Applied Computer Science, vol. 19, no. 2, pp. 55–62
doi: 10.35784/acs-2023-14

Submitted: 2023-04-02 | Revised: 2023-04-30 | Accepted: 2023-05-03

Keywords: evolutionary algorithms, genetic algorithms, traveling salesman problem, TSP

Tomasz SIKORA [0009-0009-2721-6796]*,
Wanda GRYGLEWICZ-KACERKA [0000-0003-4656-0540]**

APPLICATION OF GENETIC ALGORITHMS TO THE TRAVELING SALESMAN PROBLEM

Abstract

The purpose of this paper was to investigate in practice the possibility of using evolutionary algorithms to solve the traveling salesman problem on a real example. The goal was achieved by developing an original implementation of the evolutionary algorithm in Python, and by preparing an example of the traveling salesman problem in the form of a directed graph representing Polish voivodship cities. As part of the work an application in Python was written. It provides a user interface which allows to set selected parameters of the evolutionary algorithm and solve the prepared problem. The results are presented in both text and graphical form. The correctness of the evolutionary algorithm's operation and the implementation was confirmed by performed tests. A large number of tested solutions (2500) and the analysis of the obtained results allowed for a conclusion that an optimal (relatively suboptimal) solution was found.

1. INTRODUCTION

The subject of this paper is an implementation of an evolutionary algorithm, including all necessary genetic operators allowing for application of the algorithm to solving the traveling salesman problem non-trivial real-life cases (relatively difficult or impossible to solve with exact methods). Such cases do not need to concern hundreds or thousands of locations, but it should be possible to solve problems with a dozen or more than twenty locations. Such examples cannot be solved using exact algorithms due to excessively long computation time.

The traveling salesman problem can be represented as a problem of finding a cycle in a directed graph. In this paper, geographic locations of cities in Poland were used. In the mathematical sense, it can be stated that every two cities, i.e. vertices of the graph, were connected by two edges with the same weight-distance, but with opposite directions. In other words, there was a road from city A to city B, and from city B to city A, and the lengths of the roads were equal. It was assumed that the distance between cities is equal to the distance on the Euclidean plane. In reality this assumption is not met. However, with a large distance between cities (e.g. voivodships), the differences between the Euclidean distance and the real one will be negligibly small. Therefore, it can be concluded that the obtained results are close to reality.

* Akademia Humanistyczno-Ekonomiczna, Łódź, Poland, tomek.sikora@pm.me
** Akademia Humanistyczno-Ekonomiczna, Łódź, Poland, wgryglewicz@ahe.lodz.pl

For the purposes of this paper an application was written in Python. The application includes an implementation of an evolutionary algorithm together with a decoding procedure and all necessary genetic operators.

In addition, an example of the traveling salesman problem was developed based on the map of Poland. Sixteen voivodship cities were taken into account: Białystok, Bydgoszcz, Gdańsk, Katowice, Kielce, Kraków, Lublin, Łódź, Olsztyn, Opole, Poznań, Rzeszów, Szczecin, Warszawa, Wrocław, and Zielona Góra. It was assumed that there is a connection between each pair of cities, and the length of the connection was based on geolocation data of individual cities. Therefore, the problem solved was not simple and it can be predicted that the time of determining the optimal solution using exact methods (e.g., complete review, dynamic programming, or branch & bound methods) would be so long that the calculations would be ineffective (in terms of time waiting for the result). Hence it is a good example to be solved using heuristic algorithms, e.g., an evolutionary algorithm.

2. EVOLUTIONARY ALGORITHM FOR THE TRAVELING SALESMAN PROBLEM

The solution to the TSP problem can be written as a permutation of the first N natural numbers, where N is the number of vertices in the graph describing the problem. This means that each such permutation „codes" a solution, so it can be used as a genotype for an admissible solution of the TSP problem.

The advantage of such coding is that it will always yield an acceptable solution to the problem. Also, crossover and mutation operators operating on permutations are described in the literature.

