A comparison of GA crossover and mutation methods for the traveling salesman problem*

Robin T. Bye, Magnus Gribbestad, Ramesh Chandra, and Ottar L. Osen

Cyber-Physical Systems Laboratory
Department of ICT and Natural Sciences
NTNU—Norwegian University of Science and Technology
Postboks 1517, NO-6025 Ålesund, Norway
Email for correspondence: robin.t.bye@ntnu.no

Abstract. The traveling salesman problem is a very popular combinatorial optimization problem in fields such as computer science, operations research, mathematics, and optimization theory. Given a list of cities and the distances between any city to another, the objective of the problem is to find the optimal permutation (tour) in the sense of minimum traveled distance when visiting each city only once before returning to the starting city. Because many real-world problems can be modelled to fit this formulation, the traveling salesman problem has applications in challenges related to planning, routing, scheduling, manufacturing, logistics, and other domains. Moreover, the traveling salesman problem serves as a benchmark problem for optimization methods and algorithms, including the genetic algorithm. In this paper, we examine various implementations of the genetic algorithm for solving two examples of the traveling salesman problem. Specifically, we compare commonly employed methods of partially-mapped crossover and order crossover with an alternative encoding scheme that allows for single point, multipoint, and uniform crossover. In addition, we examine several mutation methods, including twors mutation, centre inverse mutation, reverse sequence mutation, and partial shuffle mutation. We empirically compare the implementations in terms of the chosen crossover and mutation methods to solve two benchmark variations of the traveling salesperson problem. The experimental results show that the genetic algorithm with order crossover and the centre inverse mutation method provides the best solution for the two test cases.

Keywords: TSP · Genetic algorithm · Crossover · Mutation · Permutations · Inversion sequence.

1 Introduction

The traveling salesman problem (TSP) is a very popular combinatorial optimization problem in fields such as computer science, operations research, mathematics, and optimization theory. In its most basic description, the TSP consists

* This research was supported by the European Research Consortium for Informatics and Mathematics (ERCIM).

of a list of cities and the distances between any city to another, where the objective of the problem is to find the optimal permutation (tour) in the sense of minimum traveled distance when visiting each city only once before returning to the starting city. Moreover, the decision version of the TSP, where one must deciding whether there exists any shorter tour than a given tour with some distance belongs to the class of NP-complete problems, meaning that "it is possible that the worst-case running time for any algorithm for the TSP increases superpolynomially (but no more than exponentially) with the number of cities."[1]

Many real-world problems can be modelled to fit the TSP formulation and hence, the problem has applications related to planning, routing, scheduling, manufacturing, logistics, and many other domains. Moreover, the traveling salesman problem serves as a benchmark problem for optimization methods and algorithms, including the genetic algorithm (GA).

The GA is an evolutionary algorithm for solving search and optimisation problems and is inspired by elements in natural evolution, such as inheritance, mutation, selection, and crossover (e.g., see [10]). A GA is robust, easy to implement, and easily applicable for multiobjective optimization problems, e.g., [1,2,11]. The GA is generally attributed to Holland (1975) [15], with subsequent popularisation by Goldberg (1989) [8], and is still a very popular optimisation tool across many different disciplines. In our Cyber-Physical Systems Laboratory (CPS Lab),[2] we have many years of experience utilizing the GA for a variety of purposes, including adaptive locomotion of a caterpillar-like robot [18], autonomous ships and dynamic positioning of tug vessels along the Norwegian coast (e.g., [6,7,3]), and as a core part of a generic software framework for intelligent computer-automated design (CautoD) [4], which was successfully applied for optimized CautoD, or virtual prototyping, of offshore cranes and winches (e.g., [13,14,5].

In this paper, we focus on solving the TSP using a GA implemented with a set of different combinations of crossover and mutation methods. For crossover, we use the partially-mapped crossover (PMX), order crossover (OX), single point crossover, multipoint crossover, and uniform crossover methods. We also test several mutation methods, namely twors mutation, centre inverse mutation (CIM), reverse sequence mutation (RSM), and partial shuffle mutation (PSM). Each combination of crossover and mutation methods are implemented and evaluated for two benchmark problems called Western Sahara (29 cities) and Djibouti (38 cities) obtained from the TSP website of the University of Waterloo, California, USA.[3]

We empirically evaluate the performance of the different GA implementations for solving the two TSP benchmark problems. The experimental results show that OX crossover with CIM mutation outperforms the other implementations.

