Research Article

Genetic algorithm and a double-chromosome implementation to the traveling salesman problem

Amin Riazi[1]

Abstract
The variety of methods used to solve the traveling salesman problem attests to the fact that the problem is still vibrant and of concern to researchers in this area. For problems with a large search space, similar to the traveling salesman problem, evolutionary algorithms such as genetic algorithm are very powerful and can be used to obtain optimized solutions. However, the challenge in applying a genetic algorithm to the traveling salesman problem is the choice of appropriate operators that could produce legal tours. In the literature, additional repair algorithms have been introduced and employed and the offspring produced by these genetic algorithm operators are modified to ensure that the generated chromosomes represent legal tours. Rather than sticking to repair algorithms, a double-chromosome approach is proposed in this article. The proposed method can be employed to optimize problems similar to the traveling salesman problem. The double-chromosome approach has been tested with a variety of traveling salesman problems, and the results indicated that the proposed method has a high rate of convergence toward the shortest tour.

Keywords Best solution · Exact solution · Genetic algorithm · Optimization

1 Introduction

The traveling salesman problem (TSP) with its extremely large search space is one of the most well-studied NP-hard combinatorial optimization problems [1–5]. The classic form of TSP has been successfully applied to many real-world applications such as the genome sequencing, scan chains, drilling problems, and aiming telescopes [6].

In a symmetric TSP where the distance between two cities is same in each opposite direction, the total number of possible tours (θ) of n cities can be obtained by eliminating the similar tours as:

$$\theta(n) = \frac{(n-1)!}{2}. \quad (1)$$

In order to solve a TSP with even a moderate number of cities and in the interest of optimized tour, an extremely huge search space should be investigated and a massive computational time will be required. Although exact algorithms similar to the brute force approach (BFA) that allow for evaluating all possible solutions are guaranteed to find the optimal solution, they can only be applied to small TSPs up to 10 cities [7, 8], and thus, for average and big TSPs the direct methods are, in fact, useless. Therefore, the development and application of heuristic algorithms that could find the optimal or near-optimal solution in a limited time frame [7] have been massively studied. Algorithms like but not limited to tabu search [9], Lin–Kernighan heuristic [10] have been significantly improved over years as successful methods for obtaining the optimal or near-optimal solutions [11] as well as genetic algorithm (GA) [8, 12–17].

For problems with a large search space, like the TSP, evolutionary algorithms such as genetic algorithm are very powerful and can be used to obtain the optimized solution [18]. However, the pure genetic algorithm cannot be applied to combinatorial optimization problems and proposing a genetic algorithm that can be applied

✉ Amin Riazi, ariazi@ciu.edu.tr | [1]Civil Engineering Department, Cyprus International University, 99258 Nicosia, North Cyprus, Turkey.

SN Applied Sciences (2019) 1:1397 | https://doi.org/10.1007/s42452-019-1469-1

Received: 7 August 2019 / Accepted: 3 October 2019 / Published online: 14 October 2019

to TSP-like problems is quite challenging. To apply GA to the TSP with n cities, a chromosome representation for genetic information is required. The classic type of GA chromosomes is not applicable as the possible tours cannot be mapped by binary strings. Instead, in the literature, path representation has been widely used [14]. In the path representation, the chromosome corresponding to a TSP tour is an array of n integers. Each chromosome with a length of n genes will be a permutation of (1, 2,…, n). In the generated chromosome, the gene with a value of i in the position of j indicates that the city i is visited in the j-th time order instant [15].

As each chromosome should be a representative of a legal tour, the objective of GA is to find a legal chromosome that represents the shortest tour. The crossover and mutation genetic algorithm operators are used to produce new solutions that could be used to explore the entire search space. However, the operators do not necessarily result in a legal tour. Through pure GA, there are many possibilities that illegal tours are generated and a city is visited more than once, or a city is not visited at all. Thus, the main challenge in applying GA to the TSP is to propose operators that could produce legal tours. What could be observed from the literature is that instead of proposing modifications to the operators, usually additional repair algorithms have been proposed. The offspring produced by the GA operators are modified by the repair algorithms to ensure that the obtained chromosomes represent the legal tours [13, 15, 17]. Consequently, these repair algorithms affect the GA by changing the pure process of GA. Rather than proposing a repair algorithm that will cause a change in pure GA, a double-chromosome approach is proposed in this article. The proposed approach could also be applied to problems similar to the TSP.