A disadvantage of this coding method is the fact that many different permutations correspond to the same admissible solution. More important than the specific permutation is the relative arrangement of its elements. For example, in Fig. 1, two different permutations are shown that describe the same admissible solution to the TSP problem with ten locations. For N locations in the TSP problem, there will be exactly N different permutations that describe the same solution.

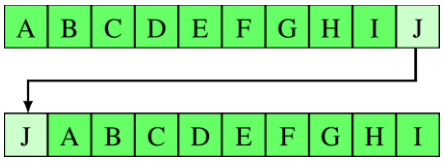

Fig. 1. Two permutations of one solution

2.1. Selection

Selection is a very important stage of EA. Without it EA would basically be quite a complex random algorithm.

The basic task of selection stage is the choosing individuals for the so-called reproductive pool. Individuals from this pool will be cross bred. Selection is based on the value of the so-called fitness function of the individual. Better adapted individuals have a greater chance (probability) to be selected for the breeding pool.

The selection can be done in many ways.

Relatively the most popular is the so-called roulette wheel selection. Each individual is assigned an area of the circle proportional to the value of its fitness function. The whole circle corresponds to the whole population. Choosing of an individual is based on a random selection of a point on the circumference of the circle: each point belongs to exactly one individual. The better-fitted individuals will have a better chance in such a draw, but even the least-fitted individuals can be selected.

This form of selection has some disadvantages. The size of the part of the roulette wheel corresponding to the value of the individual's fitness function is most often calculated according to the formula:

$$s_i = \frac{f_i}{\sum_{i=1}^{N} f_i} \tag{1}$$

where s is the size of the circle part of the i-th individual, and f_i is the value of the fitness function. This formula works under the assumption that all individuals have positive fitness values, and that the higher the fitness function, the better. When these assumptions are not met, fitness function values need to be properly scaled before the roulette selection is applied. Another disadvantage of this selection method is the fact that real values are used, on which operations are not always performed with high accuracy. However, this is more of an implementation problem.

Tournament selection is devoid of both disadvantages. It consists of a completely random selection of a certain number of individuals from the population, and then conducting a "tournament" between these individuals. The best adapted individual wins the tournament. Tournament selection is much simpler to implement and requires only comparison of the fitness function values of individuals.

Due to the greater genetic diversity, the project decided to use roulette wheel selection.

2.2. Crossover operators

A permutation can be defined as an ordered set of elements containing all the elements of another set. For a set with N elements, the number of different permutations is N!.

One point or two-point codings known from binary coding cannot be used for permutation coding. „Intersection” of two permutations at a randomly selected point (or points), and then exchanging parts of these permutations and thus creating an offspring will in most cases lead to the destruction of the permutation: the ordered result set will contain repeated elements, while other elements will not be present at all.

Permutation coding has the advantage that many crossover operators operating on permutations are known. The most known include:

- partially matched crossover (PMX) (Goldberg & Lingle, 1985),
- cycle crossover (CX) (Oliver, et al., 1987),
- order crossover (OX) (Davis, 1985).

In this research the order crossover (Fig. 2) was chosen due to its relatively simple implementation and high efficiency in the case of the TSP problem (Abdoun & Abouchabaka, 2011).

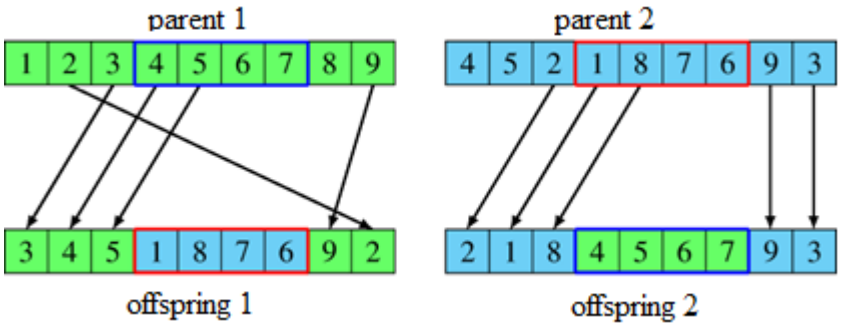

Fig. 2. Order crossover

In this operator, a fragment of the genotypes of the parents is first selected. This fragment has constant length and includes genes at the same positions in both genotypes. The fragment from the second parent is copied to the first child, without any changes. The remaining genes, at positions outside the selected fragment, are completed in the first offspring from the genes of the first parent, according to the order of their occurrence in the first parent, omitting those genes that were copied from the second parent.