The paper is organized as follows. Section 2 describes related work. Section 3 presents the various crossover and mutation methods used in the implementation

[1] Wikipedia: https://en.wikipedia.org/wiki/Travelling_salesman_problem
[2] https://www.ntnu.no/blogger/cpslab/
[3] http://www.math.uwaterloo.ca/tsp

of the GA for solving the two TSPs. Section 4 presents the results and an analysis of these. Section 5 discusses the results and the analysis. Finally, Section 5.1 concludes the paper and provides possible future directions.

2 Related work

Several common approaches exist for solving TSP problems, e.g., employing a GA [21], ant colony optimization [24], or artificial neural networks [20]. The TSP can have real-world application, e.g., the author in [12] developed a GA-based method for the TSP to find the optimal route for the Istanbul Electricity Tram and Tunnel Operations (IETT).

Variations of the GA has proved to be very successful in obtaining good results for solving TSPs (e.g., [17]). Authors in [11] employed a GA-based approach using several recombination operators, whilst Goldberg et al. [9] proposed an approach to improve the GA using a PMX operator. In [22], authors present a novel GA-based approach by introducing a new recombination operator to produce new offspring, whereas authors in [19] proposed a new hybrid GA in which the crossover operator is improved by utilizing the local search strategy. In yet another study [23], the authors used an improved GA by combining random crossover and dynamic mutation that provided better results as compared to the conventional genetic algorithm for TSP problem.

A more recent paper by Abid Hussain et al.[16] modify the crossover operator, examining PMX and OX and a new proposed operator. The authors apply the three crossover operators using TSP datasets for 42, 53 and 170 cities. Finally, in [25], Üçoluk proposed a method for alternative chromosome encoding that allows the GA to be solved without using permutation crossover methods. According to the author, this method is supposed to perform slightly worse than other methods in terms of the solution but many times faster[25].

3 Crossover and mutation methods

In this paper, we have implemented two different types of crossover methods: (i) two conventional crossover methods for permutation problems (PMX and OX), and (ii) three ordinary crossover methods (single-point, two-point, and uniform crossover) normally used for non-permutation problems enabled by the alternative encoding scheme suggested by Üçoluk [25].

3.1 Conventional crossover methods for permuation problems

We have implemented the PMX crossover method and the OX crossover method, both of which are standard crossover methods for GAs solving TSPs. Both methods introduce measures to avoid duplicates, which are not allowed in permutations.

PMX crossover The PMX crossover method is used for crossover in permutation problems. Being somewhat complex to explain purely in words, we resort to an example of PMX crossover provided by Ücoluk in his paper [25], and reproduced here in Fig 1. First, a single crossover point is randomly selected for

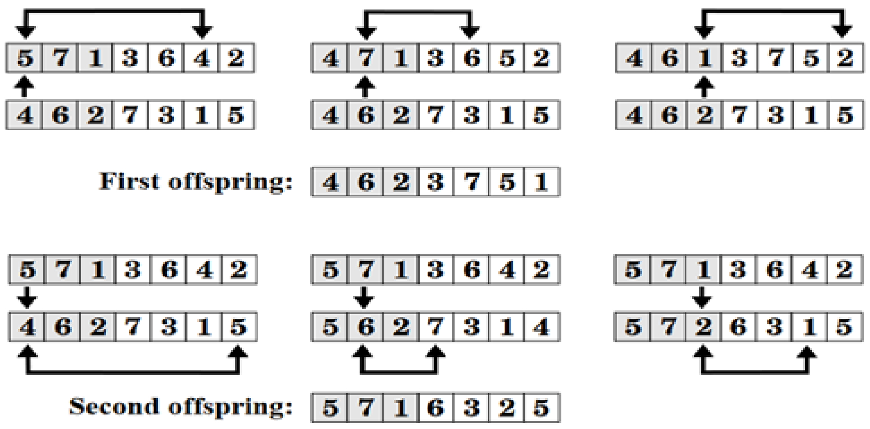
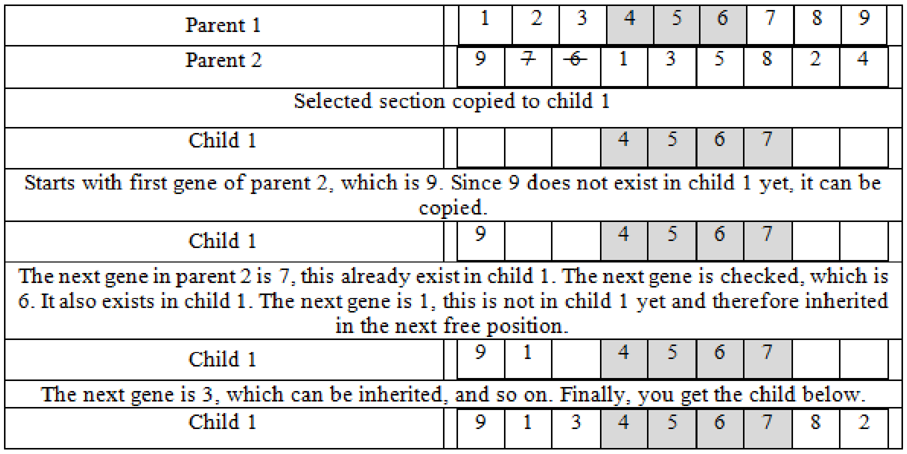


