Journal of Engineering Research and Reports

20(10): 97-102, 2021; Article no.JERR.69217
ISSN: 2582-2926

Studying the effect of Eliminating Repeated Individuals from the Population in a Genetic Algorithm: Solution Perspectives for the Travelling Salesman Problem

Laura Michele Báez Villegas[1*] and Santiago Omar Caballero Morales[1]

[1]Department of Logistics and Supply Chain Management, Universidad Popular Autonoma del Estado de Puebla (UPAEP), Mexico.

Authors' contributions

This work was carried out in collaboration between both authors. Author LMBV performed the design of the GA, statistical analysis of results, computer coding, selection of test instances, and wrote the first draft of the manuscript. Author SOCM designed the test methodology, the base GA, revised the computer coding of the GA and the final draft of the manuscript. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/JERR/2021/v20i1017393
Editor(s):
(1) Dr. Tian- Quan Yun, South China University of Technology, China.
Reviewers:
(1) Tobias Rupp, University of Stuttgart, Germany.
(2) Min Xu, Carnegie Mellon University, USA.
Complete Peer review History: https://www.sdiarticle4.com/review-history/69217

Original Research Article

Received 10 May 2021
Accepted 16 July 2021
Published 23 July 2021

ABSTRACT

The Travelling Salesman Problem (TSP) is one of the main routing problems in the Logistics and Supply Chain Management fields. Given its computational complexity, metaheuristics are frequently needed to solve it to near-optimality. In this aspect, Genetic Algorithms (GA) are promising methods, however, their search performance depends of populations of solutions which can increase computational processing. Thus, the management of this component is subject to adaptations to reduce its computational burden and improve overall performance. This work explores on the elimination of repeated individuals within the population which may represent a significant fraction of its size and do not add valuable information to the solution search mechanisms of the GA. This cleaning process is expected to contribute to solution diversity. Experiments performed with different TSP test instances support the finding that this cleaning process can improve the convergence of the GA to very suitable solutions (within the 10% error limit). These findings were statistically validated.

**Corresponding author: Email: laura.baezvs@gmail.com;*

Keywords: TSP; genetic algorithms; mixed populations.

1. INTRODUCTION

The Travelling Salesman Problem (TSP) is one of the main routing problems in the Logistics and Supply Chain Management (SCM) fields [1-6]. These problems are usually focused on analyzing the scenario of a "travelling agent" which departs from an "origin location", visits all "service locations" within a region, and at the end of the trip returns to the "origin location". Note that all "service locations" are visited only once. This kind of trip, which is known as a "Hamiltonian Circuit", is representative of many distribution problems [7]. The objective of the TSP is to find out the Hamiltonian Circuit with the lowest distance and cost. Fig. 1 shows an example of a feasible solution for a TSP.

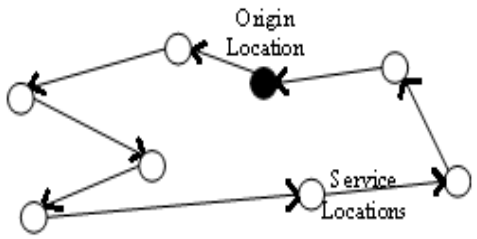


Fig. 1. Example of a feasible TSP solution

As presented, besides distribution, the TSP can be used to model transfers for warehouses, within production plants, and sequencing of jobs in manufacturing [3].

2. USING GENETIC ALGORITHMS FOR THE TSP

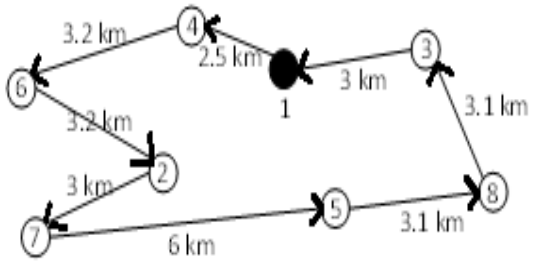
Genetic Algorithms (GAs) are search techniques which are based on the concepts of natural selection or "survival of the fittest". These type of algorithms are important because they can find very suitable solutions (in some cases, optimal solutions) within reasonable time for complex combinatorial problems. Table 1 presents the general steps of a GA. For the TSP, an individual (feasible solution) and its fitness may look as shown in Fig. 2. Due to this representation, or codification, the generation of new solutions is performed through permutation crossover and mutation operators [8,9]. For GA for the TSP the Partially Matched Crossover (PMX) and exchange mutation are frequently considered [8-10].