The second offspring is created analogously with the role of both parents reversed.

2.3. Mutation operator

The mutation operator is relatively simple in comparison to the crossover operators used for permutations. In mutation random changes are made in the genotype of an individual what can result in both favorable and unfavorable changes. Unfavorable changes will make the individual less adapted, and thus will have less chance of being selected for crossing by the selection operator. Positive changes, on the other hand, increase the fitness of an individual. In the best case, a mutation may occur that will result in a solution close to another local optimum, or even a global optimum. In the simplest case, two elements of the permutation should be chosen at random and swapped.

More complex operators can also be used, for example consisting of several such exchanges, or swapping larger fragments of permutations.

In this paper the RSM (reverse sequence mutation) mutation operator is used (Mousa, et al., 2017), presented in Fig. 3.

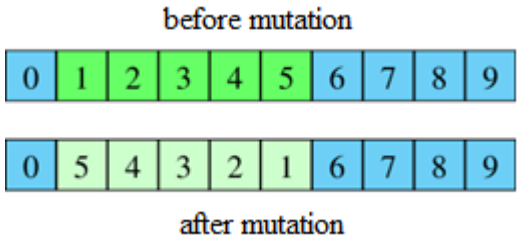

Fig. 3. Scheme of the RSM mutation operator

RSM operation is relatively simple. In the mutated genotype a fragment (string) of genes is selected. In this fragment, the order of the genes is then reversed. This operator is very easy to implement.

3. RESEARCH

An evolutionary algorithm can be implemented in various ways. Regardless of the implementation, one of the characteristics of EA is a large number of parameters on which its operation depends. Only two parameters were taken into account in the application: – population size, i.e., the number of individuals (solutions), – mutation coefficient, i.e. the probability that a given individual will be mutated. However, there can be many more parameters. For example, during a mutation, two (random) gene positions in the genotype are selected. These positions are selected from all genes. Therefore, it can be said that all genes in the genotype can (potentially) mutate, i.e., the range of the mutation operator is the entire genotype. This range could be limited if an additional parameter was adopted in the form of the maximum difference between the selected gene positions. Such a parameter was not introduced in the application, which means that its value was set at a constant level, equal to the size of the genotype.

The values related to the stopping criteria can also be treated as input parameters for EA. Several different stopping criteria have been assumed in the application:

- reaching an absolute number of generations,
- no improvement (of the best found solution quality) for 200 populations.

The number of recent generations where the best known solution has not improved is actually an EA parameter that has been set to a fixed value in the application.

The problem of determining the value of EA parameters is not easy to solve, mainly due to their large number. This paper focuses on two parameters, population size and mutation rate.

4. RESULTS AND CONCLUSIONS

Five different population group sizes were studied: 20, 50, 100, 200 and 500 individuals. Each group was tested for five different values of the mutation rate: 0.02, 0.20, 0.50, 0.80 and 1.00. All tests were performed exactly one hundred times. This gives a total of two and a half thousand individual tests.

In the course of all experiments carried out, it was noticed that no better solution than 2181 kilometers was ever obtained. Therefore, it was assumed that it is an optimal or suboptimal solution, which was used as a reference solution in further research. Due to the size of the problem, determining the optimal solution using the exact algorithm could take too long.

4.1. The quality of the solutions

The influence of the mutation parameter and the size of the population on the obtained results is shown in Fig. 4. All functions are non-decreasing, which means that as the mutation parameter increases, the quality of solutions improves, and more precisely, the frequency

of obtaining the best solutions increases. It can also be seen that the mutation is most important for small population groups. In larger groups, the influence of mutation coefficient is less significant. This means that for larger values of the mutation coefficient, the population will have a greater chance of getting out of the local minimum which is beneficial.