Fig. 1. Example of PMX crossover [25].

both parents, 5713642 and 4627315, corresponding to two permutations of cities 1 through 7. Next, beginning with a copy of the first parent, cities 462 before the crossover point in the second parent will take up the same gene positions in the first child. However, if merely copying these genes (cities), duplicates might occur, which is not allowed in a permutation. Hence, if there is a duplicate in the child occurring after the crossover point, this gene is replaced by the original city of the child occurring before the crossover point. In the example, putting city 4 in position 1 of the first city in the first child will lead to a duplicate because 4 also occurs in position 6. Hence, city 5 originally at position 1 is moved to position 6. Likewise for city 6 in position 2 and city 2 in position 3. The process repeats for the second child but with cities before the crossover point from the first parent (cities 571) taking up the same gene positions of the second child and the same replacement procedure to avoid duplicates.

OX Crossover Order 1 crossover (often referred to as OX or order crossover) is also a conventional crossover method for permutation problems. This method is based on randomly selecting a section of genes within the parents, for example, the 4 middle genes. Child 1 will then directly inherit these 4 middle genes from parent 1 (into the same position in the child), while child 2 will inherit from parent 2. The remaining genes are then filled with values from the other parent. Again, since chromosomes represent permutations, it is important that there are no duplicate genes (values) in the child. Therefore, creating the child starts with looking at the index of the first non-assigned gene in the other parent. If this gene value does not exist in the child, it is copied into the child. If the value already exists in the child, the procedure continues to check the next gene of the other parent. The process can be illustrated with the example in Fig.2.



Parent 1	1	2	3	4	5	6	7	8	9
Parent 2	9	~~7~~	~~6~~	1	3	5	8	2	4
Selected section copied to child 1									
Child 1				4	5	6	7		
Starts with first gene of parent 2, which is 9. Since 9 does not exist in child 1 yet, it can be copied.									
Child 1	9			4	5	6	7		
The next gene in parent 2 is 7, this already exist in child 1. The next gene is checked, which is 6. It also exists in child 1. The next gene is 1, this is not in child 1 yet and therefore inherited in the next free position.									
Child 1	9	1		4	5	6	7		
The next gene is 3, which can be inherited, and so on. Finally, you get the child below.									
Child 1	9	1	3	4	5	6	7	8	2

Fig. 2. Example of Order 1 crossover (OX).

Having generated one child, the process repeats with parent 1 becoming parent 2 and vice versa.

3.2 Ordinary crossover methods enabled by alternative encoding

Whilst standard crossover methods such as n-point crossover or uniform crossover do not take measures to avoid duplicate genes, they can still be used if chromosomes are encoded (transformed) using the encoding scheme presented by Üçoluk [25]. Here, we first give a short revision of the alternative encoding scheme, which enables use of the three ordinary crossover methods we have examined here, namely single-point crossover, two-point crossover, and uniform crossover.

Alternative chromosome encoding As explained by Üçoluk [25], a permutation can be represented by its inversion sequence.[4] Most conveniently, and different from the permutation itself, there is no restrictions on having duplicates in the inversion sequence. As a consequence, ordinary crossover operations can be applied to chromosomes encoded as inversion sequences, as long as the inversion sequences are decoded back to permutations for which the tour distance can be calculated.

[4] E.g., see Wikipedia: https://en.wikipedia.org/wiki/Inversion_(discrete_mathematics)

An example provided by Üçoluk [25] is presented in Fig. 3, which shows this alternative chromosome encoding combined with single-point crossover.

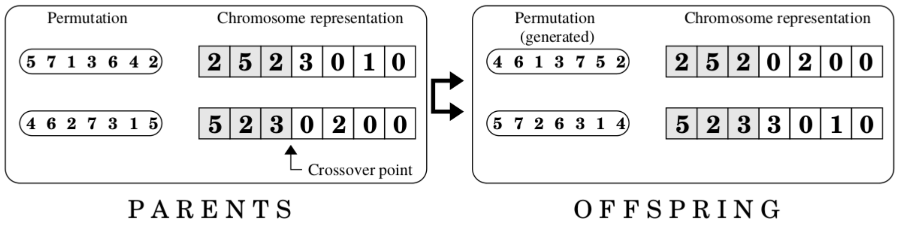


Fig. 3. Example of single-point crossover for chromosomes encoded as inversion sequences (adapted from [25]).

