Proceedings of the Second Asia Pacific International Conference on Industrial Engineering and Operations Management, Surakarta, Indonesia, September 14-16, 2021

Comparison of crossover operators in genetic algorithm for vehicle routing problems

Eric Wibisono, Iris Martin, Dina Natalia Prayogo
Department of Industrial Engineering, Faculty of Engineering
University of Surabaya
Surabaya, Indonesia
ewibisono@staff.ubaya.ac.id

Abstract

Genetic algorithm (GA) is a popular metaheuristic with wide-ranging applications, e.g. in routing problems such as traveling salesman problem (TSP) or vehicle routing problem (VRP). Seeking the best combination of parameters in GA application is the key objective in the line of research involving GA. One possible factor to be tested is the operator used for crossover. For VRP, a number of research reporting good performance use the order crossover (OX) operator. For TSP, one paper proposed the modified cycle crossover (CX2) operator and reported that it is better than OX and the partially mapped crossover (PMX). The interest and objective of this paper is to test these three operators in a VRP setting. Excluding the crossover operator, other good principles of GA for VRP obtained from the literature are maintained. The experiment results suggest these findings. Firstly, CX2 is expensive in run time and has difficulty escaping from local optimum but leads to the best fitness value compared to the other operators. Secondly, PMX ranks second both in the fitness performance and run time. Thirdly, while OX has slightly inferior performance, it is able to explore wider search space and therefore still has lots of potential for future research.

Keywords
Genetic algorithm, Vehicle routing problem, Crossover operator.

1. Introduction

Genetic algorithm (GA) is a branch of metaheuristics under the population-based category. It works by emulating the process of natural selection as proposed in Darwin's theory of evolution. In this process, members of population (chromosomes) will compete for opportunity to mate, produce offspring**,** and pass their traits to the next generation. Because the winners of such competition are members with better attributes, the offspring tend to be better than the majority of the existing population members. Over time, the population as a whole will also get better in the respective attributes. GA as a field of study was initially theorized by Holland (1975), but it gained popularity after promoted by Goldberg (1989), one of Holland's students. It has now been used in wide range of applications including production scheduling (Grosch et al., 2021), engineering design (Sergeeva et al., 2017), finance and management (Vizcaíno-González, et. al., 2016)—to name a few. In logistics, it can be applied in routing problems (Saif-Eddine et al., 2019) and study using multi-objective optimization is also commonly found (Borhani, 2021).

As a population-based algorithm, GA consists of several building blocks that form the backbone of its processes. These are population management, crossover, and mutation. Population management deals with mechanism to maintain the population members by deciding which member to be added to and which to be removed from the population in each iteration; crossover operator deals with how members will "mate" and produce offspring; and mutation is a procedure triggered once in a while to break away from the current search process thus enabling it to explore new areas. Each of these building blocks, together with fine tuning of GA parameters (such as population size, mutation rate, etc.), will determine the performance of the algorithm.

In its application for logistics, population members can be coded in many ways, but generally the chromosome structure will consist of nodes of the problem and their sequence. In routing problems such as the traveling salesman problem (TSP) or vehicle routing problem (VRP), the sequence of nodes indicates the order of visits to those nodes. Crossover and mutation operators have the role to alter this sequence following a certain rule. Despite their importance, studies on the impact of crossover operators on GA performance are relatively rare. Examples of these studies can be found in Nazif and Lee (2012) and a good summary can be found in Awad et al. (2018).

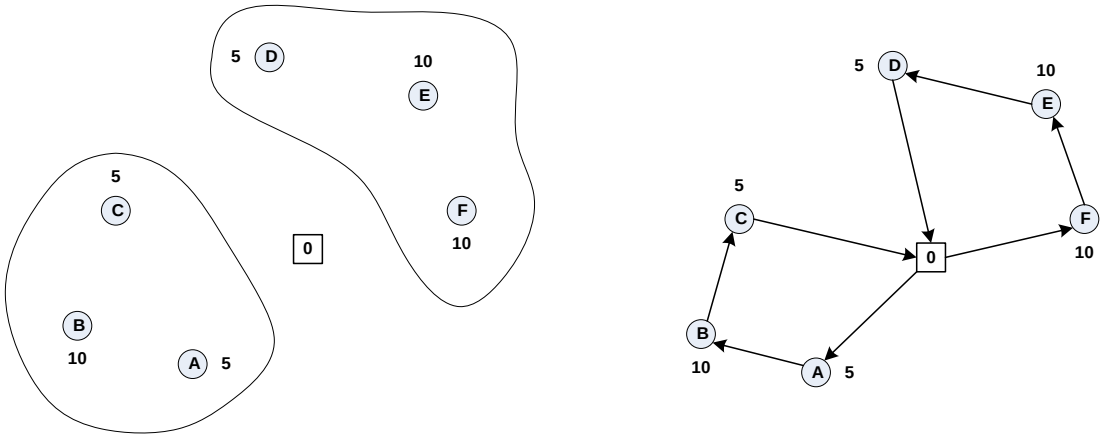
An algorithm that stands out among many GA applications in VRP is the one proposed in Prins (2004) for capacitated VRP, followed by its extension for heterogeneous VRP (HVRP) in Prins (2009). In both papers, the author used only one crossover operator called the order crossover (OX). Except for its comparison to the linear order crossover (LOX), not much discussion was elaborated on the selection of OX compared to the other parameters that were tested and analyzed with regard to their impact to the GA performance. In another study, Hussain et al. (2017) compared OX, the partially mapped crossover (PMX), and the modified cycle crossover (CX2) for TSP. The authors concluded that the performance of operators is instance-dependent but in general CX2 performs better than the other two operators in a larger number of instances.