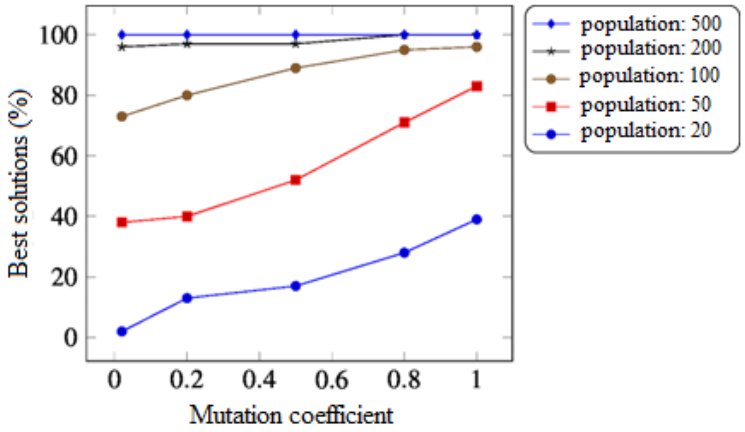

Fig. 4. Effect of input parameters on the quality of solutions

4.2. Number of iterations and best solutions

During the research, the impact of the selection of mutation parameters and population size on the number of iterations at which the best solutions appeared was also checked (Fig. 5). The analysis of the plot shows an adverse effect of the increased mutation coefficient on the number of generated generations (time) until the appearance of the best individual. It was also noticed that with too small populations, the lack of or too low a mutation coefficient also had a negative effect on the number of iterations required to reach the state of the population with the best-fit individual.

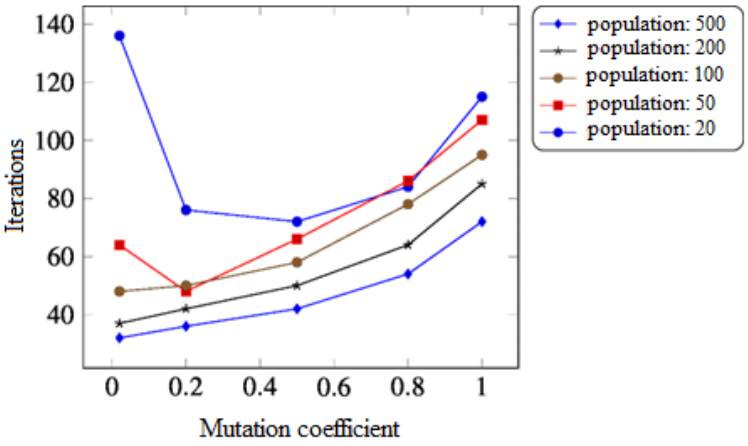

Fig. 5. Effect of input parameters on the number of iterations

It was therefore observed that too frequent mutations result in a slower rate of improvement of the population. Nevertheless, a very small population falls into a local minimum quicker. With too low a mutation rate, it gets out of such trap more slowly, and thus, reach the best adaptation more slowly. It was also noticed that the most numerous populations evolve the fastest.

5. SUMMARY

It should be remembered that the purpose of the calculations is not to increase the chance of the population leaving the local minimum but to obtain the best possible solution (although one may be related to the other). The larger the population size, the longer the computation time. This results directly from the EA itself: individuals are subjected to different operators, so the more individuals in the population, the longer (in terms of time) these operations will be performed. Therefore, one should rather strive for a situation in which the size of the population is as small as possible at the same time maintaining solutions of good quality.

The research shows that even in the case of small populations (eg. 20 individuals) it is possible to obtain a good (optimal or suboptimal) solution. Taking the calculation time into account, the value of the mutation coefficient should be at a relatively high level (0.50). Also the size of the population can be adjusted at the level of 50 individuals.

To sum up, it can be stated that the selection of the values of the EA parameters is not a simple task - even considering only two parameters. Experiments (research) need to be carried out to obtain good results in a relatively short time. The resulting parameter values are appropriate for the problem analyzed in this paper, but they do not necessarily have to be appropriate for other problems. Although EA can be regarded as a "universal" tool that can be easily applied to solve problems of various classes, the choice of values for the algorithm parameters makes this application difficult.

