An Effective Genetic Algorithm for Large-Scale Traveling Salesman Problems

Son Duy Dao, Kazem Abhary, and Romeo Marian

Abstract—**Traveling salesman problem (TSP) is an important optimization problem in many fields such as mathematics, computer science, engineering, bioinformatics, operations research, etc. In this paper, an effective Genetic Algorithm (GA) is developed to solve large-scale TSP. The proposed GA with three innovations, i.e. generating good initial population by considering the city locations, doing the crossover in crossover window and doing the mutation in mutation window, can deal with large-scale TSP very effectively. Effectiveness of the proposed GA is demonstrated through five case studies, ranging from small-scale to large-scale TSP (up to 13509 cities).**

Index Terms—**Genetic algorithm, large-scale traveling salesman problem, crossover window, mutation window**

I. Introduction

TRAVELING salesman problem (TSP) is a very well-known combinatorial optimization problem, which has been attracting a lot of attention from researchers in various fields such as mathematics, computer science, engineering, bioinformatics, supply chain, operations research, etc. [1, 2]. In general, TSP can be stated as follows. There is a list of cities with a given locations; and a salesman wants to find the cheapest tour in terms of distance, time, money, etc. to visit all of the cities, each exactly once, and then return to the city of origin [3]. An effective solution method for TSP has an important application in many areas such as transportation, logistics, biology, chemotaxis, computer science, engineering, etc. [4].

Finding the optimal solution to a small-scale TSP is very easy, but for a large-scale TSP, it is very difficult, most of the times impossible, due to extremely large search space involved. From computational complexity theory point of view, TSP is a NP-hard problem [5]. It should be noted that NP is a technical term in computational complexity theory in computer science and mathematics, which stands for Non-deterministic Polynomial-time. NP problems are decision problems that can be solved by non-deterministic polynomial-time bounded Turing machines [6]. In addition, NP-hard problems are decision problems which are as hard as any NP problem [7]. Moreover, NP-hard problems are algorithmically solvable but computationally intractable [7]. There is no exact method that can find the global optimal solutions to NP-hard problems in polynomial time, and fast meta-heuristics are the popular approaches to search for high-quality/practical solutions to the problems [8].

Genetic Algorithm (GA) is a popular meta-heuristic method, often used to solve complex large-scale optimization problems [9]. In this paper, an effective GA with innovative chromosome, crossover and mutation is developed to solve large-scale TSP. Effectiveness of the proposed GA is demonstrated through a number of comprehensive case studies.

II. Literature Review

TSP is one of the oldest optimization problems in the field of operations research; it has been the subject of intensive study for more than three decades [4]. A lot of progress in solving TSP has been made so far, due to the development of new optimization algorithms and computing techniques. Nevertheless, solving large-scale TSP is still very challenging. According to Robert Bosch [10], finding an optimal solution to the 100,000-city Mona Lisa problem would set a new world record for TSP. The existing best-known solution to the problem, with the total distance of 5,757,191, was found by Yuichi Nagata in July 27, 2012, which was achieved after 11.5 CPU years of computing; however, it is still not the optimal solution. To help perk up interest in searching for a better solution to the 100,000-city Mona Lisa problem, Robert Bosch [10] is offering a $1,000 prize to the first person who finds a solution with the total distance shorter than 5,757,191. There is no doubt that solving large-scale TSP is still a challenging and open problem.

Like other NP-hard problems, TSP is computationally intractable [7]. Solution methods for TSP can be generally classified into two groups: exact methods and meta-heuristic methods. On one hand, exact methods, e.g. branch and bound, dynamic programming, linear and integer programming, etc., can find the optimal solution to TSP but they are computationally expensive; and therefore, exact methods are not always feasible for solving large-scale TSP [11]. On the other hand, meta-heuristic methods are not capable of guaranteeing the optimal solution to TSP but they can find good solutions to any TSP pretty quickly; that is why meta-heuristic methods are the preferred choices for solving large-scale TSP [11-13].

GA is one of the most popular meta-heuristic methods, often used to solve large-scale optimization problems [9].

