Implementation of the genetic algorithm in an application for solving a traveling salesman problem

Student paper

Zinaid Kapić
Second cycle student
Technical faculty Bihać
Bihać, Bosnia and Herzegovina
zinaid_kapic@hotmail.com

Abstract— **The theme of this paper is the implementation of a genetic algorithm in an application that solves the problem of a traveling salesman. The problem for the traveling salesman is the problem of finding the optimal route and returning to the point of origin. Since this problem belongs to the category of NP-difficult problems and is of factorial complexity, it does not use traditional methods but rather heuristic methods that derive their logic from the behavior of nature and natural processes. A genetic algorithm is a heuristic method whose logic is based on the biological process of evolution. This paper first describes the travelling salesman problem and the genetic algorithm, and then presents an application that uses the aforementioned problem for finding an optimal route for tourists, based on the PHP and JavaScript programming languages, and uses Laravel as an environment.**

Keywords- Traveling salesman problem; Genetic algorithm; Application

I. INTRODUCTION

The Traveling Salesman Problem (TSP) is a discrete and combinatorial optimization problem. It belongs to the group of NP-heavy problems and its complexity is $O\ (n!)$. The problem of the traveling salesman is a problem of route optimization [1]. Each location represents a point in the route of the traveling salesman, and the aim of the traveling salesman is to visit each point only once, and eventually return to the point of origin. The origin of this problem dates back to the past. A similar problem, to the problem of the traveling salesman, was analyzed by Euler, and he was interested in having the knight on the chessboard visit all 64 places only once. The first literal account of the traveling salesman problem dates from 1832 in the book of German merchant traveler B.F. Voight stating that the most important point of travel planning is to determine the route without having to visit any of the cities twice. The general form of this problem appears in the 1930s. The term "traveling salesman" was first used in 1932 by Karl Menger. The first major traveling salesman problem that was solved was the problem of visiting 49 U.S. cities [2]. A traveling salesman may represent airplanes, ships, trucks, and may carry goods or passengers. With the development of technology, the transport problem of the traveling salesman has also arisen with the distribution of the Internet or in electronics when manufacturing electronic tiles with many nodes. Fifty years since solving the problem of traveling salesman in the example of 49 cities, in 2004, D. Applegate, R. Bixby, V. Chvatál, W. Cook and K. Helsgaun were able to find a solution for 24978 Swedish cities [3]. With the further development of technology and science, more efficient methods were developed and in 2006 the problem of traveling salesman was solved on a sample of 85900 cities [4]. The applicability of the traveling salesman problem lies in various optimization processes, logistics and routing [5].

Problems of combinatorial optimization, in addition to combinatorial nature, are often characterized by nonlinearity and multicriteria, which results in the mathematical modeling of a large number of problems extremely difficult and complex. Traditional methods are not used because of the factorial complexity of the problem. Heuristic methods are applied instead of traditional methods. Finding the optimal solution is often optional, so methods that find approximately optimal solutions are used, but their execution speed is much higher than traditional methods. Heuristic methods have logic based on the behavior of nature and natural processes [3].

One such algorithm is the genetic algorithm. Natural selection selects individuals that are able to adapt to the new environment. Individuals that have better environmental characteristics are more likely to survive and prolong their species. Less adapted individuals die out, and thus lead to the extinction of their species. The genetic algorithm represents the most popular type of evolutionary algorithms, and is an optimization model that bases its logic on the process of natural evolution [1]. These algorithms are based on Darwin's survival principle [3]. Genetic search algorithms do not start from a single point, but from a range of potential solutions. The potential solutions are mostly randomly generated, representing the initial population of the genetic algorithm. Each population has a certain quality that is also called fitness. Fitness is a measure of quality and is determined by the function of the goal, and tells you how good a solution is. After determining the quality, the genetic algorithm begins. The home population goes through generations, based on the principle of survival of

the best. Within each generation, the best solutions are chosen and the bad ones are rejected. The survivors of the selection are subjected to genetic operators. The first operator they undergo is a crossover, followed by a mutation. This principle creates new solutions that represent a new generation of genetic algorithm and these new solutions are better than old solutions. The creation of new individuals continues until the criteria for completing the genetic algorithm process are met. The solution found may not be optimal. The end of the genetic algorithm means that a good enough solution has been found, which may or may not be the best [3].