Given the above background and the fact that studies on the effectiveness of crossover operators in GA for routing problems are still rare, the interest of this paper is to test two crossover operators, PMX and CX2, that were tested in TSP and reported successful, particularly for CX2, but have never been tested and compared to the OX operator in VRP. These three operators, OX, PMX, and CX2, and their impact to the GA performance for VRP, will be the interest of this paper. Findings from this research will shed more light on factors that can help increase the effectiveness of GA, particularly in routing problems.

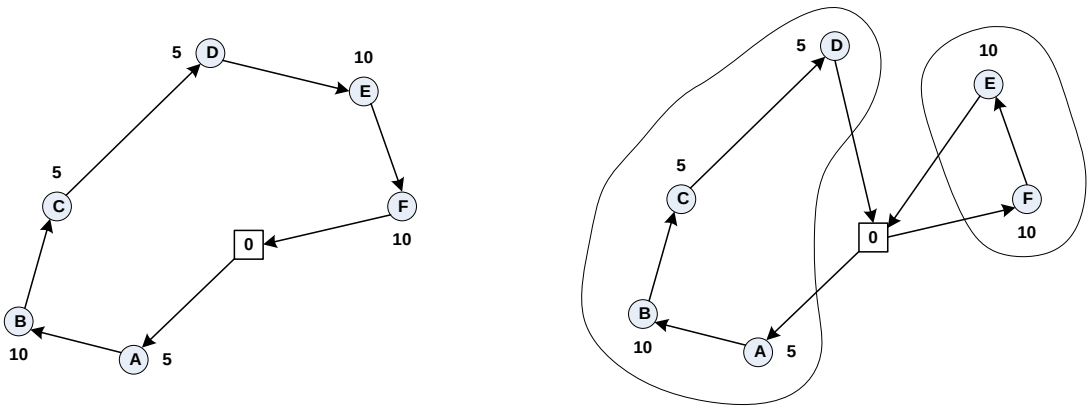
The objectives of this paper therefore are twofold. Firstly, three crossover operators, namely OX, PMX, and CX2, will be coded and embedded in a GA for VRP following the principles in Prins (2004) that have been proven effective. Secondly, comparison on the performance of each crossover operator will be analyzed based on fitness value and computation time. The code will be written in Python and C101 from Solomon data set will be used as a benchmark instance.

2. Literature Review

There are two approaches usually used to solve vehicle routing problems: cluster-first route-second and route-first cluster-second. In the first approach, nodes to be visited are clustered first according to certain method and routing is later applied within each cluster. In the latter approach, a giant tour is formed first as the main route, then the clustering follows whatever existing constraints, e.g. vehicle capacity. Illustration of these two approaches is shown in Figure 1. In this example, the depot is represented with a square denoted with 0 and six nodes denoted A, B, C, D, E, and F are to be served. The numbers next to the nodes are the demand of each node. Assume that vehicles can be procured as needed or in other words its availability is not constrained but each has a capacity of 25.



a) Cluster first… …route second



b) Route first… …cluster second

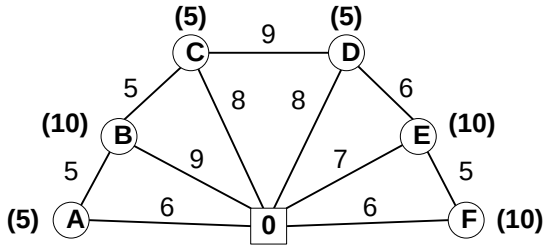
Figure 1. Two approaches to solve routing problems