Manuscript received: June 24, 2016, revised: July 18, 2016. This work was supported by Australian Government in form of Endeavour Postgraduate Scholarship.

Mr Son Duy Dao is a PhD student at School of Engineering, University of South Australia, Australia (corresponding author to provide e-mail: son.dao@mymail.unisa.edu.au).

Prof. Kazem Abhary is with School of Engineering, University of South Australia, Australia.

Dr. Romeo Marian is with School of Engineering, University of South Australia, Australia.

Solving TSP using GA is certainly not new in the literature. To date, there have been a significant number of publications, in which GA based methods were developed to solve many different versions of TSP. Due to space limit, only a few, most related and recent publications are reviewed herein. In the research of Wang et al. [14], TSP was solved by a so called multi-offspring GA, and some small and medium-scale TSP instances were used to test the proposed algorithm. Nagata & Soler [15] have developed an innovative GA for TSP, in which a local search procedure was used for generating the initial population, and the developed GA was comprehensively tested by a large number of small-scale TSP instances. In addition, a GA with a new mutation operator named greedy sub tour mutation was proposed by Albayrak & Allahverdi [16] and the proposed algorithm was tested with a number of small-scale TSP instances. In the work of Maity, Roy & Maiti [17], a modified GA with new selection and crossover was developed and tested with 11 small-scale TSP instances. Moreover, a hybrid GA with two local optimization strategies [18] and an improved GA with new crossover operator called two-part chromosome crossover [19] have been developed for TSP, and the developed algorithms were tested with small and medium-scale TSP instances. Finally, Paul et al. [9] have investigated the effectiveness of different population seeding techniques for permutation-coded GA, with a number of large-scale TSP instances.

It can be seen that the research on developing GA to solve large-scale TSP is still very limited. Almost all published works deal with small and medium-scale TSP (the scale with less than or equal to 3038 cities). To the authors' best knowledge, there has been only one research done by Paul et al. [9], dealing with large-scale TSP (the scale with up to 18512 cities). Nevertheless, the research of Paul et al. [9] mainly studied several seeding techniques for generating initial population of permutation-coded GA; effective crossover and mutation strategies of the GA to deal with large-scale TSP were not developed there. To overcome these limitations, an effective GA with innovative strategies for (1) initial population, (2) crossover and (3) mutation is developed herein to solve large-scale TSP. Effectiveness of the proposed GA is demonstrated through five case studies, ranging from small-scale to large-scale TSP (up to 13509 cities).

III. Proposed Genetic Algorithm

Like the traditional GA, the proposed GA has five components, namely chromosome encoding, crossover, mutation, evaluation and selection. The connection of these five components, called GA structure, is shown in Fig. 1; details of the components are described in the subsequent Sections.

A. Chromosome Encoding

A string of positive integer numbers as illustrated in Table 1 is a natural way to encode a solution to TSP. In Table 1, the positive integer numbers represent the corresponding cities. For example, numbers 4 and 9 represent the cities 4 and 9, respectively. The sequence of the numbers represents the sequence of the cities in the corresponding tour, e.g. the illustrated chromosome in Table 1 encodes the following tour: 4 → 9 → 2 → 1 → 10 → 14 → 5 → 12 → 8 → 6 → 15 → 11 → 13 → 7 → 3 → 4.