This paper presents practical implementation of an application that uses a genetic algorithm to solve a traveling salesman problem. The application was built using the Laravel framework and exploiting the capabilities of Google Maps.

II. APPLICATION DEVELOPMENT

An application called "Tourist TSP" is an application developed in the Laravel programming environment and adapted for mobile devices. The application solves the problem of a traveling salesman by finding the optimal route between locations. The mobile app is based on the capabilities of Google Maps. The flowchart of the application is shown in Fig. 1.

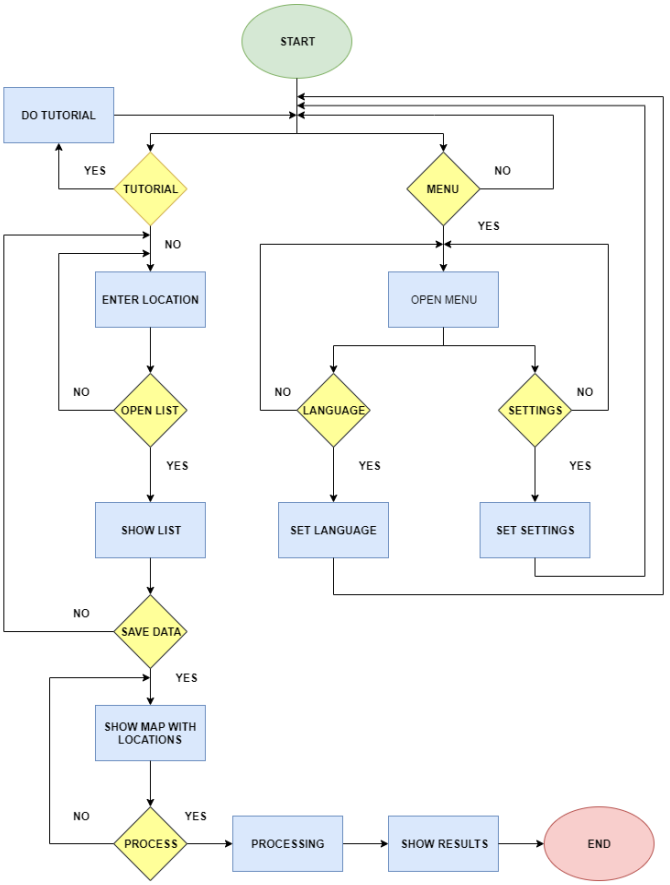

Figure 1. Application workflow diagram

Each user has the option of adjusting the parameters of the genetic algorithm. After entering the locations and saving them to the database, the parameters are used to determine the best route. The input, the rendering process and the final result are dynamically displayed on the application using Google Maps.

The design of the application is simple and user-friendly with a navigation menu that provides access to all pages of the application. Welcome page of an application is shown in Fig. 2.

