International Journal of Computer (IJC)

ISSN 2307-4523 (Print & Online)

© Global Society of Scientific Research and Researchers

http://ijcjournal.org/

Developing a Java-based Genetic Algorithm to Solve the Travelling Salesman Problem

Bashir Salisu Abubakar*

Department of Computer Science, Faculty Computing and Mathematical Science, Kano University of Science and Technology, Wudil, Kano Nigeria

Email: bsalisu2016@gmail.com

Abstract

In this paper, software was developed to solve the travelling salesman problem. The Travelling Salesman Problem is a computational optimization problem that requires a lot of time to solve using brute force algorithm. The research aims at developing java-based software that provides an optimum solution to the Travelling Salesman Problem using the concepts of genetic algorithm within reasonable time frame.

Keywords: Genetic Algorithm; Travelling Salesman Problem; Initial Population; Mutation; Crossover.

1. Introduction

The Travelling Salesman Problem (TSP) is a well-known example of a classical combinatorial optimization problem. The TSP is about finding the shortest possible path; given N number of cities where N must be greater than two ($N > 2$). The salesperson only visits a city exactly once and then moves on to the next city. After visiting all the cities, the salesman returns to the starting city. One of the factors to consider when optimizing the TSP is the order in which the cities are visited. The TSP is a relatively old problem. This idea came into place when Euler in 1759 developed an interest in solving the knight tour problem [16]. The term travelling salesman was first used in 1832 in a book written by a German salesman B.F. Voigt [16]. Planning a tour for the salesman is an expensive job when using brute force to solve the problem. If there are N cities to visit, then the number of possible paths is (N - 1) / 2. Assuming n = 6, then the number of tours for the salesman is 60 and he might solve it within one or two days, but what if N is 20, then the number of tours is approximately $6 \text{ X } 10^{16}$ which might take him years to solved. Genetic algorithms (GA) come into play when N is bigger. GA solves this kind of problem within a short period and with an optimum result.

----

* Corresponding author.

The TSP has occupied the thoughts of numerous researchers. Some of the reasons include the following; it has a wide area of application, it is NP-hard in nature, which makes it easy to define, but difficult to solve. Last, a method that will yield a better result has not yet been found. GA is search method that mimics the principles of natural evolution (such as selection, recombination, inheritance, and mutation) to find an exact or approximate solution to a complex optimization or search problem. The concept is better used in problems that involve discrete optimization or where conventional optimization techniques cannot yield a better solution that is non-differentiable or presents too many nonlinearly related parameters [2]. The search technique can be divided categorically into three different phrases: evaluation of the fitness of each phenotype, selection of a parent phenotype based on principles, and applications of genetic operators to the parent phenotype [1]. GA is used in numerous research fields such as engineering, economics, manufacturing, medicine, computational science, and bioinformatics to solve complex problems. In the mid-1970s Holland and his student at the University of Michigan proposed the GA, which started from cellular automata. He described the idea in his book titled Adaption of Natural and Artificial System (1975) [3]. GA became famous with the book written by David Goldberg in 1989 [15]. The first international conference on GA held in Pittsburgh, Pennsylvania in the mid-1980s changed the perspective of the algorithm from theoretical to practical [3]. The algorithm was developed with the intention of optimizing a numerical function and learning tasks. In the GA, the search space consists of phenotypes, which are the sets of possible solutions to the problem. A phenotype is the set of physical characteristics (traits) of an individual. GA operates on phenotypes [2]. A genotype is the genetic material of a phenotype that is inheritable.