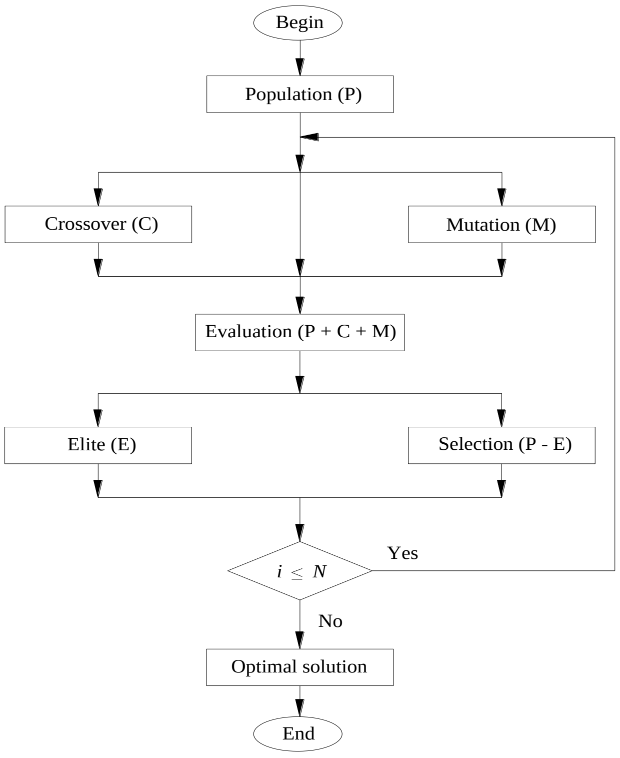

Fig. 1: Structure of the proposed GA

Table 1: Chromosome

4	9	2	1	10	14	5	12	8	6	15	11	13	7	3

Normally, chromosomes in the first generation of a GA, called initial population, are randomly generated. However, this approach is not very effective when solving TSP, because it does not take advantage of location information of the cities. The city location information in TSP is always available and it is valuable to generate better chromosomes that will serve as starting points of GA search. A good initial population plays an important role in performance of a GA, especially when solving large-scale TSP. In this paper, an innovative approach for generating a good initial population is developed. The fundamental idea of the proposed approach is generating a chromosome in which two adjacent genes should contain two cities that are very close, sometimes but not always closest, to each other. The proposed approach is implemented by the following steps.

Step 1: Sort the cities according to the increase of their *x* coordinate (named matrix A).
Step 2: Sort the cities according to the increase of their *y* coordinate (named matrix B).
Step 3: Randomly select one city and assign it to the first gene of the chromosome.
Step 4: Assign the selected city as the seed.
Step 5: Incrementally and symmetrically expand the segments starting from the seed in matrices A and B as illustrated in Fig. 2, until the two segments have at least one city, except the seed, in common.

Step 6: Determine the common city/cities, except the seed, in two segments in Step 5.
Step 7: Assign the common city in Step 6 to the next gene of the chromosome. If there is more than one common city in Step 6, they are assigned in the same manner but in a random order.
Step 8: Select the city in the last gene, filled so far, of the chromosome, for example, city 14 as illustrated in Fig. 2.
Step 9: Remove all common cities (including the seed) in two segments in Step 5, but except the selected city in Step 8, from matrices A and B.
Step 10: Assign the selected city in Step 8 as the new seed.
Step 11: Repeat Steps 5-10 until all cities have been assigned to all genes of the chromosome.

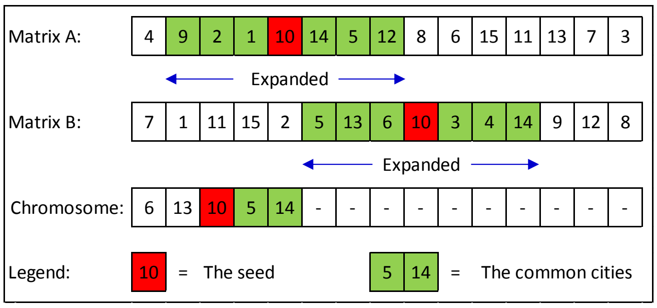

Fig. 2: Proposed chromosome generating procedure

It is noted that the segment(s) in Step 5 may not be symmetrically expanded when the seed is located at the position which is very close to the first or last element of matrices A and/or B. In those cases, the segment(s) will be asymmetrically expanded and the lengths of the two segments must be still the same.

B. Crossover

The traditional crossover [20] is not very effective when solving large-scale TSP, due to extremely large search space of the problem. An innovative crossover is therefore proposed herein. The proposed crossover is implemented by the following steps.