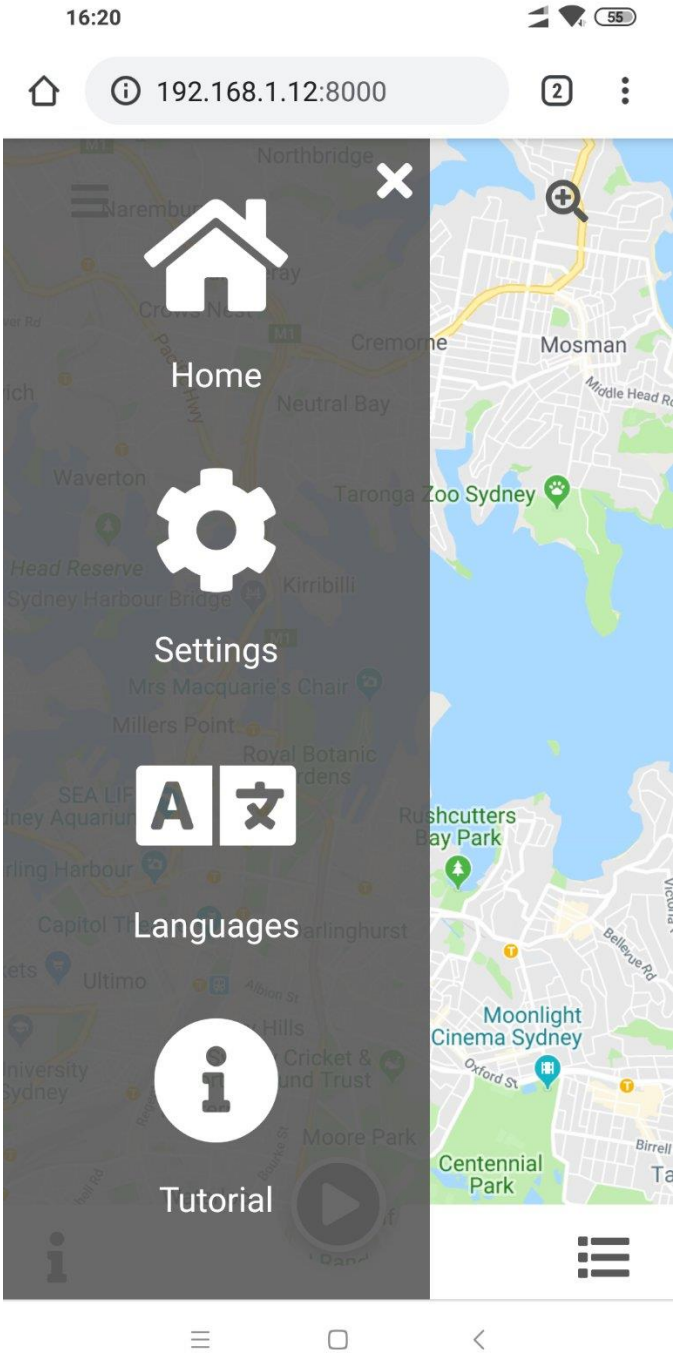

Figure 2. Application welcome page

User has an option to review app tutorial at every moment, to change language, adjust genetic algorithm parameters and to enter desired locations. Effect of every parameter will be explained through application testing.

Genetic algorithm pseudocode used in application is presented on Fig. 3.

```
GENETIC ALGORITHM
{
    t = 0;
    generate initial population of potential solutions P (0);
    determine fitness of P (0);
    while(condition of evolution process is not satisfied){
        t = t + 1;
        find parent P'(t) from P(t-1);
        recombine individuals from P'(t) and save children in P(t);
        mutate individuals P(t);
        find fitness P(t);
    }
write solution;
}
```

Figure 3. Genetic algorithm pseudocode

Genetic algorithm is consisted of steps shown in Fig. 3. Algorithm enters into a loop after determining initial population and calculating fitness function of this population. This loop repeats itself until condition of evolution process is satisfied. Through this loop, population is recombining and creating new children. These children are mutated and their fitness function is evaluated. If new children have fitness function that satisfies loop condition, genetic algorithm stops and writes solution.

Application interaction with Google Maps and database is presented on Fig. 4.

Figure 4. Application interaction with Google Maps cloud system and database

Application has an interaction with Google Maps in processes of retrieving desired locations through autocomplete search. Those locations are saved into database and retrieved from database for application processing and route displaying. Google Maps acts as a cloud server and provides services such as location autocomplete search, displaying location information, calculating location distances via Distance Matrix services, and plotting routes on the map. Searched information through Google Maps are saved into application database with location details such as latitude, longitude and location name. Those details are retrieved from database when user decides to determine a route. Application uses genetic algorithm and with information from database and with information gathered from Google Maps cloud server delivers plotted route on a map.

III. APPLICATION TESTING

The work of the application will be analyzed on the example of locations within Europe. The impact of genetic operators on mobile application processing results will be analyzed. Tested cities are:

- Madrid, Spain
- Paris, France
- Sarajevo, Bosnia and Herzegovina
- Ljubljana, Slovenia
- Berlin, Germany
- Warsaw, Poland
- Belgrade, Serbia
- Athens, Greece
- Milan, Italy
- Brussels, Belgium

Tested parameters are population, mutation and crossover rate. Parameters are set through user interface shown in Fig. 5.