2. Related Work

Many systems exist that solve the TSP using the genetic algorithm concepts. One of them is the work of Lee [13], "*Applying a genetic algorithm to the traveling salesman problem*." The system uses the genetic algorithm concept to solve the TSP. He uses the generational replacement selection method as his selection criteria and swap mutation as his mutation process. The major problem of the work is a pure Java code, not in the form of a graphical user interface (GUI), and all the genetic algorithm parameters are hard coded and the cities coordinate. Last, only students with a Java running environment can work with it because is not developed in a form of software that can be installed on any computer. Jeff Heaton [20] in Chapter 8 of his textbook entitled *Introduction to Neutral Networks with Java* solves the same problem using the genetic algorithm concept. The software starts by generating random genotypes, which are converted to phenotypes. A population of phenotypes is formed, and each phenotype in the population is evaluated. The best phenotypes are selected for reproduction in order to form a new population. Mutation is used in the new population to introduce diversity into the population. The software will continue to run until it finds the same phenotype for selection 100 times consequently. The major problem to this software is the lack of GUI (it is just a Java code), and random generation of city coordinates (not allowing the user to specify the coordinates). In addition, the parameters are hard coded, and the termination conduction is weak. The project will address the problems of the reviewed software. First, the TSP will be implemented with modified operators to suit the problem instead of a simple genetic algorithm in Java. Second, the problem of hard coded parameters will be solved by asking the user to input values from the interface of the application. Third, the problem of randomly generating the genotype will be solved by asking the user to input the city coordinates or read the coordinates from a text file. Fourth, the problem of selection criteria will be solved by implementing the selection criteria using appropriate methods, and a friendly user interface will be designed. Last, the problem of a weak terminating condition will be solved by asking the user to enter the number of iterations, and if a premature converge occurs before the number is reached, the application will stop.

3. Methodology

In this paper the concepts of GA will be used to develop software that will find the optimum solution to the TSP. GA is a technique that mimics natural selection and evolution to find a solution to a harder optimization problem, by explicitly using Charles Darwin's principle of survival of the fittest. GA usually starts with generating random genotypes. The genotypes are then converted to phenotypes. The phenotypes will be grouped together to form an initial population. Each phenotype in the initial population is assigned a numerical score. Phenotypes with better numerical scores are selected and paired together to produce offspring. The phenotypes that are paired are called parents, and the product of their pairing is called offspring. This offspring will form a new population, and the genetic materials of each offspring are altered to introduce diversity in the new population. Reaching a terminating condition will stop the algorithm otherwise the algorithm will continue to repeat itself. The general algorithm for a GA:

- [**Initial Population**] An arbitrary set of phenotypes that are randomly selected out of all possible solutions to form an initial population.
- [**Fitness**] An evaluation of the fitness level of each phenotype present in the population
- [**Selection**] Select two phenotypes randomly from the initial population on the basis of fitness.
- [**Crossover**] Mate the two selected parent phenotypes in step three to form a new offspring.
- [**Mutation**] applied to the offspring
- Stages three, four, and five are repeated until all the phenotypes are selected for pairing.
- [**Check**] if termination condition is met, stop and return the best solution. Otherwise replace the population with the offspring and go to step two.

