Solving the Goods Transportation Problem Using Genetic Algorithm with Nearest-Node Pairing Crossover Operator

Amirah Rahman, Nazmi Syazwan Shahruddin and Ismail Ishak
School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia

Corresponding author: amirahr@usm.my

Abstract. Goods transportation is a critical part of supply chain management and the distance covered in the delivery process indirectly reflects the sustainability level of the supply chain, especially in the environmental aspect. In the wake of climate change issues faced worldwide and the increasing public concern in pollution reduction, it would be in the best interest of companies to not just minimize their operational costs but to also do their part in reducing externalities such as air pollution and noise pollution. In this study, a goods transportation problem is tackled to reduce the distance covered in the transportation process by modelling the problem as a Travelling Salesman Problem (TSP) and solving it using Genetic Algorithm. We propose a new crossover operator namely the Nearest-Node Pairing Crossover (NNPX) that is specifically designed to tackle a Travelling Salesman Problem (TSP) by exploiting the distance aspect of the problem. We evaluate the performance of NNPX compared to two other crossover operators: Order Crossover (OX) and Position-Based Crossover (PBX). The results reveal that the performance of NNPX is outstanding compared to OX and PBX. We found that NNPX has a better rate of convergence as it consistently yields lower distances in fewer iterations. In addition, NNPX does not depend on a large population size for faster convergence. In a nutshell, this study proposes a new crossover operator NNPX that is comparatively more efficient when used to solve the goods transportation problem, thus reducing the associated operational cost and externalities.

1. Introduction
One of the most common routing problems involves planning the delivery route from distribution centres to consumers. This final stage within the logistics network is more commonly referred as last-mile logistics, and is often in the junction of profitability and environmental sustainability which encompass consuming fewer resources and emitting less pollution. Fortunately, addressing delivery operations can benefit both sides. Minimizing route distance means lesser fuel consumption, which would lessen both operations cost and harmful gas emissions. This will also reduce the time consumption of the delivery operation, thereby increasing the efficiency of the delivery operation which is important to the reputation of any delivery service provider.

Good transportation problems are usually modelled and solved as Vehicle Routing Problems (VRPs) as they involve routing multiple capacitated vehicles on a network. The VRP is known to be an extension of the Travelling Salesman Problem (TSP), which routes a single uncapacitated vehicle on a network. In this study, we consider a goods transportation problem that involves routing a single delivery vehicle that departs from a particular depot, visits predetermined consumer drop points before returning to the
same depot. Also, the problem we study is one where vehicle capacity is never an issue. That is, the vehicle will never have to carry more than it can hold. Thus, we model our uncapacitated single vehicle goods transportation problem as a Travelling Salesman Problem (TSP). The TSP is an NP-hard combinatorial optimisation problem and its purpose is to find the shortest route in a network, where every node in the network is visited once, except for the initial node (depot), which is visited twice. In a goods transportation problem, the purpose of the TSP is to find the shortest route for the delivery agent to distribute the goods at every drop point, starting and ending the delivery process at the distribution depot. The complexity of TSPs increase as the number of nodes in networks increase. Continuous research has been done in this field to find the best method in tackling TSPs. Generally, approximate approaches are simple and require less computational time but lack accuracy in the solutions obtained, while exact approaches are complex and time consuming but produce results whose closeness to optimality can be verified.

Exact methods are able to guarantee the closeness of the obtained solution to the optimal solution, which is a much-desired trait to any decision maker. However, they have one major weakness: the enormous computational effort needed to solve the problem. One of the well-known exact methods is the Branch-and-Bound technique, which was used by [1] to solve the asymmetric cost matrix of a TSP by implementing it in Java. Different number of cities were tested and as the problem size increased, the average computational time also increased. Dynamic Programming methods were proposed by [2] to solve a TSP with 25 cities, where the study was able to provide the optimal solution given the problem complexity and smaller size.

Approximate methods do not guarantee the solution they provide to be optimal, but they are surely time efficient when tackling more complex problems. The study by [3] proposed combining Fuzzy Particle Swarm Optimisation with Simulated Annealing (SA) and neighbourhood information communication to solve a TSP, and it was reported to have better convergence speed and produce better quality solutions compare to basic SA and Ant Colony Optimization. The study by [4] carried out a comparative study on the Nearest Neighbourhood method and Genetic Algorithm (GA). The authors reported that the performance of GA was much better when it came to larger problems.