Single-point crossover Single-point crossover is a standard method for non-permuation problems, i.e., where duplicates are allowed. A random index point in the two parent chromosomes is chosen and both parents are split into two sections at this point. Child 1 takes the first section from parent 1 and the second section from parent 2. Child 2 takes the first section from parent 2 and the second section from parent 1. An example of single-point crossover in depicted in Fig. 4.

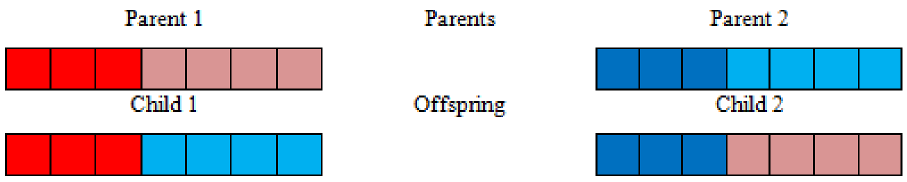


Fig. 4. Example of single-point crossover.

Two-point crossover Two-point crossover is similar to single-point crossover, but for this method two points are selected. This means that each parent is divided into three sections. The first child will then inherit the first and last section from parent 1, and the middle section from parent 2. The second child will inherit the first and last section from parent 2 and the middle section from parent 1 [1]. The procedure is illustrated in Fig. 5.

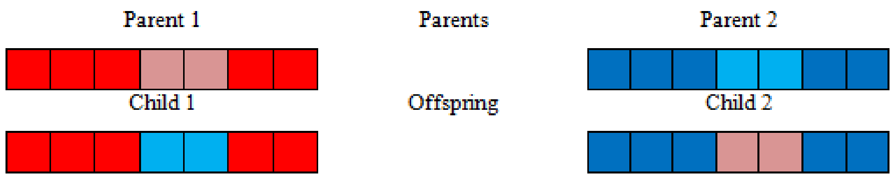


Fig. 5. Example of two-point crossover.

Uniform crossover Uniform crossover is different from the other methods so far. This method goes through every gene, and determines if it should be inherited from parent 1 or 2. If the probability is set to 0.5, each gene would have equal probability of being from parent 1 or parent 2 [1]. An example is shown in Fig. 6, where crossover is executed with a 0.5 probability.

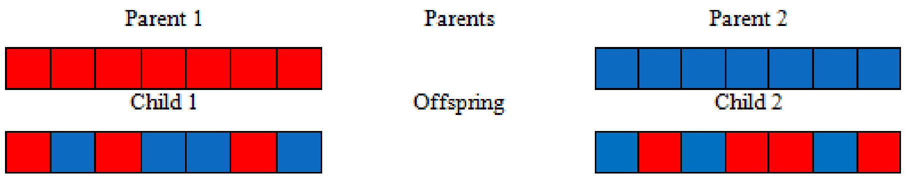


Fig. 6. Example of uniform crossover.

3.3 Mutation methods

For the mutation operator, we have implemented four different mutation methods, namely twors mutation, centre inverse mutation (CIM), reverse sequence mutation (RSM), and partial shuffle mutation (PSM).

Twors mutation Twors mutation is a mutation method that also can be referred to as swap. Two genes are randomly chosen, and their positions are swapped [1]. An example of twors mutation is shown in Fig.7, where gene number 3 and 5 are randomly chosen.

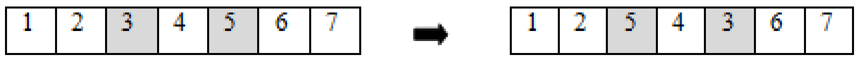


Fig. 7. Example of Twors mutation.

Centre inverse mutation (CIM) The CIM method chooses one random point, which divides a chromosome into two sections. The two sections are flipped

[1]. An example is shown in Fig. 8, where the random point is selected between gene 3 and 4.

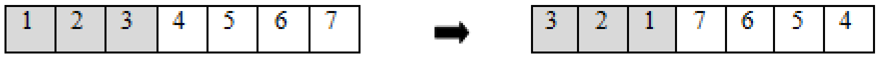


Fig. 8. Example of CIM method for mutation.