Step 1: Randomly select one chromosome called parent.
Step 2: Randomly select one segment containing a certain number of genes, which is called crossover window, as illustrated in Fig. 3.
Step 3: Randomly select one cut point to divide the selected crossover window into two pieces as highlighted in Fig. 3.
Step 4: Swap the two pieces as illustrated in Fig. 3.

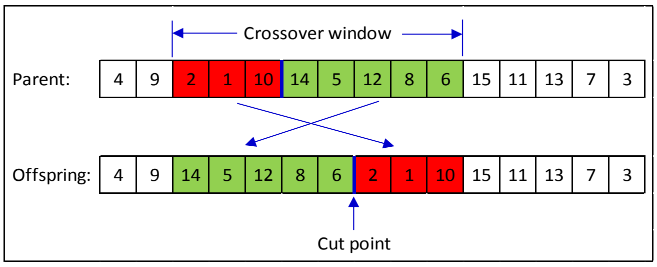

Fig. 3: Proposed crossover

It is noted that the length of crossover windows, measured in term of gene quantity, is a constant selected by the users. In addition, each crossover operation has only one input (one parent) and one output (one offspring). Finally, the offspring chromosome is always feasible and therefore no further repair is required.

C. Mutation

Like the traditional crossover, the traditional mutation [20] is not very effective when dealing with large-scale TSP. An innovative mutation is therefore developed herein. The developed mutation is described as follows.

Step 1: Randomly select one chromosome called parent.
Step 2: Randomly select one segment containing a certain number of genes, which is called mutation window, as illustrated in Fig. 4.
Step 3: Randomly select two different genes in the selected mutation window, for example, two genes containing two cities 1 and 12 as shown in Fig. 4.
Step 4: Exchange the two selected genes as illustrated in Fig. 4.

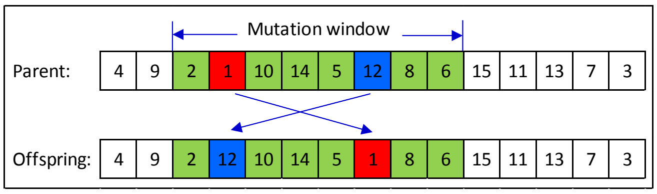

Fig. 4: Proposed mutation

It is noted that like crossover windows, mutation windows have a constant length, measured in term of gene quantity and selected by the users. Each mutation operation has only one input (one parent) and one output (one offspring). In addition, no further repair is required because the offspring chromosome is always feasible.

D. Evaluation

The objective function of TSP, to be minimized, is the total distance of the travelling tour. The distance between two cities is calculated by Eq. 1, where x_i, y_i, x_j and y_j are the coordinates of cities i and j. Quality of a chromosome is evaluated through its total travelling tour distance. A chromosome with a better quality is more likely to be selected for the next generation. How to do the selection will be presented in the next Section.

$$D_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (1)$$

E. Selection

The Roulette Wheel selection, one of the popular selection methods in the literature, is adopted herein. The Roulette Wheel method selects a new population based on the probability distribution associated with the fitness values or qualities of chromosomes [20]. In this paper, the elite strategy is also used in the selection process, in which the best chromosome in one generation is always guaranteed to pass to the next generation.

Effectiveness of the proposed GA will be demonstrated through five case studies in the next Section.

IV. Case Studies

A. Problem Description

Five TSP instances from an online library named TSPLIB [21] are used herein to verify the robustness of the proposed GA. The five TSP instances are Ulysses22, Gr202, U2152, Pla7397 and Usa13509. Due to space limit, full dataset of the instances are not presented in this paper. For the dataset, it is advised to refer to the online library at TSPLIB [21]. The number of cities in each instance is shown in Table 2. To solve these five TSP instances, the proposed GA and the traditional GA were applied; performances of the two approaches will be reported in the next Section.

B. Result and Discussion

Qualities of solutions to the five TSP instances, obtained by the proposed GA and the traditional GA in the same computing time of 90 minutes, are compared to each other. Each algorithm was independently run for 10 times, and its solution quality in terms of the best fitness value (called Best), average fitness value (called Mean) and standard deviation of the fitness values (called Std.deviation) is shown in Table 2.