The cluster-first route-second approach forms two clusters consisting of three nodes each. The formation of clusters follows simple proximity principle: nodes A, B, and C are closed to each other and likewise are nodes, D, E, and F, but A, B, C and D, E, F are distanced apart. Since both clusters do not violate capacity constraint, the solution is feasible. In contrast, the route-first cluster-second approach creates a giant tour visiting all nodes and clustering is constructed following this giant tour using any heuristic as long as the route does not violate capacity constraint. This approach also produces two clusters, but nodes A, B, C, and D are grouped together because the total demand of these four nodes can be served by one vehicle, followed by the second group with nodes E and F. It is important to note that the two approaches described here are construction heuristics and they can still be improved in the later phase. It is possible that although the starting construction routes are different, the final routes could be identical after the improvement phase (for example using 2-Opt), but with different computation times.

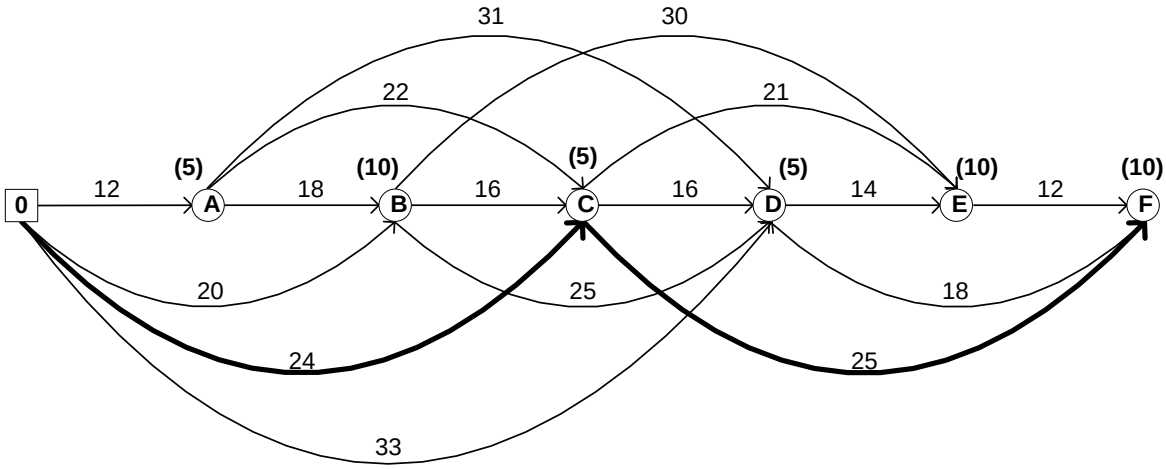
Both approaches have their backup in the literature. Laporte and Semet (2002) stated "*We are not aware of any computational experience showing that route-first, cluster-second heuristics are competitive with other approaches*." However, Prins et al. (2014) argued that since 2002, the route-first cluster-second approach has shown successful applications not just in VRP but also in the arc routing problem (ARP). Part of the effectiveness in Prins' GA is the inclusion of an algorithm that is able to split a given chromosome (a giant tour) into optimal tours. To highlight this important feature, Prins et al. (2014) prefer to use the term order-first split-second to route-first cluster-second for the route construction process, although essentially both are similar. We will describe how the Split algorithm works in the following section.

Split algorithm works by transforming the bi-directed graph problem into a directed acyclic graph corresponding to a minimum-cost path problem. Consider again the routes given in Figure 1b. If the order of nodes A-B-C-D-E-F is coded as a GA chromosome, the bi-directed graph of that order is given in Figure 2a (with added information on distances from the depot to the nodes and between nodes in the chain) and its transformation to the minimum-cost path problem is shown in Figure 2b. The task now becomes finding the shortest route from depot 0 to node F that yields minimum distance. Distances shown for each arc is read as follow. For example, a distance of 12 from 0 to A means we use one vehicle to serve node A and since the distance from 0 to A is 6, a return trip will give a total distance of 12. Furthermore, the arc connecting 0 to B means we serve nodes A and B in one trip resulting in a total distance of 20 (6 from 0-A + 5 from A to B + 9 from B back to 0). Likewise, the arc connecting 0 to C means we serve nodes A, B, and C in one trip. Note that an arc connecting A to E is infeasible since this means nodes B, C, D, and E have to be served by one vehicle and that violates the capacity constraint of 25.

