How to cite this paper:

Mohammed Bin Jubeir, Mishal Almazrooie, & Rosni Abdullah. (2017). Enhanced selection method for genetic algorithm to solve traveling salesman problem in Zulikha, J. & N. H. Zakaria (Eds.), Proceedings of the 6th International Conference of Computing & Informatics (pp 69-76). Sintok: School of Computing.

ENHANCED SELECTION METHOD FOR GENETIC ALGORITHM TO SOLVE TRAVELING SALESMAN PROBLEM

Mohammed Bin Jubeir, Mishal Almazrooie, and Rosni Abdullah

School of Computer Sciences
Universiti Sains Malaysia, USM,
Pulau Pinang, 11800, Malaysia

ABSTRACT. Genetic algorithms (GAs) have been applied by many researchers to get an optimized solution for hard problems such as Traveling Salesman Problem (TSP). The selection method in GA plays a significant role in the runtime to get the optimized solution as well as in the quality of the solution. Stochastic Universal Selection (SUS) is one of the selection methods in GA which is considered fast but it leads to lower quality solution. Although using Rank Method Selection (RMS) may lead to high quality solution, it has long runtime. In this work, an enhanced selection method is presented which maintains both fast runtime and high solution quality. First, we present a framework to solve TSP using GA with the original selection method SUS. Then, the SUS is replaced by the proposed enhanced selection method. The experimental results show that a better quality solution was obtained by using the proposed enhanced selection method compared to the original SUS.

Keywords: TSP, Genetic Algorithm, GA, Evolutionary algorithms, Selection Methods.

I. INTRODUCTION

Traveling Salesman Problem is one of the most widely studied problems in combinatorial optimization. Having a list of cities and distances between them, the aim of the problem is getting the shortest possible path among cities. Although TSP is NP-Hard, the traveling salesman needs to visit each city only once and to return to the first city in the list. Generally, there are mainly two ways of finding an optimal solution to TSP problem. The first way, which is almost closer to the exact solution, is to solve this problem optimally which can be called exact algorithms, and thus finding the exact length through using exhaustive search. The second way, heuristic algorithms or approximate method, consists of algorithms that either give an optimized solution or a specific solution, although this is not for all instances of the problem (Nilsson, 2003); Anitha Rao & Hegde, 2015; Rasit Er & Erdogan, 2013). Since solving TSP algorithms offers optimal solutions and takes too long time, it is preferred to use heuristics algorithms because they give researchers information on how bad solutions can be obtained. The GA result is one of approximation algorithms which can give the nearest optimal solution within the time deemed reasonable (Nilsson, 2003). Although it has been pointed out that GA is a simple method, it is an efficient heuristic method that can be used when the search space is large, and complex. Also, the continues improvement of solution quality has made GA attractive for a lot of problems (Islam, Pandhare, Makhthedar, & Shaikh, 2014; Anitha Rao & Hegde, 2015).

The main purpose of this work is to improve the solution quality of the GA and to maintain a reasonable runtime. Hence, we present an enhanced method for the parent selection in GA. In this study, the performance of the enhanced GA was evaluated and compared with the original GA and tested with different problem sizes.

This paper consists of six sections. While the related works are reviewed in Section II, Section III gives a description of TSP and presents the structure of TSP. The definition of Genetic Algorithm and explanation of the biological processes on which they are based are presented in Section IV. Section V is devoted to the underlying idea behind this proposed work. While the experiments and their results are presented in Section VI, the final section provides the conclusion of the study and suggestions for future research.

II. RELATED WORKS

This study highlights methods that are developed to solve the TSP using GA based on sequential approaches to improve the solution quality and to reduce the required time to come up with a practical solution.

Aranganayaki (2014) in this paper, TSP is solved by using GA. The main aim of the paper is to present a new method that can be employed to improve the quality solution and to reduce the runtime. So, a new crossover method, the Sequential Constructive Crossover (SCX) method is used. This method selects the better edges from the parent chromosome and produces a new offspring which may have the same edges as the parents or it may have new edges which are not present in the parent chromosomes. Furthermore, another aim of this study is to propose a binary matrix representation of chromosomes.