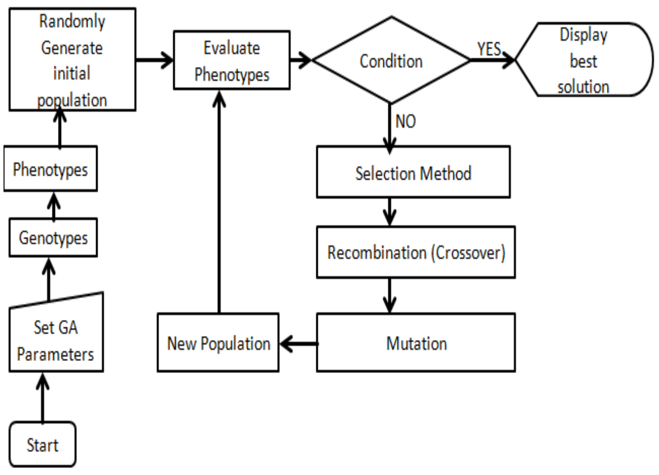
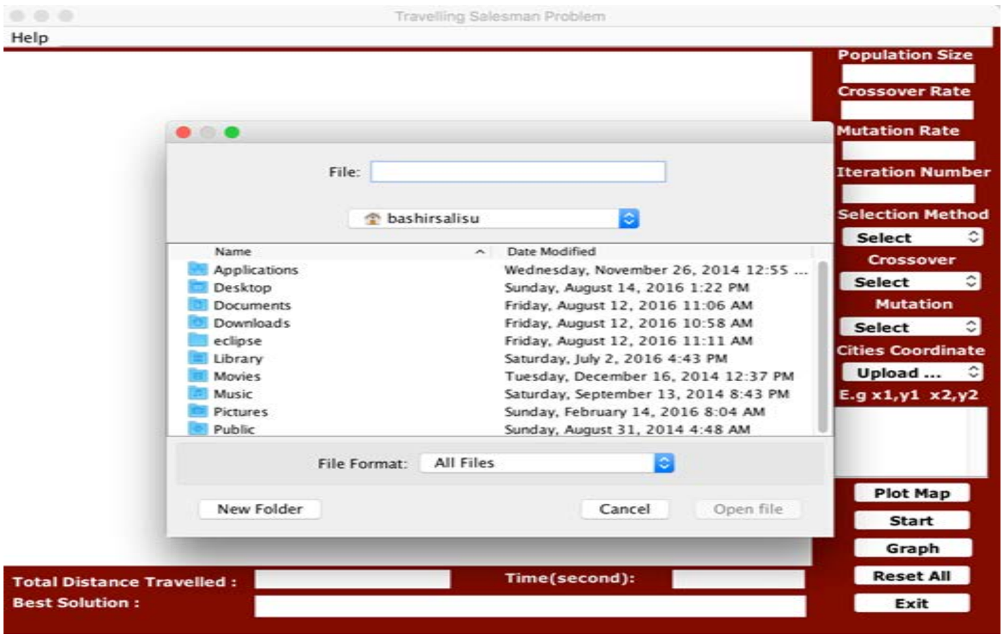


Figure 1: General Genetic Algorithm

4. Implementation

When the application is run, a GUI is shown to the user. This interface enables the user to enter the population, crossover rate, mutation rate, selection method, crossover method, mutation method and coordinates. This application implemented the tournament and the proportionate selections, so the user decides which one to use. The application also implemented three different crossover operators and two mutation operators. The application implements two different methods of entering the coordinates. The method provides the coordinates by either typing them in the text area or uploading them from a text file. If any file format apart from the text file is chosen, an error message is displayed**.**

After the user provides all the parameter values, the user clicks on the start button. The city coordinates input by the user are transformed into cities (genotypes) and this transformation is done in the ***City.Java*** class. Then a method called ***distanceBetweenCities(City city)*** is used to determine how far it is between two cities. Next, the cities are transformed into tours (phenotype) in the ***Tour.Java*** class. A method called ***totalDistance()*** calculates the total distance of each tour. Then, the fitness of the tour is evaluated, which is inversely proportionally to the total distance and is done by a method called ***getFitness().*** Next, the tours with the best fitness score are selected for crossover. This selection is handled by the either the tournament (Population) or the proportionate (Population) methods, which depends on the user's choice. The selected tour (phenotype) will undergo the crossover process by one of the crossover methods, which will return offspring. Diversity is introduced into the new population of the offspring by the ***swapMutation(mutation Rate, tour)*** method or the ***insertionMutation(mutation Rate, tour***) method, which depends on the user's choice. Then, the program displays the cities as dots in the map. As the process is repeated, the red line between the blue cities updates. The final output of the software is the shortest possible path as the best solution and total distance travelled, which will be displayed to the user. The figure below shows the implemented software.