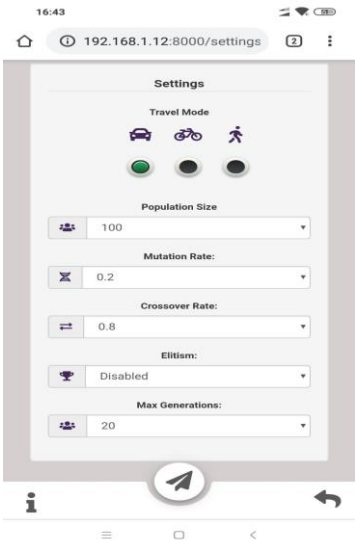

Figure 5. User interface for setting parameters of genetic algorithm

Travel mode used for these tests is by car. User has an option to adjust population size, mutation rate, crossover rate, elitism and maximum number of generations. Maximum number of generations is the number of loop iterations for genetic algorithm. These parameters are saved into database and used in application.

A. *Population impact on test results*

The parameters used to analyze the impact of the population on the test results are given in Table 1.

TABLE I. POPULATION PARAMETERS

Population	Mutation	Crossover	Elitism	Result
10	0.1	0.5	1	99.62 h
50	0.1	0.5	1	98.55 h
100	0.1	0.5	1	96.78 h

Population, mutation, crossover and elitism are parameters that are included in these tests. Using elitism in a genetic algorithm means that the best individual from the current generation will go directly to the next, without crossing and mutation, and that individual can further participate in the creation of children. The result of the application based on the presented parameters is shown in Fig. 6.

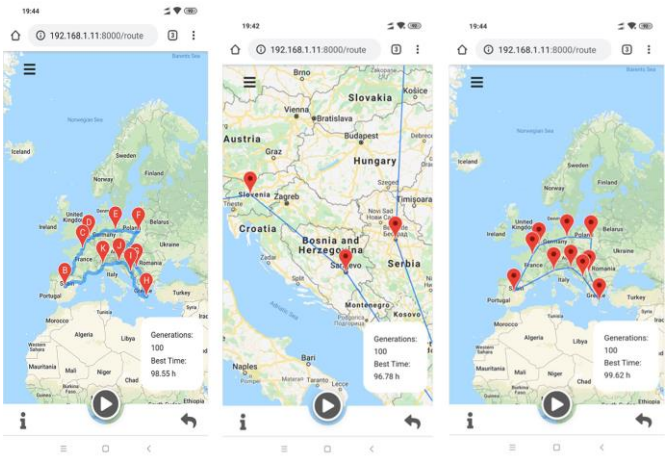

Figure 6. The operation of the application depending on the impact of the population

The tests conclude that population growth results in finding better solutions. As the population increases, the process is prolonged, so large populations are recommended only on larger samples, and samples of this application find a population value of 20 sufficient.

B. *Mutation impact on test results*

A mutation is defined in nature as a random change of a gene. The likelihood of these changes is not constant, so genes may be stable or unstable, or more or less susceptible to change. In a genetic algorithm, mutation is a genetic operator used to obtain next generation genetic diversity from the current one. If a binary representation of the solution is used, the mutation is the probability that some bit will change from zero to one or vice versa [6].

The parameters used to analyze the impact of the mutation on the test results are given in Table 2.

TABLE II. MUTATION PARAMETERS

Population	Mutation	Crossover	Elitism	Result
50	0.01	0.5	1	111.62 h
50	0.4	0.5	1	106.84 h
50	1.0	0.5	1	104.45 h

The result of the application based on the presented parameters is shown in Fig. 7.

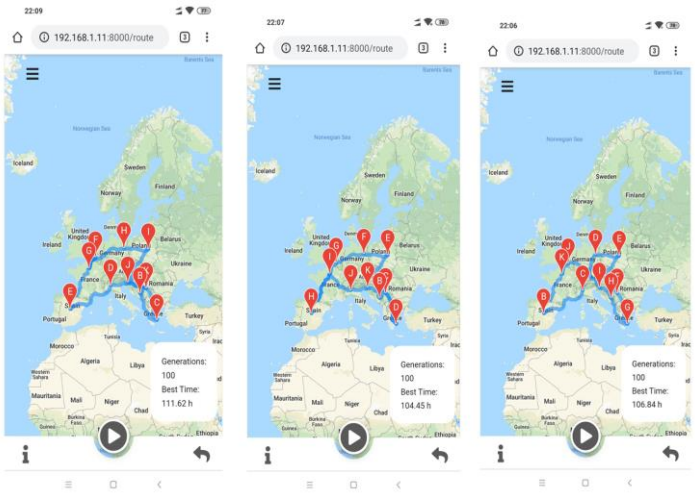

Figure 7. The operation of the application depending on the impact of the mutation

The optimal value of the mutation is 1, which means that the mutation occurs on each individual.