This report is organized in four sections. After Introduction, the proposed methodology is discussed. In Sect. 3, a discussion of the proposed method as applied to different problems and the results are presented. Finally, in Sect. 4, the conclusion of the study is presented.

2 Methodology

A variety of GA approaches have been used to solve the TSP problem [8]. However, in those algorithms, chromosomes are based on path representation that could hardly be directly optimized through the GA approach. A new form of chromosome is, therefore, presented in this article that could be simply optimized through the GA approach.

2.1 A double-chromosome representation

Firstly, similar to the path representation, one chromosome is generated to represent a legal tour. This chromosome will be referred to as the map chromosome. Next, a new population of chromosomes are generated, each of them with the possibility of being attached to the map chromosome, thus creating a double chromosome. The new population of chromosomes will be referred to as guide chromosomes. The guide chromosomes are used to rearrange the map chromosome. In the proposed guide chromosomes, each pair of genes will determine which two cities in the map chromosome should be swapped. The guide chromosomes thus act in a way to sort the map chromosome to achieve the shortest tour. As the guide chromosome contains a set of genes, the length of it will always be an even number. Each of the two connected elements can be considered as a pair to be used to swap the position of two cities in the map chromosome. As illustrated in Table 1, this implies that, for a double-chromosome method, there will be one main path (map chromosome) and a population of guide chromosomes. For instance, in Table 1, the first guide chromosome [1, 2, 5, 6] indicates that in the map chromosome, first, $c1$ and $c2$ and after that $c5$ and $c6$ should be swapped.

The guide chromosomes have the advantage to be optimized through the GA without requiring any repair algorithms. The crossover and mutation, pure GA operators, can be applied to the selected guide chromosomes to generate a new population of guide chromosomes. The new guide chromosomes are applied to the map chromosome, and as a result, always a legal tour is generated. Instead of a population of maps, one map will be considered and a population of guide chromosomes will be generated (Fig. 1).

The length of the guide chromosomes depends on the number of cities. By considering α as the number of the set of genes in the guide chromosome ($\alpha > 1$), the maximum number of tours that could be generated by the guide chromosome from a map chromosome can be calculated as:

$$\tau = \left(C_{n,2}\right)^{\alpha} = \left(\frac{n!}{2!(n-2)!}\right)^{\alpha} \tag{2}$$

Table 1 Proposed map and guide chromosomes for a TSP with 7 cities

Map	Guide chromosomes, first generation	Generated maps, based on guide chromosomes
[c1,c2,c3,c4,c5,c6,c7]	[1, 2, 5, 6]	[c2,c1,c3,c4,c6,c5,c7]
	[1, 2, 5, 7]	[c7,c5,c3,c4,c2,c6,c1]
	[1, 5, 7]	[c1,c2,c3,c4,c7,c6,c5]