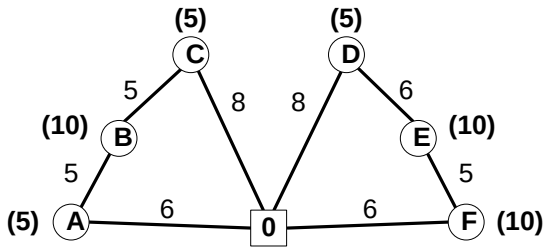
Using a dynamic programming approach, solution to the minimum-cost path problem is shown as the bold lines in Figure 2b, and the corresponding actual solution (partition of trips) is shown in Figure 2c.



a) Bi-directed graph of the problem



b) Transformation to a directed acyclic minimum-cost path problem



c) Solution of the problem showing partition of trips

Figure 2. Illustration of the Split algorithm

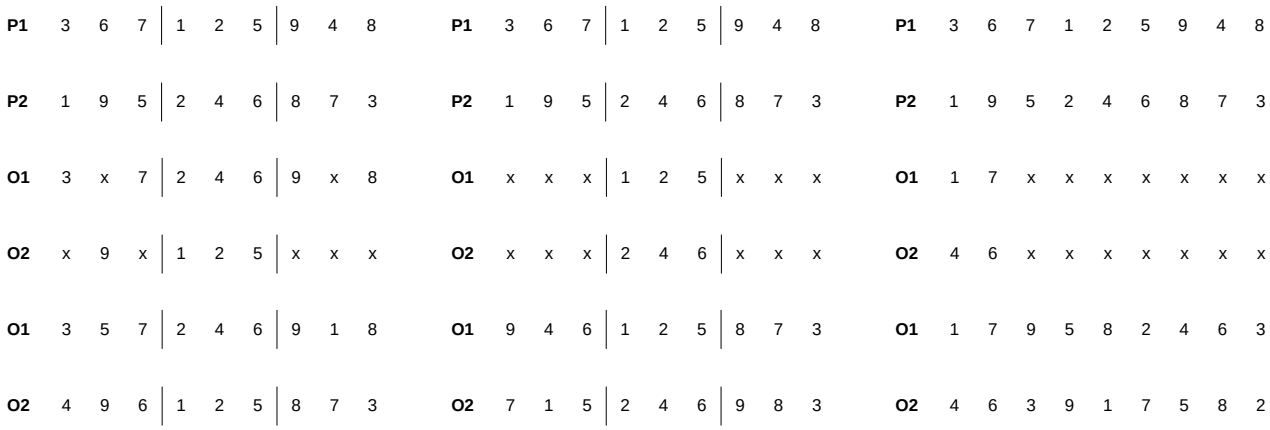
On capacitated VRP (CVRP), Split algorithm works very efficient with complexity $O(n^2)$, with n being the number of nodes, and is able to find optimal solution for any given giant tour (Prins, 2004). On heterogeneous VRP (HVRP), however, the complexity raises to $O(mtn^t)$ with m being the number of arcs in the acyclic graph and t being the number of vehicle types (Prins, 2009). In other study with a multi-objective setting, running times can be as high as five to six hours (Wibisono and Jittamai, 2017). For CVRP, Split can be implemented without worsening the efficiency of the GA. In other words, possible factors for inefficiency remain in the original building blocks or parameters of the GA itself, one of them is the selection of crossover operator.

A number of crossover operators are found in the literature. Awad et al. (2018) listed ten operators, namely: one-point crossover (1PX), two-point crossover (2PX), order crossover (OX), partially mapped crossover (PMX), cyclical crossover (CX), route-based crossover (RBX), sequence-based crossover (SBX), single parent crossover (SPO), genetic vehicle representation crossover (GVR), and best cost-best route crossover (BCBRC). Of all these operators, OX is the most widely used, including in Prins (2004). However, not much discussion was given nor testing was done to investigate the impact of OX on the GA performance. Excluding the crossover operator, other factors tested in that study include parameters such as population size and mutation rate, inclusion of a few good chromosomes obtained

using heuristics during the population initialization stage, population management by devising a spacing criterion to avoid having chromosomes with similar fitness value, parent selection method, etc.

GA can also be applied in other types of routing problems such as TSP. Hussain et al. (2017) proposed the modified cycle crossover (CX2) operator and compared it to OX, PMX, and the original CX in TSP. The authors concluded that, while each operator excels in some instances, CX2 has a better performance in a larger number of instances. The good performance of CX2 on TSP reported in this study sparks interest and motivation to test it in VRP. The main features of GA for VRP as outlined in Prins (2004) will be used as the key algorithm and three crossover operators, OX, PMX, and CX2 will be compared.