To solve TSP, Ahmed (2010) analyzed and compared three types of Crossover: Sequential Constructive Crossover (SCX), Generalized Npoint Crossover (GNX), and Edge Recombination Crossover (ERX). The empirical result of his study revealed that the SCX has a high quality solution compared to GNX, and ERX.

Marg, Campus, & Pradesh (2014) applied Roulette Wheel method and Stochastic Universal Sampling method for selecting parents to solve TSP. In addition, Order Crossover (OX) was used. Stochastic Universal Sampling gave satisfactory results when the population size is small. Using the SUS with Elitism Method, the GA can give better solutions with a big population size.

Most studies have employed GA for solving TSP to reduce the runtime and to improve the quality of the end result. In addition, most researchers who intended to improve the quality solution used the Ranking Selection (Noraini & Geraghty, 2011; Karakati & Podgorelec, 2015), which consumes a lot of time when the population size increases. Although some other researchers used Roulette wheel method, the Stochastic Universal Selection or Tournament Selection, but the quality of the results was lower (Noraini & Geraghty, 2011; Baker, 1987; Pandey, 2016) Thus, the main purpose of this work is how to improve the solution quality of the GA and maintain a reasonable runtime.

III. TRAVELING SALESMAN PROBLEM

In operational research, TSP has been identified as an interesting issue. Additionally, it is regarded as one of the most significant optimization topics. TSP is modelled as a graph where nodes represent the cities, and edges represent the distance between each city pair. TSP, which is sometimes referred to as an NP-complete problem, can be developed to be an acceptable solution for any other problems that are related to NP-complete class. The traveller's task is to visit all cities in the list and to go back to the first city he visited. Furthermore, the traveller has to travel to each single city only once (Rao, IAnitha and Hegde et al., 2015).

The two classes of TSP are Symmetric Travelling Salesman Problem (STSP) and Asymmetric Travelling Salesman Problem (ATSP) (Sánchez, 2015; Wang, Ersoy, He, & Wang, 2016). Symmetric TSP is bi-directional and it allows more focus on the development of the algorithm. In addition, the heuristics discussed in this paper will mainly be concerned with the STSP. Several researchers have used STSP because the distance can be easily checked.

IV. GENETIC ALGORITHM

Genetic Algorithms (GAs) are search methods that rely on principles of natural selection and genetics (Nilsson, 2003; Deshmukh, 2016). They have been applied successfully to several problems in engineering, business, and science (Deb, 2001). GA is a subset of evolutionary algorithms (EAs) which transforms a set (population) of solutions into a new population, using some of genetic operators such as Selection, Crossover, and Mutation to optimize the solutions for the problem. Figure 1 presents the basic components of Genetic Algorithm (Herrera, Lozano, & Verdegay, 1998).

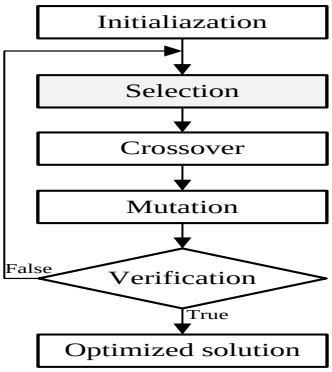

Figure 1: The basic components of GA.

As given in Figure 1, the objective of selection is to distinguish between individuals based on their quality, and, in particular, to choose the fitter individuals as the parent in the population that will create offsprings for the next generation, commonly known as mating pool. The mating pool is used to generate new offsprings and to push the population to the next generation (Sánchez, 2015; Sivanandam & Deepa, 2007).

The Selection phase in GA is time consuming, especially when the population size is very large (Rasit Er & Erdogan, 2013; Nowostawski & Poli, 1999; Pandey, 2016). There are many selection mechanisms such as Roulette Wheel Selection (RWS), Rank Selection (RS), and Stochastic Universal Selection (SUS), which can help in choosing a number of possible solutions based on fitness value instead of considering all populations (Sánchez, 2015).

