International Journal of Computer Sciences and Engineering, 2016, p.140-142.

4. Kuznetsov Al. L., Kirichenko Al. L., Popov G. B. Chimerical genetic algorithm for sea route rationalization.// Vestnik Gosudarstvennogo universiteta morskogo i rechnogo flota imeni admiral S.O. Makarova, 2017, p.456–467.

5. Tsou M., Chao-Kuang H. The study of ship collision avoidance route planning by ant colony algorithm.// Journal of Marine Science and Technology №18, 2010, p.746–756.