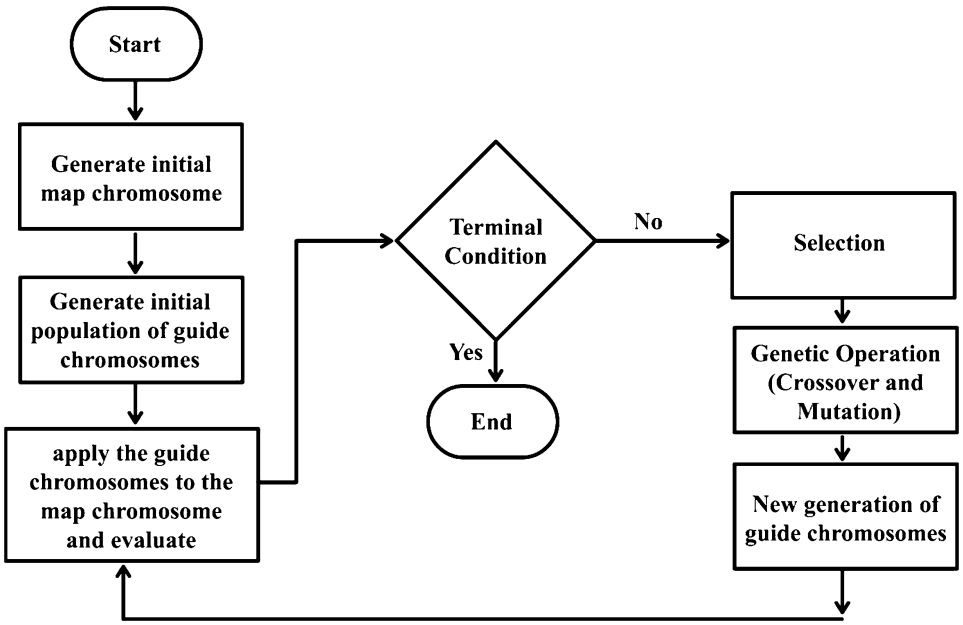

Fig. 1 Flowchart of the proposed method. In the process of selection, elitism, roulette-wheel, or tournament selection can be employed

where τ is the maximum number of tours. The maximum number of tours calculated through Eq. (2) contains repeated tours that can be eliminated. The minimum unique tours could therefore be calculated using Eq. (3) where indeed there are unique tours that are not considered:

$$\varphi = \prod_{i=0}^{2\alpha} \left(C_{n-2i,2}\right), n - 2i > 2 \tag{3}$$

where φ is the minimum number of unique tours. The exact number of unique tours is, in fact, within the interval of $[\varphi, \tau]$. For example, for a TSP with 5 unique cities and a guide chromosome with a length of 4 genes ($\alpha = 2$) the maximum number of tours could be calculated through Eq. (2) and equals 100, and the minimum number of unique tours could be calculated through Eq. (3) and equals 30. However, the exact number of unique tours that could be generated from a map containing 5 cities and guide chromosomes with a length of 4 genes is 43.

Although there are maps that with lesser length of guide chromosome could converge toward the shortest tour, Eq. (3) is employed to obtain the required length of the guide chromosomes to ensure that always the shortest tour will be obtained. As it was mentioned earlier, the total number of unique tours for a TSP with n cities can be obtained through Eq. (1). Hence, for an α where Eq. (3) is equal or bigger than Eq. (1), one could make sure that with guide chromosomes of the length of 2α, all possible tours for the TSP with n cities could be generated from a single map:

$$\left[\prod_{i=0}^{2\alpha} \left(C_{n-2i,2}\right), n - 2i > 2\right] \geq n!. \tag{4}$$

The constrain $n - 2i > 2$ plays the main role in Eq. (4). Through this constrain, it can be estimated that α should be equal to the number of cities. However, for an odd number of cities, $n+1$ should be used. Therefore, in general, α is considered to be equal to $n+1$. For instance, for a problem with 9 cities, α should be equal to 10 leading to the length of the guide chromosomes to be 20 genes.

2.2 Genetic algorithm operators

The main advantage of the proposed double-chromosome approach is that it could simply be optimized through all traditional and more recent genetic algorithm operators and the offspring will always be a legal tour. The operators that have been used to test the proposed algorithm are reviewed here.

2.2.1 New mutation operator

Mutation operators are typically used to modify one or more gene values in a chromosome. There are a variety of mutation operators that could be applied to a chromosome. A random replacement is one of the mutation operators that has been widely used and easily applied to chromosomes where the values of the genes are integers. In the random replacement, a random gene is selected and its value is changed with a random value between lower and upper bounds [16]. In the proposed guide

chromosome, the lower and upper bounds are considered as 1 and *n*, respectively.

In the process of generating guide chromosomes, the values of the genes are generated randomly. Thus, applying the generated guide chromosomes to a map chromosome can lead to unnecessary swaps decreasing the accuracy of the results. To avoid unnecessary swaps, in addition to the mentioned random replacement mutation operator, a new mutation operator is defined. In the new mutation operator, as shown in Fig. 2, a random pair of genes is selected and the values are equaled to each other. This means the modified genes will no longer affect the map chromosome and the order of the visited cities will not change.

2.2.2 Crossover

In this study, simple and well-known crossover operators, namely single-point crossover, two-point crossover, and uniform crossover, have been employed. Simple crossover operators have been selected to test the applicability of the proposed method. Recent and more advanced crossover operators, like but not limited to mixed crossover [19], parent centric crossover [20], sequential crossover, and random mixed crossover [21], could be used to increase the capacity, reliability, and accuracy of the proposed method.