Reverse sequence mutation (RSM) The RSM method chooses two random points in the chromosome and selects the section of genes in between. The gene sequence inside the selected section is then reversed [1]. An example is shown in Fig. 9, where index 3 and 6 are randomly chosen.

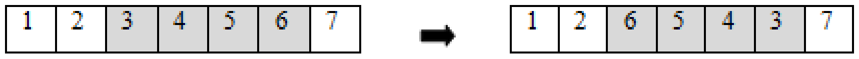


Fig. 9. Example of RSM method for mutation.

Partial shuffle mutation (PSM) The PSM method iterates through each gene in a chromosome. Each gene uses the mutation probability to determine if the gene should be swapped with another. If the gene is determined to be swapped, the gene will swapped with another randomly chosen gene [1]. An example is shown in Fig. 10, where genes number 2 and 6 were determined to be swapped with genes 4 and 5, respectively.

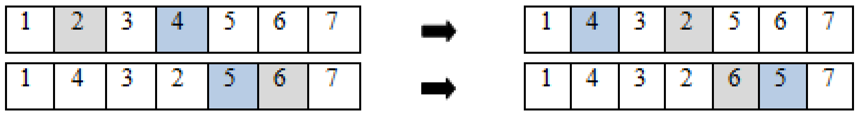


Fig. 10. Example of PSM method for mutation.

4 Results and analysis

The optimal tour distances for the two different datasets *Western Sahara* (29 cities) and *Djibouti* (38 cities), are 27603 and 6656, respectively. Both these datasets are freely available at the TSP website of the University of Waterloo, California, USA.[3]

For simplicity we have used the same GA settings for all methods, which we found empirically to work well: maximum number of generations: 1000, stall generation limit: 200, population size: 300, crossover probability: 0.7, mutation probability: 0.05, and elitism ratio: 0.20.

We ran the GA on an ordinary laptop with each combination of crossover and mutation methods presented previously, and each combination was run 10 times for each dataset to obtain the best tour distance (**b.distance**) and its percentage deviation from the optimal distance (**b.deviation**), the best running time in seconds (**b.time**), and the average percentage devation (**avg.deviation**) and average running time in seconds (**avg.time**).

Tables 1–2 summarise the results for the Western Sahara dataset, whereas Tables 3–4 summarises the results for the Djibouti dataset. Moreover, results using the conventional crossover methods for TSP, namely PMX and OX, are shown in Tables 1 and 3, whereas results using the alternative chromosome encoding suggested by Üçoluk [25] that enables single-point (1p), two-point (2p), and uniform crossover are shown in Tables 2 and 4.

4.1 Analysis

For the Western Sahara dataset with 29 cities and conventional crossover methods for TSP (Table 1), PMX crossover with all mutation methods but the Twors methods resulted in the optimal solution, whereas OX crossover resulted in the optimal solution for the CIM and PSM mutation methods. The average computational running time ranged from 24 to 32 seconds for these combinations of crossover and mutation methods. Using the alternative chromosome encoding (Table 2), single-point crossover with all mutation methods apart from Twors resulted in the optimal solution, as did uniform crossover with the PSM and RSM mutation methods. The average computational time ranged from 56 to 110 seconds for these combinations of crossover and mutation methods.

For the Djibouti dataset with 38 cities and conventional crossover methods for TSP (Table 3), PMX and OX crossover both resulted in the optimal solution when combined with the RMS mutation method. The average computational running time was 46 and 36 seconds, respectively, for these two combinations. Using the alternative chromosome encoding (Table 4), single-point, two-point, and uniform crossover all resulted in the optimal solution when combined with

Table 1. Western Sahara dataset with PMX and OX crossover.

Crossover	**Mutation**	**b.distance**	**b.deviation**	**b.time**	**avg.deviation**	**avg.time**
PMX	TWORS	28864.92	4.37	32	19.68	24
PMX	CIM	27603	0	36	2.80	30
PMX	PSM	27603	0	28	15.78	32
PMX	RSM	27603	0	34	1.80	28
OX	TWORS	27748.71	0.52	29	11.52	23
OX	CIM	27603	0	22	0.93	24
OX	PSM	27603	0	20	8.05	30
OX	RSM	27748.71	0.52	25	2.36	19

Table 2. Western Sahara dataset with single-point, two-point, and uniform crossover.