Before concluding this section, the mechanism of each crossover operator is explained next. Figure 3 displays the schematic on how the three crossover operators work. Two parent chromosomes, i.e. 3-6-7-1-2-5-9-4-8 and 1-9-5-2-4-6-8-7-3, will be processed with two cut points between nodes 3 and 4 and between nodes 6 and 7 for PMX and OX (CX2 does not require cut points). In PMX, nodes inside cut points are copied to the offspring with opposite numbers (P1 to O2 and P2 to O1), then we setup a mapping table between the nodes inside the cut points, i.e. 1↔2, 2↔4, and 5↔6. The nodes outside cut points that do not conflict with the copied nodes inside cut points are copied next, so we have 3-x-7 for the first part and 9-x-8 for the second part of O1, and x-9-x for the first part and x-x-x for the second part of O2. The remaining "x" are filled based on the mapping table, for example the "x" in 3-x-7 in O1 corresponds to 6 and from the mapping table we found that 6 is mapped to 5 (and vice versa). For the "x" in 9-x-8 in O1, we cannot fill it with 2 because 2 is already copied, so we have to continue the mapping process from 4↔2 to 2↔1 to get 9-1-8 for that portion.



a) Partially Mapped Crossover (PMX) b) Order Crossover (OX) c) Modified Cycle Crossover (CX2)

Figure 3. Mechanism of three crossover operators: PMX, OX, and CX2

For OX, we start by copying the nodes inside cut points from the parents to the corresponding section in the offspring (P1 to O1 and P2 to O2). Next, to fill the remaining nodes in O1, we look up in P2 and starting from the first node after the second cut point, we copy to O1 the remaining nodes in the same order, i.e. 8-7-3-9-4-6, also starting from the position right after the second cut point in O1. In the same manner, we redo the process to get O2.

Finally, CX2 process is explained as follow. First, we take the first node in P2 and copy it as the first node in O1 (node 1 in this example). We then look up node 1 in P1 and find the node in P2 in the same position (the fourth bit), which gives us node 2, and we redo this process one more time from 2 in P1 to 4 in P2 (the fifth bit). We copy 4 to the first position in O2. Next, using node 4 as a starting node, we redo the process in one move and the eighth bit shows the mapping of 4 to 7 so we copy 7 to the second position in O1. Redoing the process again in two moves (7→5 and 5→6), we get 6 to be filled in the second position in O2. We carry on in this manner (one move to fill O1 and two moves to fill O2) until finish or until the loop is closed before all nodes are copied. In case of the latter, the solution is as follow. Suppose we continue from the previous process, we get 6→9 and 9 is put after 7 in O1; then two moves later we obtain 3 for the successor in O2. However, we cannot add 1 (3→1) in the next cycle, since 1 is already in O1.

In this case, we carry on by taking the leftmost node in P2 that has not been assigned, i.e. 5, and put it as our next bit in O1, before we continue the process as previously explained.

3. Methods

The GA for VRP in this paper uses the same basic principles from Prins (2004). Some of these principles, as indicated by the author, will produce good performance if they are maintained. These are:

- Not allowing chromosomes with nearly identical fitness value
- Inclusion of good chromosomes in the initial population obtained from some heuristics
- Use of binary tournament instead of roulette wheel to choose parent chromosomes
- Aggressive mutation rate based on local search

Good population management involves a number of factors. First, chromosomes with nearly identical fitness value are not allowed to co-exist in the population. This is called "spaced criterion" and the definition of "nearly identical" is managed by a threshold value Δ. In any stage of chromosome production, the newly created chromosome will be compared to the other population members and if its fitness value F is not "spaced" against the fitness value of the other members, or $|F - F_t| \geq \Delta, t = 1..n$ with n being the population size, then the new chromosome is rejected.

The second principle involves the use of certain heuristics to create "good" chromosomes in the initial population so that the search process does not start from completely random domain. The number of good chromosomes suggested is between two and five; too few will have little to no effect, but too many will risk of reaching premature convergence. In Prins (2004), three heuristics were used, namely Clarke-Wright's savings, Mole-Jameson, and Gillett-Miller. In this paper, we will use Clarke-Wright's savings and nearest neighbor heuristics.

The third principle is on the mutation rate. In a general GA, mutation probability is advised to be set low, e.g. 1%, to follow the idea of actual genetic mutation that indeed rarely occurs. However, in GA for VRP, mutation is performed by a local search and to leave it at a low rate means we rely too much on the reproduction process (crossover). On the other hand, just like any heuristic that can still be improved by a local search after the construction stage, local search in GA for VRP could also help exploring the neighborhood of the chromosome to find a better solution. For this reason, mutation is set at an aggressive rate of 20%. When it is triggered, a local search in 9 moves (M1-M9) exploring all possible neighborhood is employed. These 9 moves will loop until no improvement is made in M9, i.e. starting from M1, if an improvement is achieved at any stage of node exchanges, M1 will restart. If after all possible node exchanges in M1 no improvement is made, the search continues with M2. If during exchanges in M2 an improvement is reached, the process will restart from M1. The 9-move local search is detailed below (Prins, 2004), with $T(u)$ being the trip visiting u.