Name	Date Modified
Applications	Wednesday, November 26, 2014 12:55 ...
Desktop	Sunday, August 14, 2016 1:22 PM
Documents	Friday, August 12, 2016 11:06 AM
Downloads	Friday, August 12, 2016 10:58 AM
eclipse	Friday, August 12, 2016 11:11 AM
Library	Saturday, July 2, 2016 4:43 PM
Movies	Tuesday, December 16, 2014 12:37 PM
Music	Saturday, September 13, 2014 8:43 PM
Pictures	Sunday, February 14, 2016 8:04 AM
Public	Sunday, August 31, 2014 4:48 AM

Figure 2: Software Interface

5. Testing

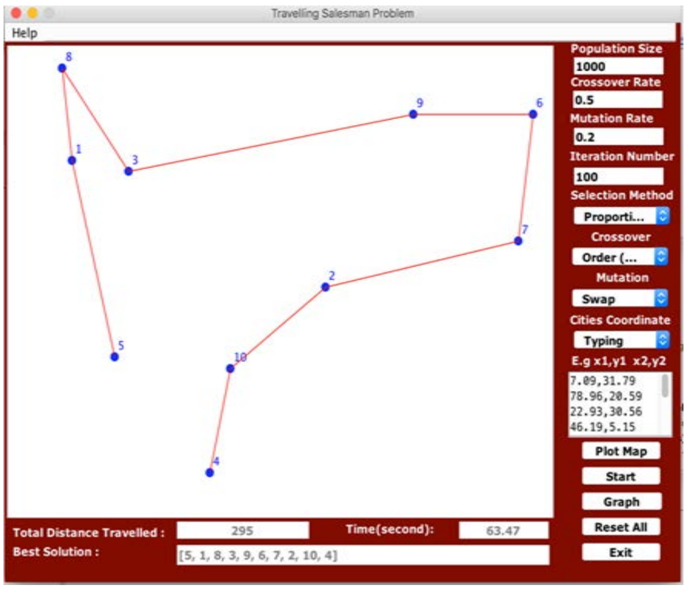
To test the software after fully implemented, the coordinates (that's longitude and latitude) of 10 countries in the world are used. Table below shows the cities with their respective coordinates.

Table 1: Cities with their Coordinate

S/N	CITY	LONGITUDE	LATITUDE
1	Morocco	7.09	31.79
2	India	78.96	20.59
3	South Africa	22.93	30.56
4	Somalia	46.19	5.15
5	Chad	18.73	15.45
6	Japan	138.25	36.20
7	Australia	133.77	25.27
8	Spain	3.75	40.46
9	China	104.19	35.86
10	Brazil	51.92	14.23

5.1 Result

The result of this test shows that the shortest possible path to visit each of the ten (10) countries once is [5, 1, 8, 3, 9, 6, 7, 2, 10, 4] which corresponds to the countries as: Chad, to Morocco, to Spain, to South African, to China, to Japan, to Australia, to India, to Brazil, to Somalia.



Population Size: 1000
Crossover Rate: 0.5
Mutation Rate: 0.2
Iteration Number: 100
Selection Method: Proporti...
Crossover: Order (...
Mutation: Swap
Cities Coordinate: Typing
E.g x1,y1 x2,y2

```
7.09,31.79
78.96,20.59
22.93,30.56
46.19,5.15
```

Total Distance Travelled : 295
Time(second): 63.47
Best Solution : [5, 1, 8, 3, 9, 6, 7, 2, 10, 4]

Figure 3: Solution

6. Recommendation

To make improvements to the software system, some work will be necessary in order to make the application more flexible for researchers or students to understand the concepts of the genetic algorithm. The future tasks involve implementation of more selection criteria and a different crossover method. This will make the application more usable. Also, the transformation of the software into an android or iPhone application will be of good future use. Since almost everyone works with a phone, this will encourage the students to play more with it.

7. Conclusion

This research paper presented software that solves the travelling salesman problem using the concept of genetic algorithm using java-programming language. The application produced an optimum solution to the different sets of problems.

References

[1]. Jean-Yves, P. (1996) ‘Genetic algorithms for the travelling salesman problem’, Annals of Operations Research. 63(3), pp 337-370

[2]. Dario, F. and Claudio M. (2008) Bio-Inspired Artificial Intelligence. London: The MIT Press Cambridge, Massachusetts.

[3]. Holland, J. H., Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, USA,1975.