Crossover	Mutation	b.distance	b.deviation	b.time	avg.deviation	avg.time
1p	TWORS	28972.52	4.727	67	15.47	76
1p	CIM	27603	0	118	8.32	85
1p	PSM	27603	0	61	7.42	62
1p	RSM	27603	0	51	1.95	56
2p	TWORS	28373.51	2.7156	42	13.24	54
2p	CIM	28256.09	2.3113	63	10.59	79
2p	PSM	27748.71	0.525	66	16.8	69
2p	RSM	27748.71	0.525	43	2.67	53
Uniform	TWORS	28189.75	2.0814	67	8.84	63
Uniform	CIM	27748.71	0.5251	134	8.55	89
Uniform	PSM	27603	0	89	4.87	110
Uniform	RSM	27603	0	89	2.45	110

Table 3. Djibouti dataset with PMX and OX crossover.

Crossover	Mutation	b.distance	b.deviation	b.time	avg.deviation	avg.time
PMX	TWORS	8236.96	19.19	40	28.7	41
PMX	CIM	6758.28	1.51	49	5.31	60
PMX	PSM	7605.59	12.48	71	25.42	60
PMX	RSM	6656	0.0	48	6.92	46
OX	TWORS	7290.80	8.71	54	20.06	37
OX	CIM	6656	0	54	1.80	42
OX	PSM	7532.99	11.64	46	23.02	49
OX	RSM	6656	0	34	3.26	36

Table 4. Djibouti dataset with single-point, two-point, and uniform crossover.

Crossover	Mutation	b.distance	b.deviation	b.time	avg.deviation	avg.time
1p	TWORS	7415.21	10.24	109	25.97	112
1p	CIM	9011.86	26.14	163	34.06	132
1p	PSM	8134.72	18.18	166	24.31	144
1p	RSM	6656	0	156	5.96	132
2p	TWORS	7415.61	10.24	149	23.13	119
2p	CIM	7594.54	12.36	143	24.46	150
2p	PSM	7363.26	9.61	139	21.28	146
2p	RSM	6656	0	119	3.01	140
Uniform	TWORS	7688.32	13.43	154	21.87	119
Uniform	CIM	7421.81	10.32	165	21.42	141
Uniform	PSM	7313.06	8.98	166	21.70	161
Uniform	RSM	6656	0	161	3.89	149

the RSM mutation method. The average computational time was 132, 140, and 149 seconds, respectively, for these three combinations of crossover and mutation methods.

The only combinations of crossover and mutation methods that found the optimal solution for both datasets were (i) PMX, (ii) single-point, and (iii) uniform crossover with RSM mutation, and (iv) OX crossover with CIM mutation. These four combinations had average running times of (i) 28 and 46 seconds, (ii) 56 and 132 seconds, (iii) 110 and 149 seconds, and (iv) 24 and 42 seconds. Their average percentage deviation from the optimal distance for the 10 runs were (i) 1.80 and 6.92, (ii) 1.95 and 5.96, (iii) 2.45 and 3.89, and (iv) 0.93 and 1.80.

Hence, for the aforementioned GA settings used here, taking into account both the ability to find the optimal solution and to find near-optimal solutions, as well as running time, the OX crossover with CIM mutation appears to be the best choice. Using PMX with RSM mutation is slightly worse when comparing running times, and particularly also when comparing the average deviation for the largest dataset, Djibouti.

The two combinations using the alternative encoding scheme have much longer running times than the two best combinations (around 2–4 times higher), and also higher average deviations for the West Sahara dataset. For the Djibouti dataset, these two alternative encoding combinations yield better results than PMX with RSM for average percentage deviation, but worse than OX with CIM.

The best routes found by OX crossover and CIM mutation with corresponding cost function convergence plots for the two datasets are shown in Fig. 11.

5 Discussion and conclusions

It can be observed from the results above that when the dataset becomes more difficult (more cities) fewer combinations of crossover and mutation methods were able to find the optimal solution. When considering average percentage deviation as well as average running time, the GA with OX crossover and CIM mutation was the best choice. However, it is important to note that this claim is only valid for the two datasets and the choice of GA settings described previously. It may be that for other settings of population size, crossover and mutation probability, and elitism ratio, different results may be obtained. Nevertheless, we have examined by trial and error a number of other GA settings without observing any clear counter-indications that the results do not generalise.