V. ENHANCED SELECTION METHOD FOR GA

The Rank Selection (RS) always gives the highest solution quality in terms of distance (Pandey, 2016; Noraini & Geraghty, 2011). Since the RS depends on the sort population, the best fitness is the position = 0, while the worse fitness is the position = N. In addition, this position number gives a high fitness proportion when it is chosen. RS consists of two processes. The population is first sorted according to the fitness values, and then the individuals are ranked. Such process leads to high time complexity because the sorting process is performed twice. On the other hand, the original Stochastic Universal Selection (SUS), which was proposed by Baker (1987), is an enhancement of RWS. SUS provides zero bias and minimum spread. In RWS, there is one element (pointer) which indicates the winner while SUS gives a group of winners. Throughout the selection process in SUS, the sum (T) of the select-

ed winner is calculated and updated during the process. For a new individual to be selected to the group, the value of the new member should be less than T. Algorithm 1 illustrates the steps of SUS (Blickle & Thiele, 1995; Baker, 1987). Figure 2 shows the selection process of the individuals using SUS. As shown in Figure 2, a number N of pointers are the selected individuals as parents. The position of the first pointer is given by a randomly generated number in the range [0, 1]. After that, each individual is mapped to the segment over the line using a probability proportional to the fitness of the individual, and it accepts the selection of individual. The probability is computed as shown in Eq. (1) (Goldberg & Deb, 1991; Pandey, 2016; Eiben & Smith, 2015; Blickle & Thiele, 1995).

$$Ps\ (i) = \frac{f_i}{f_{max}} \tag{1}$$

where f_i is the fitness value of an individual i which is chosen and f_{max} is the maximum fitness in the population (Lipowski & Lipowska, 2011).

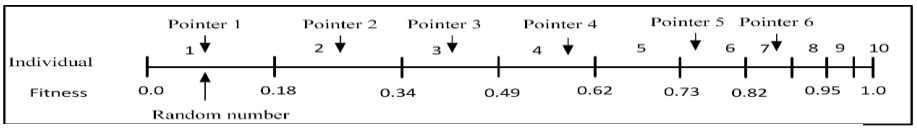

Figure 2: Stochastic Universal Sampling.

Algorithm 1: Code Fragment for original SUS (Baker, 1987).

Input $ExpVal(i)$ gives the expected value of individual i, and i is an index over population members and each individual i is guaranteed to reproduce at least $ExpVal(i)$ but no more than $ExpVal(i)$

```
1. ptr = rand(); /* Returns random number uniformly distributed in [0,1] */
2. for (sum = i = 0; i < N; i++)
3.     for (sum+= ExpVal(i); sum > ptr; ptr++)
4.         SelectInd[i];
```

The original SUS has less time complexity than RS because it selects the parent randomly without any sorting process. Subsequently, this may lead to low quality of the selected parent (Noraini & Geraghty, 2011). In this work, the focus is narrowed down to the way of selecting an individual for the next generation. The underlying idea behind this alternative method depends on four steps:

1. Choose any individual from the population randomly.
2. Dividing the fitness value of the selected individual by the total number of organisms (population size) to find the starting point, as shown in the Eq. (2):

$$starting\ point = \frac{f_{rand}}{population\ size} \tag{2}$$

The remainder of this formula is used to determine the starting point. Such method gives a different starting point in each iteration and it provides zero biased to any individual's position. To find out the probability of selecting individual (enhanced SUS), the following two steps are employed:

3. Find the minimum fitness value in the population which has the minimum distance over the cities, where the time complexity is equal to O (*n*).
4. Select individuals whose fitness values are less than the minimum fitness value plus a proportion *P* which is located between $\mathbf{0} < \boldsymbol{P} < \mathbf{1}$, as shown in the formula below:

$$Ps\ (i) = \frac{f_{(i)}}{f_{best}+P} \tag{3}$$

where $f_{(i)}$ is the fitness which is chosen, f_{best} is the best fitness (minimum fitness value) in the current population, and P is proportion that is added to the best fitness.

The main aim of the proposed enhanced method is to allocate the selection probabilities to individuals through Eq. (3). (Eq. (3) is the alternative enhanced selection method compared to the original selection method which is represented by Eq. (1). This occurs according to their actual fitness values rather than a constant selection pressure by sorting the population on the basis of fitness, which is present in RS method (Eiben & Smith, 2015). Algorithm 2 illustrates the steps of the enhanced SUS.