3. CLEANING PROCESS FOR THE GA POPULATION

As the GA iterates through many generations, copies of the fittest individuals increase within the population and dominate the individuals surviving from one generation to the next (convergence to a local optimum). While the crossover and mutation operators are expected to reduce this situation, they are not enough and full convergence of the population to the same individuals is frequently observed.

Table 1. General steps of a GA

Steps	Description
1	Build an initial set of *N* solutions which will be improved through the execution of the GA.
2	Evaluate the *N* individuals of the population to determine their "fitness". Order population according to their fitness to solve the considered problem (minimization, maximization).
3	Perform an iterative search process to improve the initial population. Each iteration, or "generation", consists of the following tasks: • Select the fittest "parents" for reproduction (pairs of individuals within the current population) and offspring generation (new solutions or individuals). • Reproduction of parents through crossover and mutation operators. This step generates an offspring population with X new solutions. • Evaluate the fitness of the X offspring solutions. • Sort the whole *N*+*X* population according to the problem's objective. • Select the best *N* individuals and update the current population with these individuals. • If stop condition is not met, continue with the next iteration (generation) until condition is met.
4	Return the best solution within the final population.



Individual = [1 4 6 2 7 5 8 3 1] fitness = [2.5 + 3.2 + 3.2 + 3 + 6 + 3.1 + 3.1 + 3]
= [27.1]

Fig. 2. Codification of a feasible TSP solution for a GA

If there is a pair of repeated individuals, PMX will not produce different individuals. On the other hand, the mutation operator can modify any single solution or individual to make it different. In the step of selecting the best N individuals, the offspring produced by PMX (copies of the repeated individuals) will be favored over those produced by mutation. With enough iterations of the GA, all individuals in the population will be copies of the original pair.

If there are repeated individuals, only the mutation operations increase genetic diversity, but next those individuals are eliminated. The elimination of repeated individuals gives an opportunity to mutated individuals to survive to the next GA iteration, preserving the above genetic diversity.

From the computational point of view, the application of the reproduction operators to very similar individuals do not create significantly better individuals (offspring). In contrast, it wastes computational resources and time. The idea behind the elimination of repeated individuals is to preserve genetic diversity, and thus, force the algorithm to get out of the local search space.

For this purpose, the general GA (see Table 1) was adapted at the reproduction steps. Table 2 presents the adapted GA with elimination of repeated individuals.

Implementation of the GA was performed on GNU Octave [1] and it was executed on an Intel i5 at 1.0 GHz laptop computer with 8GB RAM. Experiments were performed with 20 test instances of the TSPLIB95 database [2]. The size of these instances varied from 52 to 500 locations or nodes. The GA was configured with the following parameters: tournament method for selection of parents, PMX and exchange mutation, N = 100 individuals and 1000 generations.

For comparison purposes, the performance of the GA with elimination of repeated individuals (With Elimination) was compared with the GA without the elimination procedure (Without Elimination).

4. RESULTS AND DISCUSSION

Table 3 presents the details of the test instances considered for the present work. Also, the final results (% error from best-known solution) through 1000 generations are presented.

As presented, in some instances (4/20 = 20%), the elimination process worsens the quality of the final solution. However, in most of the cases (16/20 = 80%) the process improves the final solution. Fig. 3 presents an overview of the mean error convergence of the GA through all generations.

It is important to observe that the effect of elimination is smaller for large test instances (i.e., with # nodes > 200 location nodes). However, this may be related to the size of the population which was set to N=100 for all instances. Thus, for # nodes < N, a higher improvement is observed. To address this situation, the last eight test instances were tested with N = 500. The results are presented in Table 4. It can be observed that the increase in N leads to better average results from 11.9% and 12.1% to 7.4% for the without and with elimination scenarios respectively. Note that the number of instances where the elimination process worsens the quality of the final solution decreases by one.