Regarding the alternative encoding scheme proposed by Üçoluk [25], we were unable to reproduce the claim that this method should lead to a significant speed-up whilst only slightly worse performance (in terms of finding optimal or near-optimal solutions). On the contrary, we found the alternative encoding scheme to be typically 2–4 times slower than using conventional crossover methods. We have based our implementation (which was coded in Python 3) for finding the inverse sequences and converting back to permutations on the pseudocode provided by Üçoluk [25]. Still, there might be parts in the conversion process in

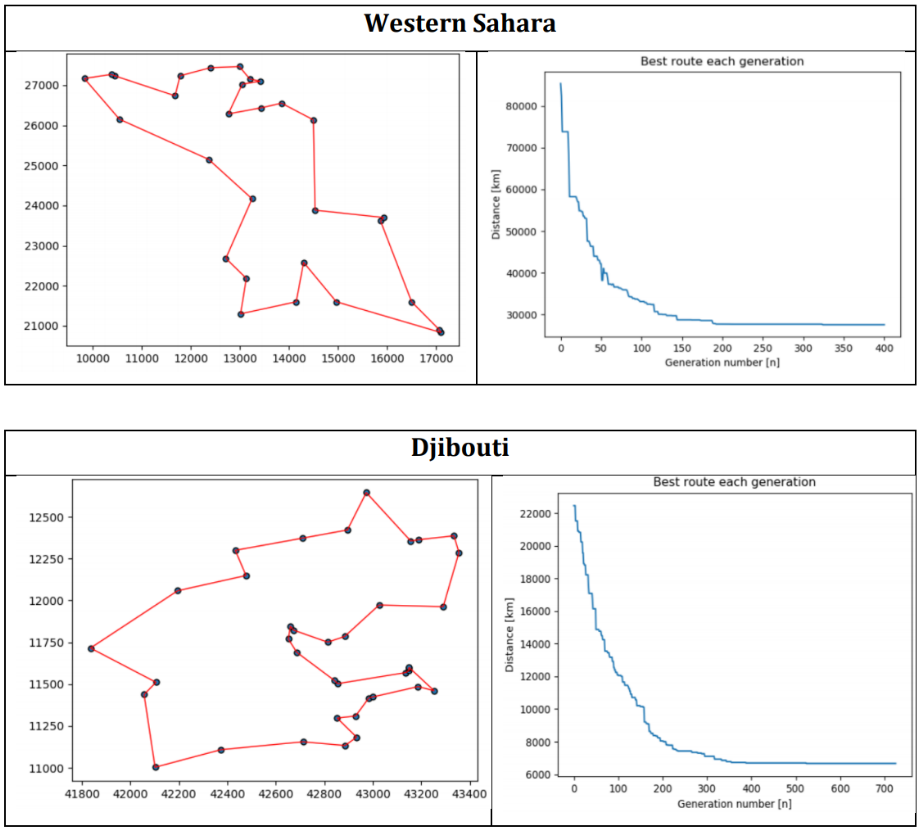


Fig. 11. Best routes and convergence plots using OX with CIM.

our implementation that causes this particular operation to be slower than in the case of Üçoluk.

5.1 Conclusions

The TSP is a popular NP-hard combinatorial optimization benchmark problem with many applications towards the real world. In this paper, we have examined using a GA implemented with various combinations of crossover and mutation methods for solving two datasets with 29 and 38 cities. We examined both conventional crossover methods for TSP (PMX and OX) and an alternative chromosome encoding scheme using inversion sequences than enabled ordinary crossover methods (single-point, two-point, and uniform crossover). All these crossover methods were tested with four different mutation methods (Twors, CIM, PSM, and RSM). For the two datasets and the population size, crossover probability, and mutation probability that we tested, OX crossover with CIM mutation was the best method.

Going forward, it would be interesting to investigate whether our code can be optimised (or not) regarding the alternative encoding, as well as examine larger datasets for further comparisons of GA crossover and mutation methods for TSPs.

6 Acknowledgement

This research was supported by the European Research Consortium for Informatics and Mathematics (ERCIM), which provided funding to Ramesh Chandra for his postdoctoral fellowship at the CPS Lab at NTNU, Ålesund, Norway.