The traditional GA is exactly the same as the proposed GA, except two following aspects. First, the initial population of the traditional GA is randomly generated. Second, crossover and mutation of the traditional GA do not have two strategies developed in this paper, i.e. crossover window and mutation window, respectively. Parameters of the two algorithms were also set exactly the same, as shown in Table 3, where P, C, M and L are population size, number of chromosomes to be crossed in each generation, number of chromosomes to be mutated in each generation and length of crossover/mutation window, respectively. It should be noted that L value of the traditional GA is not available as there is no crossover/mutation window in the traditional GA.

Table 3: Parameters of two algorithms

	P	C	M	L
Proposed GA	50	40	10	20
Traditional GA	50	40	10	-

As can be seen from Table 2, for small-scale problem, i.e. Ulysses22, fitness values obtained by the proposed GA and the traditional GA are about the same. For larger problems, the proposed GA outperforms the traditional GA in all three measures: Best, Mean and Std.deviation. The Mean values achieved by the two algorithms are visualized in Fig. 5. It is noted that Logarithmic scale (base: 10) was used to scale up/down the vertical axis of Fig. 5, for better visualization. As can be seen from Fig. 5, for large-scale problems, the proposed GA can provide much better solutions, compared to the traditional GA. It can be concluded that although the proposed GA, like other meta-heuristics, is not capable of guaranteeing the global optimal solution for large-scale TSP, it is very effective in finding the good solutions in a reasonable computing time.

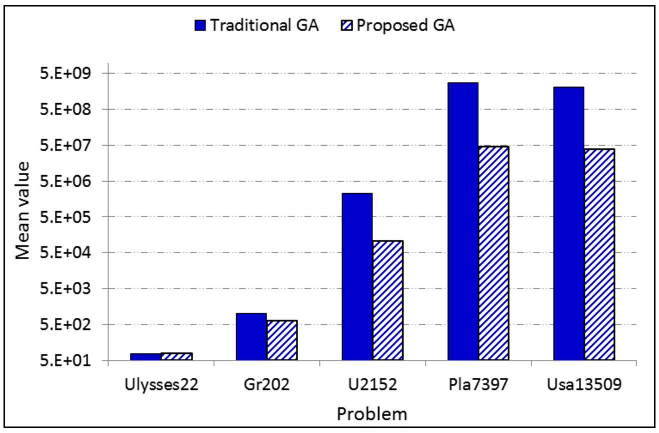

Fig. 5: Mean value of the achieved tour distances

Table 2: Performance comparison

No.	Problem (*)	No. of cities	Computing time (min)	Fitness value	Traditional GA	Proposed GA
1	Ulysses22	22	90	Best	76.1	75.3
				Mean	76.9	76.4
				Std.deviation	1.1	0.9
2	Gr202	202	90	Best	946.7	582.8
				Mean	1014.8	619.7
				Std.deviation	46.8	27.5
3	U2152	2152	90	Best	2259986.9	100817.2
				Mean	2295021.4	107023.2
				Std.deviation	26024.6	3071.7
4	Pla7397	7397	90	Best	2729030057.2	41500787.8
				Mean	2740286300.8	44261679.3
				Std.deviation	6514995.7	2388918.1
5	Usa13509	13509	90	Best	2123360930.3	37428783.1
				Mean	2132622914.5	38516350.3
				Std.deviation	5266762.9	503770.4

** Note: The problems are from an online library named TSPLIB [21]*

V. Conclusion

In this paper, an effective GA for solving large-scale TSP has been developed. With three novel features, i.e. generating good initial population by considering the city locations, doing the crossover in crossover window and doing the mutation in mutation window, the proposed GA is capable of dealing with large-scale TSP. The performances of the proposed GA have been compared with that of the traditional GA in five case studies, ranging from small-scale to large-scale TSP (up to 13509 cities). For small-scale TSP, the performances of the proposed GA and the traditional GA were about the same. For large-scale TSP, the proposed GA could provide much better solutions, compared to the traditional GA. There is no doubt that although the proposed GA, like other meta-heuristics, is not capable of guaranteeing the global optimal solution for large-scale TSP, it is very effective in finding the good solutions in a reasonable computing time.