3 Results and discussion

The proposed method was implemented in Python 3.6 on a computer with Intel(R) Core i5-4570 CPU @ 3.2 GHz with 8 GB RAM. To investigate the accuracy of the proposed method, three types of examples were used. First, the proposed method was compared with BFA. TSPs containing 10, 11, 12, 13, and 14 cities were randomly generated, and for each number of cities 9, different problems were generated. The generated TSPs were solved through BFA. For instance, for TSPs containing 14 cities, all 3,113,510,400 possible tours were checked and the best tour was selected. The process took 37 h for each problem. Next, bigger TSPs were considered. As it was impossible to solve bigger TSPs by BFA, two geometric shapes, a circle and a square, were considered and the cities were distributed over the perimeter of these two geometric shapes. Simply put, the perimeter of the shapes represents the best solution to the problems. Although the geometric-shape TSPs are easiest to be optimized, they are among complex problems for artificial intelligence (AI) approaches like GA. The geometric-shape TSPs may be considered as effective problems that could be used to evaluate the approaches that solve TSPs independent of their shape. Moreover, the applicability of the proposed method was tested by three well-known TSPs with the shortest tour as reported in the literature.

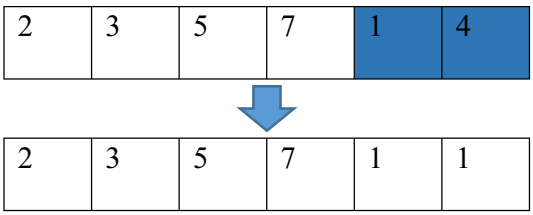

Fig. 2 The new proposed mutation operator has been applied to a sample guide chromosome. The top chromosome in the figure is a random guide chromosome that has been selected, and the new mutation operator has been applied to the last set of genes. The values are modified to be equal to each other. By applying this chromosome to the map chromosome, in the map chromosome, only genes 2 and 3, and 5 and 7 will be swapped, respectively

3.1 Proposed method applied to small-size TSPs

Forty-five different problems containing 10, 11, 12, 13, and 14 cities were randomly generated (9 different problems for each number of cities). The proposed method was applied to all the 45 problems 10 times. The aim was to check if the proposed method was able to solve TSPs and find the shortest tour. The population size for guide chromosomes was considered to be 400. The result, as illustrated in Fig. 3 (the figure contains only TSPs with 14 cities), indicates that for all 45 TSPs the proposed method was able to obtain the shortest path each and every time.

Based on Eq. (1) for each problem, the total number of possible unique tours (search space) has been calculated. The average number of iterations required to obtain the shortest tour is shown in Table 2. For each tour, the average number of paths that was checked is calculated as: average number of iterations × population size.

As it can be seen in Table 2, on average, the proposed method was able to find the shortest tour by checking only 1.02% of the total possible tours. The results obtained here indicate that the proposed method has high rate of convergence toward the shortest tour. Moreover, these results were obtained with basic crossover and mutation operators. More advanced operators may improve these results.

3.2 Proposed method applied to geometric shapes

It is almost impossible to solve TSPs with more than 14 cities through direct methods using current computers. Therefore, to test the accuracy of the proposed method, three TSPs were generated in a geometric shape where mathematically the optimized solution could be obtained.