The idea of implementing GA to solve TSPs was introduced by [5]. More recently, the study by [6] solved a TSP using GA with a greedy genetic operator. The proposed method was capable of solving large problems with small population usage and has escaped from local optima many times. The study by [7] also used GA to solve a TSP and the authors concluded that GA is a good approach to solve TSPs, but it depends on the problem encoding and the type of genetic operator used. The study by [8] introduced a new crossover named Common Sub Paths Crossover for GA to solve TSP. It was found to perform much better compared to traditional crossover operators. The study by [9] too proposed a new crossover names the modified cycle crossover operator, which was found to work well on several TSPLIB instances.

In this study, we tackle the goods transportation problem with a single delivery agent in a goods transportation network with a single depot. We model the problem as a Travelling Salesman Problem, and solve using Genetic Algorithm (GA). We propose a new crossover operator for GA called the Nearest-Node Pairing Crossover (NNPX), which we compare with two established crossovers: Order Crossover (OX) and Position-Based Crossover (PBX).

2. Problem Description

We study a goods transportation problem where goods are being transported within the following areas: Sungai Dua, Gelugor and Minden on Penang Island. We assume that the good originate from Peninsular Malaysia. The distribution network of this area (shown in Appendix A, Figure A.1) consists of 15 nodes, each of which may represent a single drop point or multiple drop points grouped together due to their geographical proximity. We choose the starting and ending node to be node 12 as it is closest to the Penang bridge. The drop points represent places of business such as sundry shops or convenience stores that carry a similar range of products. The shortest distance between each pair of nodes is calculated (in
meters) using Google Maps. In this study, the goods transportation problem is modelled as a TSP and solved using Genetic Algorithm.

3. Methodology
Genetic Algorithm (GA) is inspired by the principles of genetic and evolution. It mimics the reproduction behaviour in living beings. Every species starts with an initial population and passes their traits to the next generation via reproduction and mutation. Each chromosome represents a feasible route for the TSP. In this study, the initial population is created randomly and the fitness value of each chromosome is the total distance of the route. In the reproduction process, the selection operator used is tournament selection and the mutation operator used is swap mutation. We set a maximum number of iterations as the termination condition.

We propose a new crossover operator named the Nearest-Node Pairing Crossover (NNPX), specifically designed to tackle a Travelling Salesman Problem (TSP). We compare the performance of our proposed crossover operator NNPX against two established crossover operators used in solving TSPs: Order Crossover (OX) and Position-Based Crossover (PBX).

3.1. Nearest-Node Pairing Crossover
Our proposed Nearest-Node Pairing Crossover (NNPX) aims to exploit the distance aspect of the TSP. The following is a step-by-step explanation of how to use NNPX to obtain Child \( i \) from Parent \( i \):
1. Randomly select genes from Parent \( i \) and transfer directly to Child \( i \) in the same chromosomal location.
2. If Child \( i \)’s first gene is unassigned, randomly select a remaining gene from Parent \( i \) and assign it to this chromosomal location.
3. For each of Child \( i \)’s other unassigned genes, do the following:
   (a) Obtain the gene immediately preceding (to the left of) the current unassigned gene.
   (b) Select the remaining gene from Parent \( i \) that is closest distance-wise to the gene obtained in (a).
   (c) Assign the selected gene to the chromosomal location of the current unassigned gene.

Note that ties are broken lexicographically, and that the set of Parent \( i \)’s remaining genes shrinks whenever a gene is selected to be assigned to Child \( i \).

We now demonstrate this process via the example shown in Figure 1. The left side of Figure 1 illustrates NNPX at work on a network with 6 nodes: A, B, C, D, E, and F, which is shown on the right side. The numbers on the arcs represent the distance between the pair of nodes connected by the arc.

![Figure 1. Visual Illustration of NNPX at Work](image)

The process begins by selecting a random set of genes (E, C and B) from Parent 1 and transferring them directly to Child 1 in the same chromosomal location. As the first gene in the child is unassigned, we randomly select the gene F from set of remaining genes (A, D and F). For all other unassigned genes, we select the gene that is closest in terms of distance to the gene on the left. For example, the gene on
the right to gene E is unassigned. Of the two remaining genes (A and D), D is selected as it is closer to E in distance compared to A (1 unit vs 2 units). This is repeated until all genes have been selected. Any ties are broken lexicographically. The same process is performed again to create a Child 2 from Parent 2, as our selection operator creates a pair of parents at a time. Order Crossover and Position-Based Crossover.