In future work, more comprehensive tests and comparison will be carried out to thoroughly verify the effectiveness of the proposed GA.

References

[1] Huai-Kuang, T., et al., *An evolutionary algorithm for large traveling salesman problems.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2004. **34**(4): p. 1718-1729.

[2] Gutin, G. and A. Punnen, *The traveling salesman problem.* Discrete Optimization, 2006. **3**(1): p. 1.

[3] Nguyen, H.D., et al., *Implementation of an effective hybrid GA for large-scale traveling salesman problems.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2007. **37**(1): p. 92-99.

[4] Avşar, B. and D.E. Aliabadi, *Parallelized neural network system for solving Euclidean traveling salesman problem.* Applied Soft Computing, 2015. **34**: p. 862-873.

[5] Wang, Y., *An approximate method to compute a sparse graph for traveling salesman problem.* Expert Systems with Applications, 2015. **42**(12): p. 5150-5162.

[6] Cadoli, M., et al., *NP-SPEC: an executable specification language for solving all problems in NP.* Computer Languages, 2000. **26**(2–4): p. 165-195.

[7] Shapiro, M. and E. Delgado-Eckert, *Finding the probability of infection in an SIR network is NP-Hard.* Mathematical Biosciences, 2012. **240**(2): p. 77-84.

[8] He, K., W. Huang, and Y. Jin, *An efficient deterministic heuristic for two-dimensional rectangular packing.* Computers & Operations Research, 2012. **39**(7): p. 1355-1363.

[9] Paul, P.V., et al., *Performance analyses over population seeding techniques of the permutation-coded genetic algorithm: An empirical study based on traveling salesman problems.* Applied Soft Computing, 2015. **32**: p. 383-402.

[10] Bosch, R. *Mona Lisa TSP challenge*. 2009; Available from: http://www.math.uwaterloo.ca/tsp/data/ml/monalisa.html.

[11] Potvin, J.Y., *Genetic algorithms for the traveling salesman problem.* Annals of Operations Research, 1996. **63**(3): p. 337-370.

[12] Dao, S.D. and R. Marian, *Genetic algorithms for integrated optimisation of precedence-constrained production sequencing and scheduling*, in *Electrical Engineering and Intelligent Systems* S.-I. Ao and L. Gelman, Editors. 2013, Springer: New York. p. 65-80.

[13] Dao, S.D. and R. Marian, *Modeling and optimisation of precedence-constrained production sequencing and scheduling using multi-objective genetic algorithms*, in *The World Congress on Engineering* 2011: London, UK.

[14] Wang, J., et al., *Multi-offspring genetic algorithm and its application to the traveling salesman problem.* Applied Soft Computing, 2016. **43**: p. 415-423.

[15] Nagata, Y. and D. Soler, *A new genetic algorithm for the asymmetric traveling salesman problem.* Expert Systems with Applications, 2012. **39**(10): p. 8947-8953.

[16] Albayrak, M. and N. Allahverdi, *Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms.* Expert Systems with Applications, 2011. **38**(3): p. 1313-1320.

[17] Maity, S., A. Roy, and M. Maiti, *A modified genetic algorithm for solving uncertain constrained solid travelling salesman problems.* Computers & Industrial Engineering, 2015. **83**: p. 273-296.

[18] Wang, Y., *The hybrid genetic algorithm with two local optimization strategies for traveling salesman problem.* Computers & Industrial Engineering, 2014. **70**: p. 124-133.

[19] Yuan, S., et al., *A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms.* European Journal of Operational Research, 2013. **228**(1): p. 72-82.

[20] Gen, M. and R. Cheng, *Genetic algorithms and engineering design* 1997, New York John Wiley & Sons.

[21] TSPLIB. *Symmetric traveling salesman problem instances*. 1995 [viewed: 28 May 2016]; Available from: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html.