This proposed selection method is similar to RS but without sorting, which can select any individual that has a value less than the minimum fitness value plus a proportion P. While the original SUS selects any individual, it may give us good or bad result.

Algorithm 2: The proposed selection method.

Input: Fitness f such that f is the best fitness of the current generation, and O_s size or number of organisms.

Output: An organism O such that $fitness(O) < f + (f \times 0.03)$.

1. $rate \leftarrow f + (f \times 0.03)$
2. $ptr \leftarrow rand()\ \%\ O_s$
3. **for** $(i < O_s)$ **do**
4. $\quad ptr \leftarrow (i + ptr)\%O_s$
5. $\quad$ **if** $(fitness(ptr) < rate)$ **then**
6. $\quad\quad O \leftarrow ptr$
7. $\quad\quad$ **break**

VI. EXPERIMENTS AND RESULTS

The algorithm was implemented in C language on a machine equipped with Intel® Core (TM) i5–3210 M Processor 2.5GHz, and 4GB RAM windows 7 Operating System. With reference to the dataset, all data from TSPLIB and TSP formats are available on the TSPLIB homepage (Naval, 2015). The name of the dataset used in this work and the parameters of GA are shown in Table 1. The solution quality and execution time are affected by the selection of the parameters. GA is a stochastic algorithm whose complexity depends on the number of generation, population size, chromosome representation, and the calculation of the fitness function. In this work, the sequential time execution of GA is calculated by a function called $gettimeofday()$, which has a high resolution.

Table 1: Parameters of Genetic Algorithm.

Parameters	Value
TSP name	gr24, brazil58, si175 and pa561
Population Size	1024, 2048, 4096
Parent Selection method	Enhanced Stochastic Universal Selection (SUS)
#of iteration(as ending criterion)	1000
Crossover type	One-Point Crossover
Mutation type	Swap

The *P* value in Equation 3 is set in this work to (30%). This value has been selected after conducting several experiments. The experiments were conducted to measure the solution quality and time complexity through the execution of 10 trials on each dataset. Moreover, this experiment was done to obtain the average tour length and average execution time by using GA with the original selection method SUS and enhanced selection method SUS implementation. Table 2 below shows the average tour length and the average execution time for both methods with the optimized value of gr24, brazil58, and gr120, with variants iterations, where the optimized values for each dataset are gr24 (1272), brazil58 (25395), si175 (21407), and pa561(2763).

Table 2: Average Execution Time (In Seconds) and Average Tour Length Using GA with the Original Selection Method SUS, and Enhanced Selection Method SUS.

TSPLIB data Instance (problem size	**Iteration Number**	**Population Size**	**Original Selection Method SUS**		**Enhanced Selection Method SUS**	
			Average execution time(s)	Average tour length	Average execution time(s)	Average tour length
gr24	1000	1024	0.9	1323	0.85	1258
	1000	2048	2.2	1314	2.1	1313
	1000	4096	11.9	1361	10.6	1320
brazil58	1000	1024	4.0	99887	3.7	33695
	1000	2048	13.4	103250	18.3	70800
	1000	4096	45.9	105050	82.7	84443
si175	1000	1024	16.9	45068	15.0	33652
	1000	2048	66.2	46450	69.1	43590
	1000	4096	156.7	45968	168.4	44803
pa561	1000	1024	636.6	25814	619.7	10637
	1000	2048	1278.8	25405	1232.0	16905
	1000	4096	2626.8	25359	2539.5	18191

As shown in Table 2, the results of both the original SUS and the proposed selection method are recorded. When the population size is set to 1024, the best average tours for the three TSP sizes are observed in the proposed method. Note that, when genetic algorithm is used to solve a problem such as TSP, smaller fitness (tour length) will be better. The proposed method can achieve a better quality solution in a faster time with 1024 population size. When the population size is set to 2048 or 4096, the proposed method still achieves high quality solutions in reasonable average runtimes. For example, as revealed by the result of population of 1024 size, the better fitness (the minimum distance over the cities) have been obtained in short time, compared with the original selection method. The proposed method selects individuals which have good finesses because the fitness value is divided by the minimum fitness value plus a proportion *P*. In contrast, in the original SUS, the selection of the individuals is random where individuals of both good and bad finesses are selected. The results are represented as bar charts in both Figure 3 and Figure 4.