- M1. If u is a client node, remove u then insert it after v,
- M2. If u and x are clients, remove them then insert (u; x) after v,
- M3. If u and x are clients, remove them then insert (x; u) after v,
- M4. If u and v are clients, swap u and v,
- M5. If u; x and v are clients, swap (u; x) and v,
- M6. If (u; x) and (v; y) are clients, swap (u; x) and (v; y),
- M7. If $T(u) = T(v)$, replace (u; x) and (v; y) by (u; v) and (x; y),
- M8. If $T(u) \neq T(v)$, replace (u; x) and (v; y) by (u; v) and (x; y),
- M9. If $T(u) \neq T(v)$, replace (u; x) and (v; y) by (u; y) and (x; v).

General management of the population is described as follow. First, population size is set at 50. Second, selection of parents is done by binary tournament instead of roulette wheel. Third, if at any iteration, a chromosome with fitness value better than that of the best chromosome in the population is found, one chromosome will be randomly selected from the bottom half of population with regard to the fitness value. We strictly follow these principles because a slight deviation (e.g. enlarging the population size, or selecting the worst chromosome to be replaced) will deteriorate the performance of the GA. Finally, two stopping criteria are placed: after 100 generations or the best chromosome is not improved after 50 generations. The above procedure is described in Figure 4.

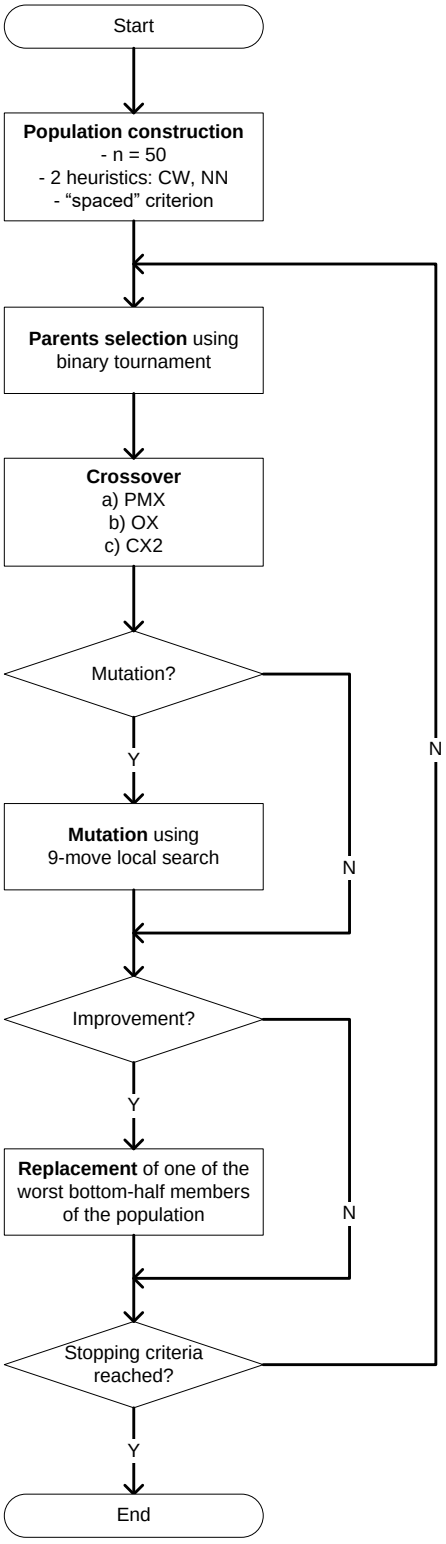


Figure 4. Flow of the GA in brief

4. Data Collection

We used C101 from Solomon benchmark instances in this research. The data contain 100 nodes and 25 vehicles, each with a capacity of 200. The C101 instance comes with time windows that are not of interest of this research. Therefore, comparison to the known best solution (KBS) of that instance is not given since the KBS assumes the inclusion of time windows. Additionally, since the GA involves probability in some parts, e.g. selection of parent chromosomes or the trigger of mutation, we ran the testing of each crossover operator 10 times with controlled random seed numbers for future benchmark. All experiment is run on Intel i7-8565U CPU running on 1.80/1.99 GHz and 8 GB of RAM. Results taken are the fitness value of the best chromosome (the lower the better) during each iteration and computation time which will be analyzed in the next section.