C. *Crossover rate impact on test results*

A crossover is the process by which a new single child is created from two parents. Crossing is often called recombination. This genetic operator has the feature that individuals born from crosses inherit the properties of the individuals from which they originated [6].

The parameters used to analyze the impact of the crossover rate on the test results are given in Table 3.

TABLE III. CROSSOVER RATE PARAMETERS

Population	Mutation	Crossover	Elitism	Result
50	0.4	0.2	1	111.60 h
50	0.4	0.8	1	110.88 h
50	0.4	1.0	1	107.94 h

The result of the application based on the presented parameters is shown in Fig. 8.

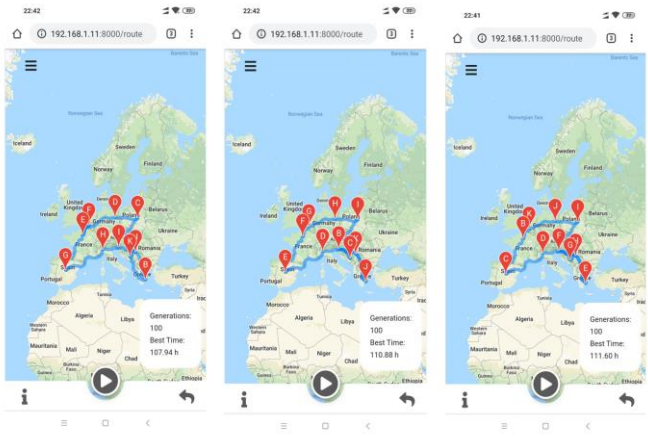

Figure 8. The operation of the application depending on the impact of the crossover rate

The influence of the crossing rate, as seen in Table 3, provides better solutions for the higher crossing rate.

IV. CONCLUSION

The applicability of the traveling salesman problem is in the various spheres of human interest. With the rapid development of technology and industry, science needed to find optimal solutions to the problems of optimizing resource consumption and optimizing profits. A traveling salesman also found a place in such an environment, which optimized the procedure for finding the optimal route. Because such problems require a great deal of processing power for computers, traditional methods have been replaced by heuristic and metaheuristic methods. These methods do not always produce optimal results, but in most situations small deviations from the optimal solution do not represent a significant problem, and high processing speed is the main reason for choosing such methods. One such method is the genetic algorithm.

Based on the analysis of the work of the application, it has been found that it is suitable for use in practice, and in most cases finds good solutions to the problem of traveling salesman. Adjusting the appropriate parameters results in approximately optimal solutions for a defined set of cities. The application is easy to use and user-friendly to simplify handling. Using heuristics in applications, business processes or even daily activities enhances the performance of those applications and processes.

REFERENCES

[1] D. Veljan, "Kombinatorna i diskretna matematika," Algoritam, Zagreb, 2001.

[2] M. Pielić, "Rješavanje problema trgovačkog putnika uz pomoć genetskih algoritama," Diplomski rad, Sveučilište u Zagrebu, 2008.

[3] D. Teodorović, "Transportne mreže," Univerzitet u Beogradu, 2007.

[4] D . Applegate, R. Bixby, V. Chvatál, W. Cook, "The Traveling Salesman Problem: A Computational Study," Princeton University Press, 2006.

[5] S. Pepić, Z. Lončarević, G. Miodragić, S. Aleksandrov, "Implementacja rešenja problema trgovačkog putnika genetskim algoritmima u Java programskog jeziku," ITOP17, 2017.

[6] M. Vištica, "Problem trgovačkog putnika," Diplomski rad, Sveučilište u Splitu, 2012.