[4]. ‘Genetic Algorithm’ (2014) Wikipedia. Available at: http://en.wikipedia.org/wiki/Genetic_algorithm (accessed on 24/06/2014)

[5]. Sivanandan, S.N., and Deepa, S.N. (2008) Introduction to Genetic Algorithms, Springer

[6]. Pedro A. and Dean F. (2007) Initial Population for Genetic Algorithm: A metric Approach: School of computer science, University of Oklahoma, Norman. USA. Available at: http://www.cameron.edu/~pdiaz-go/GAsPopMetric.pdf (accessed 12/08/2014)

[7]. Uniform crossover (2014) Wikipedia. Available at: http://en.wikipedia.org/wiki/Crossover_(genetic_algorithm) (accessed on 26/06/2014)

[8]. Proportionate Selection (2014) Wikipedia. Available at: http://en.wikipedia.org/wiki/Fitness_proportionate_selection(accessed on 27/06/2014)

[9]. Khalid J., and Mohammed M.,(2013) ‘Selection Methods for Genetic Algorithms’ int. J. Emerg. Sci. 3(4), pp.333-334. Available at: http://ijes.info/3/4/42543401.pdf

[10]. Goldberg E., and Lingle R., ‘Alleles, loci, and travelling salesman problem’. InProc. Of the International Conference on Genetic Algorithms and Their Applications, pp. 154-159, Pittsburgh, PA, 1985.

[11]. Oliver, I.M., and Smith, D.J., and Holland, J.R.C. ‘A study of permutation crossover operation on travelling salesman problem’. Proceedings of the second International Conference on Genetic Algorithms on Genetic algorithm and their application, pp. 244-230. NJ, USA 1987.

[12]. Noraini M., and John G., (2011) 'Genetic Algorithm performance with Different Selection Strategies in Solving TSP', Proceedings of the World Congress on Engineering. Vol II, London, U.K. Available at: http://www.iaeng.org/publication/WCE2011/WCE2011_pp1134-1139.pdf (accessed 28/06/2014)

[13]. Lee (2012) Applying a genetic algorithm to the traveling salesman problem http://www.theprojectspot.com/tutorial-post/applying-a-genetic-algorithm-to-the-travelling-salesman-problem/5 (accessed 28/06/2014)

[14]. Goldberg, D. E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, New York, NY, 1989.

[15]. Zbigniew, M., (1994) Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, 2nd edition, 1994.

[16]. Rajesh, M., Surya, P. S., and Murari, L. M., (2010) Travelling Salesman Problem :An Overview of Applications, Formulations, and Solution Approaches; Department of Management Studies, Indian Institute of Technology Delhi, New Delhi, Available at: http://cdn.intechopen.com/pdfs-wm/12736.pdf (accessed 12/08/2014)

[17]. Zhifeng, H., Huang, H., and Cai, R., (2008) Bio-inspired Algorithms for TSP and Generalized TSP. Travelling Salesman Problem. Shanghai, China. Available at: http://cdn.intechweb.org/pdfs/4606.pdf (accessed 20/08/2014)

[18]. Kumar R., Gopal G., (2013) 'Novel Crossover Operator for Genetic Algorithm for Permutation Problems', International Journal of Soft Computing and Engineering 3(2) pp. 2231-2307

[19]. MATLAB www.matlab.com

[20]. Heaton, J., 2005. Introduction to Neutral Network with Java. (1st ed). U.S.A: Heaton Research, Inc.

[21]. Deitel, P. and Deitel, H., (2013) Java: How to Program. (9th ed). New Jersey, U.S.A: Pearson Education, Inc.

[22]. Cornell and Horstmann (2008) Core Java, volume I/II (8th ed). U.S,A :Prentice Hall

[23]. Lewis, J. and Loftus, W., (2007). Java Software Solution: foundations of program design. (5th ed). U.S.A: Addison Wesley

[24]. Noraini, M and John G., (2011) 'Genetic Algorithm Performance with Different Selection Strategies in solving TSP'. Proceedings of the World Congress on Engineering Vol II, London, U.K

[25]. Srinivas, M., and Patnaik, L.M., (1994) 'Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms'. IEEE Transactions on System, man and cybernetic, Vol. 24, No 4, April 1994.