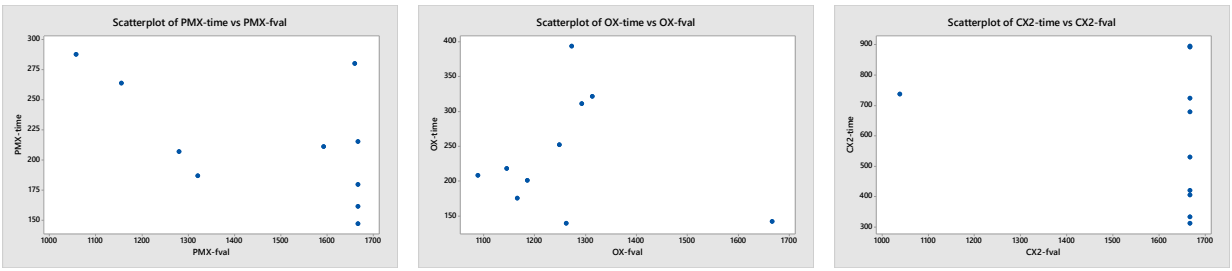
5. Results and Discussion

From the figures in Table 1, we can observe the following results. First, CX2 is more expensive in computation time compared to PMX and OX. In addition, it has a tendency to be trapped in a local optimum and have a difficulty to break away. The fitness value of 1666.53 is obtained from NN heuristic, which means that it is the best fitness value found from the initial population construction stage. Since we use as one of the stopping criteria to end the process if no improvement is made after 50 iterations, it stands to reason that the computation time of CX2 should be lower than the other operators, but this is not the case which indicates some inefficiency in the overall algorithm when CX2 is used as the crossover operator. However, interestingly, the best fitness value is obtained using this operator with a gap of 1.80% to PMX and 4.79% to OX.

Comparing PMX and OX, we obtained information that on average OX performs better than PMX in terms of fitness value but has a slightly larger average of run time, although the better fitness sides with the PMX. The range of run times is also larger on OX than on PMX. With regard to the inability of escaping from a local optimum, PMX ranks second after CX2. In general, the results from OX seem more diverse compared to the other two operators or in other words it is least prone to being trapped in a local optimum, hence more fine tuning could probably still improve the GA performance using this crossover operator. Figure 5 showing the scatter plot between the fitness value and run time which also confirms our observation on the "positive" variation in OX.

Table 1. Experiment results

Run no.	**PMX**		**OX**		**CX2**	
	Fitness	Run time (sec)	Fitness	Run time (sec)	Fitness	Run time (sec)
1	1666.53	146.89	1666.53	142.20	1666.53	332.55
2	1666.53	179.43	1145.72	217.90	1666.53	529.78
3	1320.96	186.54	1262.39	139.52	1666.53	419.35
4	1666.53	161.22	1186.50	201.08	1666.53	312.35
5	1280.77	206.93	1165.91	175.42	1666.53	405.61
6	1592.28	211.09	1088.68	208.31	1666.53	894.22
7	1666.53	215.18	1292.68	311.06	1038.88	736.78
8	1057.63	287.79	1314.02	321.20	1666.53	891.98
9	1659.71	280.13	1248.95	251.90	1666.53	723.26
10	1155.67	263.67	1272.91	393.12	1666.53	677.76
Avg.	1473.31	213.89	1264.43	236.17	1603.77	592.36
Min.	1057.63	146.89	1088.68	139.52	1038.88	312.35
Max.	1666.53	287.79	1666.53	393.12	1666.53	894.22



a) Scatter plot of PMX b) Scatter plot of OX c) Scatter plot of CX2

Figure 5. Scatter plots of fitness value vs. run time

6. Conclusion

This paper studies the impact of crossover operators in genetic algorithm applied on one type of routing problems, i.e. the vehicle routing problem. The GA principles follow the same principles from one literature which are able to produce good performance with careful implementation. Three crossover operators are studied: the partially mapped crossover (PMX), the order crossover (OX), and the modified cycle crossover (CX2). Because randomness is involved in some aspects of the algorithm, each operator is run 10 times using different random seeds. C101 from Solomon data set is used as the benchmark instance and performance evaluated is in fitness value and computation time.

Conclusion from the experiment is diverging. Different operator has distinct characteristics in each of the performance being evaluated. For example, the CX2 operator is expensive in computation time and the worst in terms of its ability to escape from a local optimum, but despite the lack of variability, it is able to find the best fitness value compared to the other two operators. On the other hand, the other two operators, while are much efficient in run time and have more variability in achievement of fitness value, lead to the best fitness values which are 5% less than that of the CX2. However, more variability observed in the results of OX indicates its strength to explore wider search space and fine-tuning other parameters with this operator paves future direction for the quest to seek a better GA.