References

1. Abdoun, O., Abouchabaka, J.: A comparative study of adaptive crossover operators for genetic algorithms to resolve the traveling salesman problem. arXiv preprint arXiv:1203.3097 (2012)
2. Abdoun, O., Abouchabaka, J., Tajani, C.: Analyzing the performance of mutation operators to solve the travelling salesman problem. arXiv preprint arXiv:1203.3099 (2012)
3. Bye, R.T.: A receding horizon genetic algorithm for dynamic resource allocation: A case study on optimal positioning of tugs. In: Computational Intelligence, pp. 131–147. Springer (2012)
4. Bye, R.T., Osen, O.L., Pedersen, B.S., Hameed, I.A., Schaathun, H.G.: A software framework for intelligent computer-automated product design. In: Proceedings of the 30th European Conference on Modelling and Simulation (ECMS '16). pp. 534–543 (Jun 2016)
5. Bye, R.T., Osen, O.L., Rekdalsbakken, W., Pedersen, B.S., Hameed, I.A.: An intelligent winch prototyping tool. In: Proceedings of the 31st European Conference on Modelling and Simulation (ECMS '17). pp. 276–284 (May 2017)
6. Bye, R.T., Schaathun, H.G.: Evaluation heuristics for tug fleet optimisation algorithms: A computational simulation study of a receding horizon genetic algorithm. In: Proceedings of the 4th International Conference on Operations Research and Enterprise Systems (ICORES '15). pp. 270–282 (2015), selected for extended publication in Springer book series Communications in Computer and Information Science (CCIS)
7. Bye, R.T., Schaathun, H.G.: An improved receding horizon genetic algorithm for the tug fleet optimisation problem. In: Proceedings 28th European Conference on Modelling and Simulation ECMS 2014, May 27th-May 30th, 2014, Brescia, Italy. ECMS European Council for Modelling and Simulation (2014)
8. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edn. (1989)
9. Goldberg, D.E., Lingle, R., et al.: Alleles, loci, and the traveling salesman problem. In: Proceedings of an international conference on genetic algorithms and their applications. vol. 154, pp. 154–159. Lawrence Erlbaum, Hillsdale, NJ (1985)
10. Goodman, E.D.: Introduction to genetic algorithms. In: Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation. pp. 205–226. GECCO Comp '14, ACM, New York, NY, USA (2014)

11. Grefenstette, J., Gopal, R., Rosmaita, B., Van Gucht, D.: Genetic algorithms for the traveling salesman problem. In: Proceedings of the first International Conference on Genetic Algorithms and their Applications. vol. 160, pp. 160–168 (1985)
12. Hacizade, U., Kaya, I.: Ga based traveling salesman problem solution and its application to transport routes optimization. IFAC-PapersOnLine **51**(30), 620–625 (2018)
13. Hameed, I.A., Bye, R.T., Osen, O.L., Pedersen, B.S., Schaathun, H.G.: Intelligent computer-automated crane design using an online crane prototyping tool. In: Proceedings of the 30th European Conference on Modelling and Simulation (ECMS '16). pp. 564–573 (Jun 2016), best Paper Nominee
14. Hameed, I.A., Bye, R.T., Pedersen, B.S., Osen, O.L.: Evolutionary winch design using an online winch prototyping tool. In: Proceedings of the 31st European Conference on Modelling and Simulation (ECMS '17). pp. 292–298 (May 2017)
15. Holland, J.H.: Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, Oxford, England (1975)
16. Hussain, A., Muhammad, Y.S., Nauman Sajid, M., Hussain, I., Mohamd Shoukry, A., Gani, S.: Genetic algorithm for traveling salesman problem with modified cycle crossover operator. Computational intelligence and neuroscience **2017** (2017)
17. Larranaga, P., Kuijpers, C.M.H., Murga, R.H., Inza, I., Dizdarevic, S.: Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artificial Intelligence Review **13**(2), 129–170 (1999)
18. Li, G., Zhang, H., Zhang, J., Bye, R.T.: Development of Adaptive Locomotion of a Caterpillar-like Robot Based on a Sensory Feedback CPG Model. Advanced Robotics **28**(6), 389–401 (2014)
19. Lin, B.L., Sun, X., Salous, S.: Solving travelling salesman problem with an improved hybrid genetic algorithm. Journal of computer and communications. **4**(15), 98–106 (2016)
20. Mirjalili, S.: Evolutionary multi-layer perceptron. In: Evolutionary Algorithms and Neural Networks, pp. 87–104. Springer (2019)
21. Razali, N.M., Geraghty, J., et al.: Genetic algorithm performance with different selection strategies in solving tsp. In: Proceedings of the world congress on engineering. vol. 2, pp. 1–6. International Association of Engineers Hong Kong (2011)
22. Whitley, L.D., Starkweather, T., Fuquay, D.: Scheduling problems and traveling salesmen: The genetic edge recombination operator. In: ICGA. vol. 89, pp. 133–40 (1989)
23. Xu, J., Pei, L., Zhu, R.z.: Application of a genetic algorithm with random crossover and dynamic mutation on the travelling salesman problem. Procedia computer science **131**, 937–945 (2018)
24. Yang, J., Shi, X., Marchese, M., Liang, Y.: An ant colony optimization method for generalized tsp problem. Progress in Natural Science **18**(11), 1417–1422 (2008)
25. Üçoluk, G.: Genetic algorithm solution of the TSP avoiding special crossover and mutation. Intelligent Automation & Soft Computing **8**(3), 265–272 (2002)