The conducted research also allowed to draw other conclusions. The most important is the fact that the relatively best results in the relatively shortest time can be obtained using a population size of 50 individuals, with a mutation rate of 0.50. EA is a stochastic algorithm, hence this regards only the probability of obtaining the result. It should therefore be concluded that for the given values of EA parameters, the probability of obtaining a good result is relatively high.

During the research, it was also noticed that higher values of the mutation coefficient allow to achieve good results even with small populations. However, this comes at the cost of more generations. Reducing the size of the population reduces the computation time, which is beneficial. On the other hand, increasing the number of generations means increasing the computation time. It is therefore necessary to reach a certain compromise, which results in the previously mentioned values of the mutation coefficient and population size.

REFERENCES

Abdoun, O. & Abouchabaka, J. (2011). *A Comparative Study of Adaptive Crossover Operators for Genetic Algorithms to Resolve the Traveling Salesman Problem.* arXiv. https://doi.org/10.48550/arXiv.1203.3097

Abellanas, M. R., & López-Ibáñez, M. (2008). An Introduction to the Traveling Salesman Problem. International Journal of Combinatorial Optimization Problems and Informatics, 1(1), 1-11. doi:10.4018/jcopi.2008010101

Crainic, T. G., Fodor, J., & Grigoras, C. (2007). A Hybrid Evolutionary Algorithm for the Traveling Salesman Problem. IEEE Intelligent Systems, 22(2), 41-48. doi:10.1109/MIS.2007.37

Davis, L. (1985). Applying Adaptive Algorithms to Epistatic Domains. *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, (vol 1, pp. 162-164).

Gao, Y., & Li, X. (2018). A Novel Hybrid Evolutionary Algorithm for the Traveling Salesman Problem. IEEE Access, 6, 7072-7081. doi:10.1109/ACCESS.2018.2848862

Goldberg, D. & Lingle, R. (1985). Alleles, Loci and the Traveling Salesman Problem. Proceedings of the 1st International Conference on Genetic Algorithms and Their Applications, (pp. 154-159).

Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Boston: Addison-Wesley Longman Publishing Co.

Grefenstette, J. J. (1986). Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1), 122-128. https://doi.org/10.1109/TSMC.1986.289287

Holland, J. H. (1975). Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. *Ann Arbor: University of Michigan Press.* https://doi.org/10.7551/mitpress/1090.001.0001

Kumar, S., & Sharma, S. (2017). A Novel Hybrid Genetic Algorithm for Solving Traveling Salesman Problem. International Journal of Computer Applications, 159(2), 1-7. doi:10.5120/ijca2017914072

Liao, Y. F., Yau, D. H., & Chen, C. L. (2012). Evolutionary algorithm to traveling salesman problems. *Computers & Mathematics with Applications*, 64(5), 788-797. https://doi.org/10.1016/j.camwa.2011.12.018

Dry, M., Lee, M. D., Vickers, D., & Hughes, P. (2006). Human Performance on Visually Presented Traveling Salesperson Problems with Varying Numbers of Nodes. *The Journal of Problem Solving*, 1(1).

Mousa, A. A., El-Shorbagy, M. A. & Farag, M. A. (2017). K-means-Clustering Based Evolutionary Algorithm for Multi-objective Resource Allocation Problems. *Applied Mathematics & Information Sciences*. 11(6), 1681-1692. https://doi.org/10.18576/amis/110615

Oliver, I. M., Smith, D. j., & Holland, J. R. C. (1987). A Study of Permutation Crossover Operators on the Traveling Salesman Problem. *International Conference on Genetic Algorithms.* (pp. 224-230).

Macgregor, J. N., & Ormerod, T. (1996). Human performance on the traveling salesman problem. *Perception & Psychophysics*, 58(4), 527–539. https://doi.org/10.3758/BF03213088

Zieliński, D., & Dereniowski, D. (2015). Evolutionary Algorithm for Solving the Traveling Salesman Problem. International Journal of Computer Science, 12(2), 1-7.