References

Awad, H., Elshaer, R., AbdElmo'ez, A., and Nawara, G., An Effective Genetic Algorithm for Capacitated Vehicle, *Proceedings of the International Conference on Industrial Engineering and Operations Management*, Bandung, Indonesia, March 6-8, 2018.

Borhani, M., Evolutionary multi-objective network optimization algorithm in trajectory planning, *Ain Shams Engineering Journal*, vol. 12, no. 1, pp. 677-686, 2021.

Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, New York, 1989.

Grosch, B., Kohne, T., and Weigold, M., Multi-objective hybrid genetic algorithm for energy adaptive production scheduling in job shops, *Procedia CIRP*, vol. 98, pp. 294-299, 2021.

Holland, J. H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Michigan, 1975.

Hussain, A., Muhammad, Y. S., Sajid, M. N., Hussain, I., Shoukry, A. M., and Gani, S., Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator, *Computational Intelligence and Neuroscience*, vol. 2017, pp. 1-7, 2017.

Laporte, G., and Semet, F., Classical and new heuristics for the vehicle routing problem, in Toth, P., and Vigo, D. (Eds.), *The Vehicle Routing Problem*, SIAM, Philadelphia, pp. 109-128, 2002.

Nazif, H., and Lee, L. S., Optimised crossover genetic algorithm for capacitated vehicle, *Applied Mathematical Modelling*, vol. 36, no. 5, pp. 2110-2117, 2012.

Prins, C., A simple and effective evolutionary algorithm for the vehicle routing problem, *Computers & Operations Research*, vol. 31, no. 12, pp. 1985-2002, 2004.

Prins, C., Two memetic algorithms for heterogeneous fleet vehicle routing problems, *Engineering Applications of Artificial Intelligence*, vol. 22, no. 6, pp. 916-928, 2009.

Prins, C., Lacomme, P., and Prodhon, C., Order-first split-second methods for vehicle routing problems: A review, *Transportation Research Part C*, vol. 40, March 2014, pp. 179-200, 2014.

Saif-Eddine, A. S., El-Beheiry, M. M., and El-Kharbotly, A. K., An improved genetic algorithm for optimizing total supply chain cost in inventory location routing problem, *Ain Shams Engineering Journal*, vol. 10, no. 1, pp. 63-76, 2019.

Sergeeva, M., Delahaye, D., Mancel, C., and Vidosavljevic, A., Dynamic airspace configuration by genetic algorithm, *Journal of Traffic and Transportation Engineering* (English Edition), vol. 4, no. 3, pp. 300-314, 2017.

Vizcaíno-González, M., Pineiro-Chousa, J., and López-Cabarcos, M. Á., Analyzing the determinants of the voting behavior using a genetic algorithm, *European Research on Management and Business Economics*, vol. 22, no. 3, pp. 162-166, 2016.

Wibisono, E., and Jittamai, P., Multi-objective evolutionary algorithm for a ship routing problem in maritime logistics collaboration, *Int. J. Logistics Systems and Management*, vol. 28, no. 2, pp. 225-252, 2017.

Acknowledgment

The authors wish to thank The Ministry of Education and Culture and The Ministry of Research and Technology of The Republic of Indonesia for funding this research under the scheme Penelitian Dasar Unggulan Perguruan Tinggi no. 003/AMD-SP2H/LT-MULTI-PDPK/LL7/2021, 004/SP-Lit/AMD/LPPM-01/Dikbudristek/Multi/FT/VII/2021.

Biographies

Eric Wibisono is an Associate Professor and a faculty member at the Department of Industrial Engineering, University of Surabaya. He obtained his B.Eng. in Industrial Engineering from University of Surabaya, Indonesia, M.Eng. in Manufacturing Management from University of South Australia, Australia, and Ph.D. in Industrial Engineering from Suranaree University of Technology, Thailand. His research interests are in the field of logistics routing and multi-objective optimization. He is a member of the Indonesian Supply Chain and Logistics Institute.

Iris Martin obtained her B.Eng. in Industrial Engineering from University of Surabaya, Indonesia. She is currently pursuing her M.Eng. in the same school at the same university. Her research interests are in logistics and routing problems.

Dina Natalia Prayogo is an Associate Professor of Industrial Engineering at University of Surabaya, Indonesia. She received her master's degree in Industrial and Systems Engineering from The National University of Singapore and her bachelor's degree in Physics Engineering from Institut Teknologi Sepuluh Nopember (ITS) Surabaya, Indonesia. Her research interests are in container terminal operations, supply chain engineering, and disaster management.