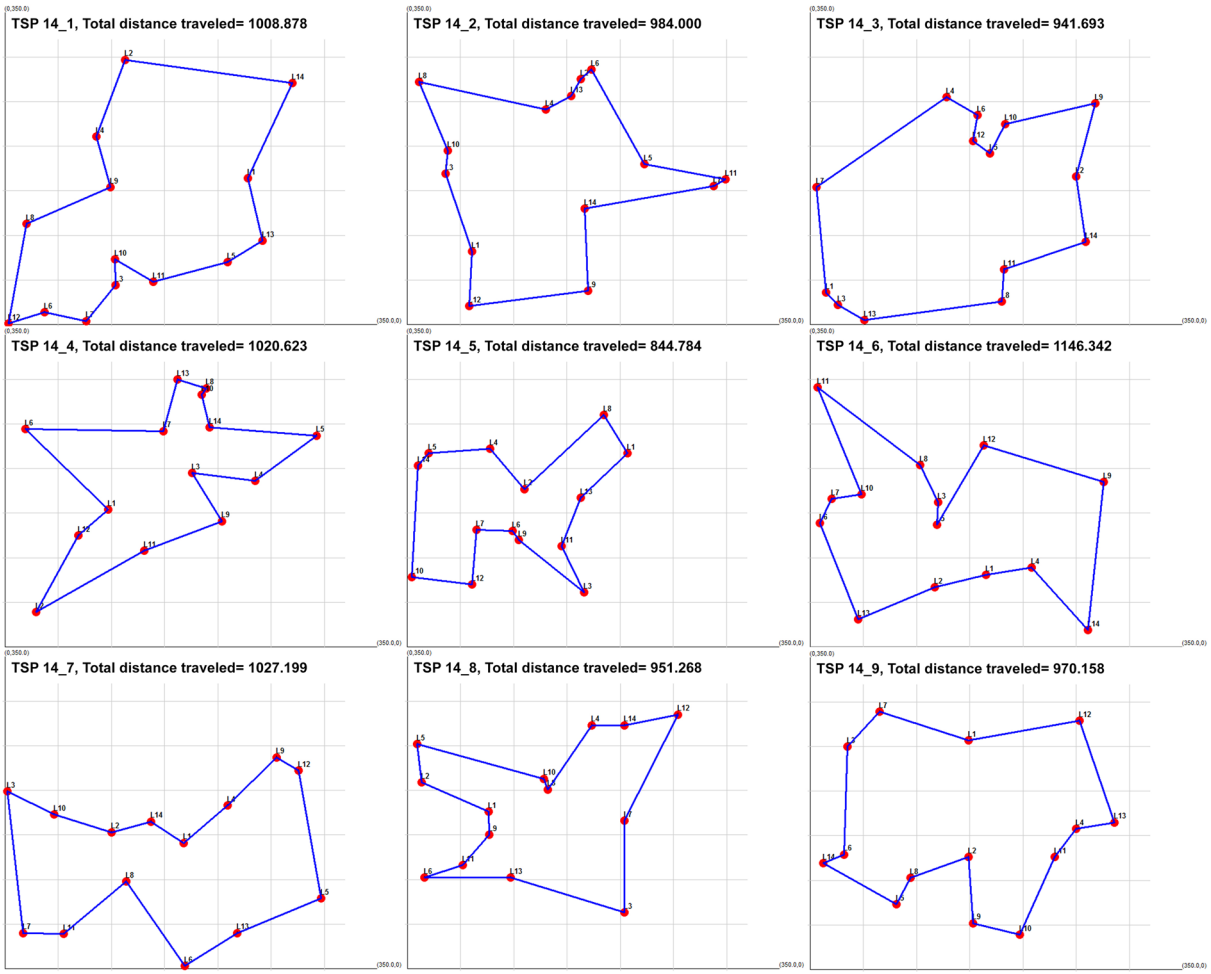

Fig. 3 The shortest path obtained by the proposed method for TSP with 14 cities. The dots in the figure show the location of the cities

Table 2 Average number of iterations required to obtain the shortest tour

Problem size (cities)	Population size	α	Guide chromosome length	Search space	Average iterations	Average paths checked (%)
14	400	15	30	3,113,510,400	2993	0.04
13	400	14	28	239,500,800	885	0.15
12	400	13	26	19,958,400	553	1.11
11	400	12	24	1,814,400	72	1.59
10	400	11	22	181,440	10	2.20
					Average	1.02%

There are a variety of methods such as the nearest neighbor algorithm (NNA) that could be applied to TSPs with geometric shape and obtain the optimized tour instantly. However, for AI algorithms, where the aim is to optimize the problem without considering its shape, these types of problems may be classified as complex problems. As illustrated in Fig. 4, the problems that are considered here are a circle with 60 cities with a radius of 150 and two squares with 24 and 100 cities where the side length equals 300. The perimeter of these shapes can be simply calculated and represents the optimized tour.

In the proposed method, the first step was to generate an initial map. For these geometric shapes, the initial map could then be the optimized path as the cities are connected to each other in a sequential order. As the aim was to test the accuracy of the proposed method,

the map chromosome was shuffled randomly to avoid any fast convergence toward the optimized solution. The main aim of genetic algorithm is to optimize problems with enormous search space. The results in Table 2 and Table 3 indicate that this aim has been satisfied. As it can be seen for bigger TSPs, the proposed method performs better and converges toward the shortest tour faster (in terms of the required number of iterations). To check the reliability of the proposed method, each problem has been solved 10 times and the average number of iterations has been calculated.