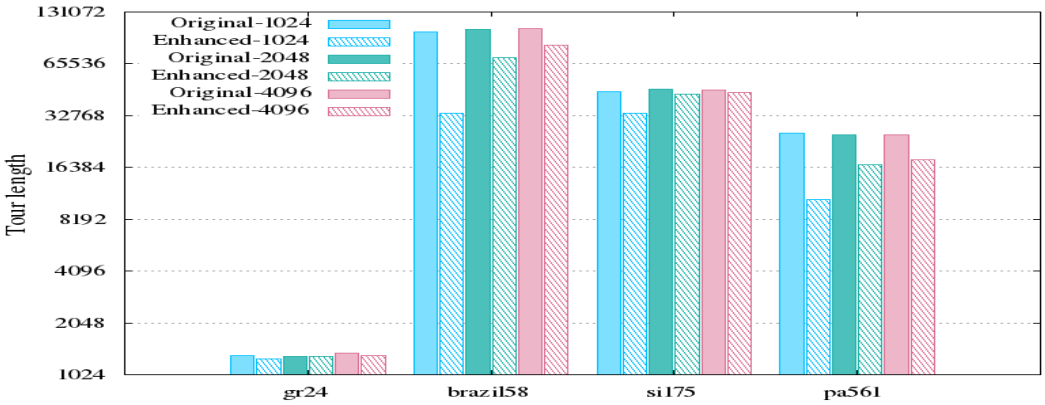

Figure 3: The results of the tour length of both the original and enhanced selection methods at population sizes of 1024, 2048, and 4096. X-axis indicates to the four datasets used in this work (gr24, brazil58, si175, and pa561). Y-axis indicates to the tour length.

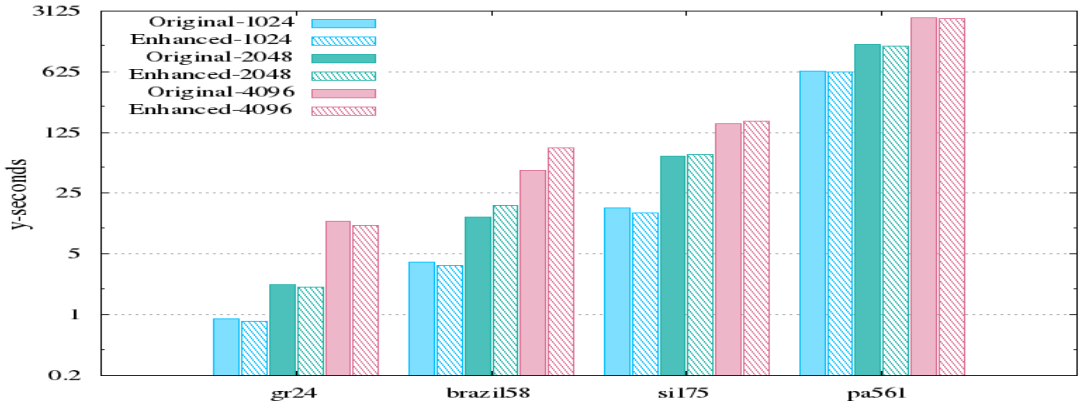

Figure 4: The results of the elapsed time of both the original and enhanced selection methods at population sizes of 1024, 2048, and 4096. X-axis indicates to the four datasets used in this work (gr24, brazil58, si175, and pa561). Y-axis indicates to the elapsed time in seconds.

VII. CONCLUSION

In this work, a new selection method of Genetic Algorithm is proposed. The main concern of the proposed selection method is to obtain a high quality solution in a reasonable time. The well-known hard problem TSP is used as a case study in this work to evaluate the proposed method with the original SUS. First, we presented a framework of solving TSP by using Genetic Algorithm. Then, we replaced the SUS selection method with our proposed method. The experimentation of GA was carried out on a well-known TSPLIB benchmark dataset of variants of the problem size. Compared to GA using SUS, GA using the proposed selection method in this work shows better results on the quality of the solution. Based on these findings, future research may study the algorithm behavior when the dataset increases. Also, this paper could be extended to focus on improving the solution quality by finding the relationship between the proportion that is added to the best fitness and the population size. This may lead to a better consistency.