Table 2. Adapted steps of a GA for elimination of repeated individuals

Steps	Description
1	Build an initial set of *N* solutions which will be improved through the execution of the GA.
2	Evaluate the *N* individuals of the population to determine their "fitness". Order population according to their fitness to solve the considered problem (minimization, maximization).
3	Perform an iterative search process to improve the initial population. Each iteration, or "generation", consists of the following tasks: • Select the fittest "parents" for reproduction (pairs of individuals within the current population) and offspring generation (new solutions or individuals). • Reproduction of parents through crossover and mutation operators. This step generates an offspring population with X new solutions. • Evaluate the fitness of the X offspring solutions. • Sort the whole N+X population according to the problem's objective. • Every 50 generations the cleaning process of the whole *N+X* population is performed as follows: • A) The fitness of consecutive individuals is compared until an equal fitness is found. Next step compares the first individual of this pair with following consecutive individuals element to element. • B) The individuals are compared element to element to verify if they are equal. If they are equal, the fitness is penalized to a large value if minimization is required. This process is repeated until an individual with a different fitness is found. Then the scan process is restarted from this last element until the whole population is scanned. • C) Then, the whole population is sorted again. The repeated individuals are sent to the region of the population with the lowest fitness. Some individuals, which would have been deleted, survive to the next generation. • Select the best *N* individuals and update the current population with these individuals. • If stop condition is not met, continue with the next iteration (generation) until condition is met.
4	Return the best solution within the final population.

Table 3. Final results with the GA without/with elimination of repeated individuals (*N* = 100)

Nodes	Instance	Without Elimination	With Elimination
52	berlins2.tsp	6.790	6.555
70	st/O.isp	3.8%	2.35
100	kroC100.tsp	9.1%	3.895
101	eaill0Ltsp	7.6%	6.3%
105	linlOS.tsp	7.1%	3.6%
107	prid?.tsp	0.256	0.1%
124	pri24.tsp	5.79	4.5%
144	prl44.tsp	4.996	4.7%
150	chi50.tsp	7.7%	5.695
152	pris2.tsp	6.6%	5.155
159	ui5S.tsp	13.0%	9.9%
198	d198.tsp	8.5%	5.7%
225	ts225.tsp	2.6%	3.1%
2b4	pr2b4.tsp	12.4%	13.9%
280	a280.tsp	14.9%	14.1%
299	pr299.tsp	11.4%	9.7%
318	lin318.p	11.6%	12.7%
400	rd400.tsp	12.7%	14.0%
443	peb442.tsp	15.795	15.0%
493	d493.tsp	13.6%	13.5%

Table 4. Final results with the GA without/with elimination of repeated individuals (N = 500 for instances with # nodes > 200 location nodes)

			N =100	N =500	
Nodes	**Instance**	**Without Elimination**	**With Elimination**	**Elimination**	**With Elimination**
225	ts225.tsp	2.6%	3.1%	1.4%	3.2%
264	pr264.tsp	12.4%	13.9%	8.9%	6.4%
280	a280.tsp	14.9%	14.1%	6.3%	7.0%
299	pr299.tsp	11.4%	9.7%	6.8%	6.7%
318	lin318.tsp	11.6%	12.7%	7.4%	9.6%
400	rd400.tsp	12.7%	14.0%	10.6%	10.2%
442	pcb442 tsp	15.7%	15.6%	9.7%	8.7%
493	d493.tsp	13.6%	13.5%	7.8%	7.6%
	Average =	11.9%	12.1%	7.4%	7.4%