3.3 Proposed method applied to eil51, eil76, and st70

The proposed method was also applied to eil51, eil76, and st70, and the average results over 10 runs were obtained. These problems have also been investigated in the literature [15, 22, 23], and the shortest tour has been obtained. The proposed method was then applied to these three problems 10 times, and the average number of iterations required to find the shortest tour was obtained. The results are shown in Table 4.

4 Conclusions

There are many approaches used to solve TSPs with a high degree of accuracy. However, problems similar to the TSP cannot be solved through GA without any modifications. The solutions proposed in the literature include different repair algorithms that could be used along with GA to solve the TSP. The repair algorithms modify the GA process and make it possible to be applied to the TSP, but they lead

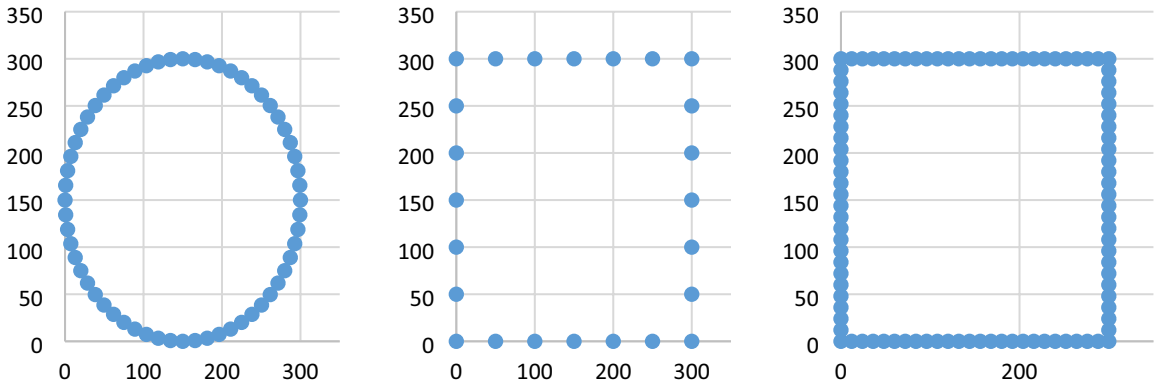

Fig. 4 TSPs with geometric shape, **a** 60 cities in the shape of a circle with radius of 150 and perimeter of 942, **b**, **c** 24 and 100 cities in the shape of a square with perimeter equal to 1200. The perimeter of each shape represents the optimized tour

Table 3 Average number of iterations required to obtain the shortest tour

Problem (size)	Population size	α	Guide chromosome length	Search space	Average iterations	Average paths checked (%)
Square (24)	1000	25	50	1.293E+22	213,612	0 (1.65E−12)
Circle (60)	1000	61	122	6.934e+79	120,055,213	0 (1.73E−72)
Square (100)	1000	101	202	4.666e+155	155,393,340	0 (3.33E−143)
					Average	0%

Table 4 Average number of iterations required to obtain the shortest tour

Problem (size)	Population size	α	Guide chromosome length	Search space	Average iterations	Average paths checked (%)
eil51 (51)	1000	52	104	1.521e+64	126,387,468	0 (8.31E−57)
eil76 (76)	1000	77	154	1.240e+109	113,919,145	0 (9.183E−102)
st70 (70)	1000	71	142	8.556e+97	105,914,341	0 (1.24E−90)
					Average	0%

to a process that is no longer pure GA. The aim of the study reported in this article was to apply pure genetic algorithm to the TSP. To this end, a double-chromosome approach was proposed that could be simply optimized by pure GA operators, namely crossover and mutation. In the double-chromosome approach, one chromosome is considered as the map chromosome. The map chromosome is similar to path representation suggested in the literature, where each gene indicates the city that should be visited. Along with the map chromosome, a population of guide chromosomes are generated. The aim of guide chromosomes is to modify the map chromosome in the direction of optimized solution. Each guide chromosome contains a number of set genes that identify which genes should be swapped in the map chromosome. The GA operators are applied to the guide chromosomes and guide chromosomes are applied to the map chromosome and new paths are generated. The advantage of the proposed double-chromosome method is that the generated paths will always be a valid tour and there will be no need for any repair algorithms.