REFERENCES

Ahmed, Z. H. (2010). Genetic Algorithm for the Traveling Salesman Problem using Sequential Constructive Crossover Operator. *International Journal of Biometrics & Bioinformatics (IJBB)*, 3(6), 96–105.

Anitha Rao, & Hegde, S. K. (2015). Literature Survey On Travelling Salesman Problem Using Genetic Algorithms. *International Journal of Advanced Research in Eduation Technology (IJARET)*, 2(1), 42.

Aranganayaki, A. (2014). Reduce Total Distance and Time Using Genetic Algorithm in Traveling Salesman Problem. *International Journal of Computer Science & Engineering Technology*, 5(2229–3345), 4.

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the second international conference on genetic algorithms* (pp. 14–21).

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms* (Vol. 16). John Wiley & Sons.

Eiben, A. E., & Smith, J. E. (2015). *Introduction to Evolutionary Computing*. (Second Edition, Ed.).

Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*, 1, 69–93.

Herrera, F., Lozano, M., & Verdegay, J. L. (1998). *Tackling Real-Coded Genetic Algorithms*: Operators and Tools for Behavioural Analysis. Artificial Intelligence Review, 12(4), 265–319. http://doi.org/10.1023/A:1006504901164

Islam, M. L., Pandhare, D., Makhthedar, A., & Shaikh, N. (2014). A Heuristic Approach for Optimizing Travel Planning Using Genetics Algorithm. *International Journal of Research in Engineering and Technology eISSN*: 2319-1163, pISSN: 2321, 7308(1).

Karakati??, S., & Podgorelec, V. (2015). A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing Journal*, 27, 519–532. http://doi.org/10.1016/j.asoc.2014.11.005

Lipowski, A., & Lipowska, D. (2011). *Roulette-wheel selection via stochastic acceptance*. http://doi.org/10.1016/j.physa.2011.12.004

Marg, S., Campus, J., & Pradesh, U. (2014). SOLVING TRAVELLING SALESMAN PROBLEM USING GENETIC ALGORITHM BASED ON HEURISTIC CROSSOVER AND MUTATION OPERATOR. *International Journal of Research in Engineering & Technology*, 2(2), 27–34.

Naval, R. the O. of. (2015). A good source for computational research on the traveling salesman problem and general optimization. Retrieved from http://www.tsp.gatech.edu/

Nilsson, C. (2003). *Heuristics for the traveling salesman problem*. Linkoping University, 3–8. http://doi.org/10.1016/S0305-0548(98)00085-9

Noraini, M., & Geraghty, J. (2011). Genetic algorithm performance with different selection strategies in solving TSP. *World Congress on Engineering*, II(978-988-19251-4–5), 4–9.

Pandey, H. M. (2016). Performance Evaluation of Selection Methods of Genetic Algorithm and Network Security Concerns. *Procedia Computer Science*, 78(December 2015), 13–18. http://doi.org/10.1016/j.procs.2016.02.004

Rao, IAnitha and Hegde, K., Rao, A., Hegde, K., Rao, IAnitha and Hegde, K., Rao, A., & Hegde, S. K. (2015). Literature Survey On Travelling Salesman Problem Using Genetic Algorithms. *International Journal of Advanced Research in Eduation Technology (IJARET)*, 2(1), 4.

Rasit Er, H., & Erdogan, N. (2013). *Parallel Genetic Algorithm to Solve Traveling Salesman Problem on MapReduce Framework using Hadoop Cluster*. Jscse, 3(3), 380–386. http://doi.org/10.7321/jscse.v3.n3.57

Sánchez, L. N. G. (2015). Parallel Genetic Algorithms on a GPU to Solve the Travelling Salesman Problem. *Revista En Ingeniería Y Tecnología*, UAZ, 8(2), 79–85.

Sivanandam, S. N., & Deepa, S. N. (2007). *Introduction to genetic algorithms*. Springer Science & Business Media.

Wang, J., Ersoy, O. K., He, M., & Wang, F. (2016). Multi-offspring genetic algorithm and its application to the traveling salesman problem. *Applied Soft Computing*, 43, 415–423. http://doi.org/10.1016/j.asoc.2016.02.021