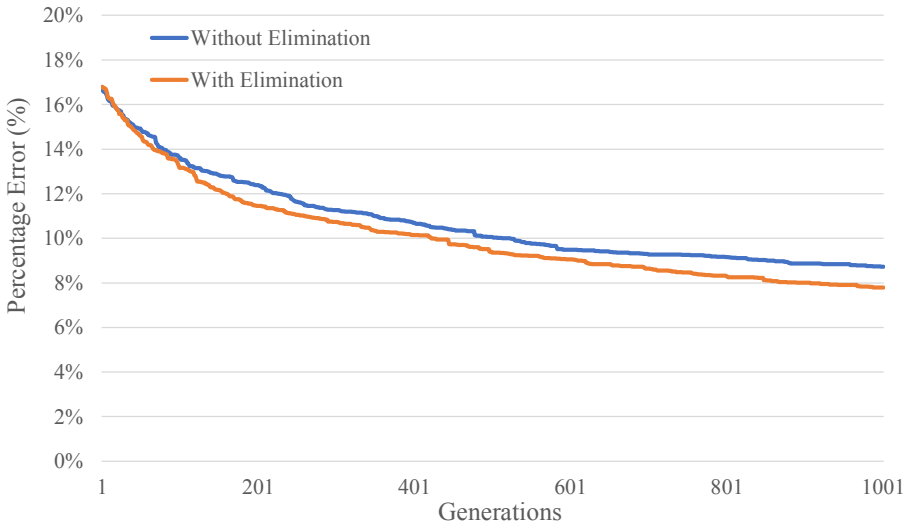


Fig. 3. Average error through all generations of the GA (*N* = 100)

On the other hand, the elimination of repeated individuals improves the convergence and the overall results of the GA. This performance was statistically validated through a 2-sample-t test. With a p-value < 0.001 it was determined that the improvement was statistically significant.

5. CONCLUSIONS

In the present work, a GA was developed to solve TSP instances with more than 200 location nodes. In particular, the GA integrated the concept of cleaning the population by eliminating repeated individuals. This is expected to provide important insights for future improvement of this method, not for current comparison purposes.

By considering scenarios with population sizes of 100 and 500 individuals we observed consistency regarding the positive effect of the cleaning process on the overall performance of the GA for TSP instances.

This observation is important to delimit the adjustment of parameters of the GA to solve the TSP. While we explored on the solution diversity of the population, we also observed a high computational load related to its size. Thus, while the cleaning process is promising to improve GA performance for the TSP, it still involves significant computational resources.

Although the size of the population influences the speed and quality of the solutions, a more strategic approach could be the development of better reproduction operators to produce more suitable offspring with small populations and fewer generations. Hence, future work is aimed to improve the crossover operators. The multi-parent method described in [10] can be a very suitable starting point for this purpose.

ACKNOWLEDGEMENTS

Laura Michele Baéz Villegas wants to thank the Consejo Nacional de Ciencia y Tecnologia (Conacyt), Mexico for its financial support in the form of a scholarship to pursue PhD studies in Logistics and Supply Chain Management at UPAEP.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Eaton JW. GNU Octave Version 6.2.0. Available:https://www.gnu.org/software/octave/index (last accessed on 05/05/2021).
2. Reinelt G. TSPLIB - Symmetric traveling salesman problem (TSP). Available:http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ (last accessed on 05/05/2021).
3. Del Moral P, Kallel L, Rowe J. Modeling Genetic Algorithms With Interactive Particle Systems. Revista de Matemática: Teoría y Aplicacione. 2001;8(2):19-77.
4. Cant-Paz E. A Survey of parallel Genetic Algorithms. Calculateurs Paralleles, Reseaux et Systems Repartis. 1998;10(2): 141-177.
5. Ishibuchi H, Yamamoto T. Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. Fuzzy Sets and Systems. 2004;141(1): 59-88.
6. Semekin E, Semenkina M. Self-configuring genetic algorithm with modified uniform crossover operator. International Conference in Swarm Intelligence. 2012;4 14-421.
7. González-Velarde JL, Ríos-Mercado RZ. Investigación de Operaciones en acción: Aplicación del TSP en problemas de manufactura y logística. Ingenierías. 1999;II(4):18-23.
8. Kumar V, Panneerselvam R. A Study of Crossover Operators for Genetic Algorithms to Solve VRP and its Variants and New Sinusoidal Motion Crossover Operator. International Journal of Computational Intelligence Research. 2017;13(7):1717-1733.
9. Kora P, Yadlapalli P. Crossover Operators in Genetic Algorithms: A Review. International Journal of Computer Applications. 2017;162(10):34-36.
10. Ting CK, Su CH, Lee CN. Multi-Parent extension of partially mapped crossover for combinatorial optimization problems. Expert Systems with Applications. 2010;37:1879– 1886.

Peer-review history:
The peer review history for this paper can be accessed here:
https://www.sdiarticle4.com/review-history/69217