The proposed method was also applied to a variety of examples, and it was found that the proposed method holds high rate of convergence toward the shortest tour. It should, however, be noted that simple and basic selection method, mutation, and crossover operators were used. More advanced operators could also be employed to improve the results even more significantly.

Acknowledgements Author would like to thank Professor Umut Türker for his valuable comments and suggestions on an earlier draft of this article and Professor Mehdi Riazi for his careful reading and editing recommendations that have improved the text of the article.

Compliance with ethical standards

Conflict of interest The author declares that have no conflict of interest.

References

1. Garey MR, Johnson DS (1979) A guide to the theory of NP-completeness. W. H. Freeman & Co, New York
2. Gouveia L, Leitner M, Ruthmair M (2017) Extended formulations and branch-and-cut algorithms for the black-and-white traveling salesman problem. Eur J Oper Res 262(3):908–928
3. Michail O, Spirakis PG (2016) Traveling salesman problems in temporal graphs. Theoret Comput Sci 634:1–23
4. Rego C, Gamboa D, Glover F, Osterman C (2011) Traveling salesman problem heuristics: leading methods, implementations and latest advances. Eur J Oper Res 211(3):427–441
5. Rocki K, Suda R (2013) High performance GPU accelerated local optimization in TSP. In: IPDPSW IEEE 27th international, pp 1788–1796
6. Applegate DL, Bixby RE, Chvatal V, Cook WJ (2011) The traveling salesman problem: a computational study. Princeton University Press, Princeton
7. Ozden SG, Smith AE, Gue KR (2017) Solving large batches of traveling salesman problems with parallel and distributed computing. Comput Oper Res 85:87–96
8. Potvin JY (1996) Genetic algorithms for the traveling salesman problem. Ann Oper Res 63:337–370
9. Knox J (1994) Tabu search performance on the symmetric traveling salesman problem. Comput Oper Res 21(8):867–876
10. Lin S, Kernighan BW (1973) An effective heuristic algorithm for the traveling salesman problem. Oper Res 21(2):498–516
11. Helsgaun K (2000) An effective implementation of the Lin-Kernighan traveling salesman heuristic. Eur J Oper Res 126(1):106–130
12. Davis L (1985) Applying adaptive algorithms to epistatic domains. Int Joint Conf on Artif Intell 85:162–164
13. Goldberg DE, Lingle R (1985) Alleles, loci, and the traveling salesman problem. Proc Int Conf Genet Algorithms Their Appl 154:154–159
14. Larranaga P, Kuijpers CM, Murga RH, Inza I, Dizdarevic S (1999) Genetic algorithms for the travelling salesman problem: a review of representations and operators. Artif Intell Rev 13(2):129–170
15. Ray SS, Bandyopadhyay S, Pal SK (2007) Genetic operators for combinatorial optimization in TSP and microarray gene ordering. Appl Intell 26(3):183–195
16. Sivanandam SN, Deepa SN (2007) Introduction to genetic algorithms. Springer, Berlin
17. Vahdati G, Yaghoubi M, Poostchi M (2009) A new approach to solve traveling salesman problem using genetic algorithm based on heuristic crossover and mutation operator. In: IEEE international conference soft computing and pattern recognition, 2009. SOCPAR'09. pp 112–116
18. Riazi A, Türker U (2018) A genetic algorithm-based search space splitting pattern and its application in hydraulic and coastal engineering problems. Neural Comput Appl 30(12):3603–3612
19. Hasançebi O, Erbatur F (2000) Evaluation of crossover techniques in genetic algorithm based optimum structural design. Comput Struct 78(1–3):435–448
20. Deb K, Anand A, Joshi D (2002) A computationally efficient evolutionary algorithm for real-parameter optimization. Evol Comput 10(4):371–395
21. Kaya M (2011) The effects of two new crossover operators on genetic algorithm performance. Appl Soft Comput 11(1):881–890
22. Jiao L, Wang L (2000) A novel genetic algorithm based on immunity. IEEE Trans Syst Man Cybern Part A 30(5):552–561
23. Tsai CF, Tsai CW, Yang T (2002) A modified multiple-searching method to genetic algorithms for solving traveling salesman problem. IEEE Int Conf Syst Man Cybern 3:6–9