Order Crossover (OX) was introduced by [10] and it requires both parents to create both children. The first step is to randomly select a substring of genes from Parent 1 and pass it down to Child 1. Next, the genes already passed down from Parent 1 are eliminated from Parent 2 and the remaining genes in Parent 2 are passed down to the child by filling the empty slots according to the order from left to right. Child 2 is created similarly using Parent 2 as the main parent.

Position-Based Crossover (PBX) was introduced by [11] and it also requires both parents to create both children. First, randomly select a set of genes from Parent 1 (not necessarily a substring of genes), which are transferred to the Child. Next, the genes already passed down from Parent 1 are eliminated from Parent 2 and the remaining genes in Parent 2 are passed down to Child 1 by filling the empty slots according to the order from left to right. Child 2 is created similarly using Parent 2 as the main parent. Figure 2 illustrates OX (left) and PBX (right) at work for a network similar to that shown in Figure 1.

4. Results and Discussion

In this study, we use Genetic Algorithm with the following parameters: tournament size of 2, mutation rate of 5% and termination condition is set to be 10000 iterations. We create 30 instances per crossover operator, where the initial population for all 30 instances are determined in advance. This is to ensure all crossover operators have the same starting point. We implement our solution method in MATLAB R2013b, executed on a machine with Intel® Core™ i5-4210U @ 1.70 GHz and 4.00 GB RAM. An illustrative example of a route obtained by solving our problem is shown in Appendix A, Figure A.2

We perform the following experiments to test the performance of our proposed crossover operator NNPX against established crossover operators OX and PBX:

- Explore the effect of varying population size on route distance and computation time.
- Compare the convergence rates of all crossover operators using a population size of 10.

4.1. Effect of Varying the Population Size

We experiment by setting the population size to be 10, 25, 50, 100 and 200. Thus, we perform a total of 5 (population sizes) × 3 (crossover operators) × 30 (instances) = 450 total test runs. We explore the performance of each crossover operator on route distance and computation time.

4.1.1. Route Distance. Table 1 shows the best, worst and average route distances obtained from each (crossover operator, population size) combination. Observe that the best distance obtained for all combinations is 19090 meters. However, the worst and average distances obtained varies wildly between the combinations. It can be seen that for both OX and PBX, the worst and average distances obtained
reduce with the increase of population size. However, this is not the case for NNPX as it delivers solutions consistently within a small range of values (< 1000 meters) regardless of population size. This observation is confirmed in Figure 3, that shows the comparison of average route distance between all (crossovers, population size) combinations.

Table 1. Best, Worst and Average Route Distances Obtained from each (Crossover Operator, Population Size) Combination

| Crossover | Population Size | 10   | 25   | 50   | 100  | 200  |
|-----------|-----------------|------|------|------|------|------|
| OX        | Best            | 19090| 19090| 19090| 19090| 19090|
|           | Worst           | 21960| 22370| 22970| 21540| 20140|
|           | Average         | 20530.33 | 20479.33 | 20499.67 | 19845.00 | 19339.00 |
| PBX       | Best            | 19090| 19090| 19090| 19090| 19090|
|           | Worst           | 22200| 22510| 22060| 22670| 20830|
|           | Average         | 20775.33 | 20654.33 | 20175.67 | 19996.33 | 19569.33 |
| NNPX      | Best            | 19090| 19090| 19090| 19090| 19090|
|           | Worst           | 19610| 19990| 19610| 19610| 19610|
|           | Average         | 19375.33 | 19422.67 | 19373.33 | 19291.67 | 19264.33 |

Figure 3. Comparison of Average Route Distance between All (Crossovers, Population Size) Combinations
4.1.2. Computational Time. Figure 4 shows the comparison of computation time between all (crossovers, population size) combinations. It can be seen that the population size has very little effect on computation time, but the crossover operator used gives the most impact. OX has the least average computation time and PBX is in second place. NNPX has the greatest average computation time and is about twice as much as OX, an increase of about 2 seconds. This is probably due to the complexity of the NNPX. However, recall that NNPX yields much better results in the additional 2 seconds it takes that is longer than OX.

![Figure 4](image)

Figure 4. Comparison of Computation Time between All (Crossovers, Population Size) Combinations

4.2. Solution Convergence
Solution convergence refers to the question of how many iterations are needed for the route distance obtained to reach its minimum value. Thus, the solution convergence is better when fewer iterations are needed. In this study, we use the population size of 10 in performing this convergence experiment. The route and its distance are recorded every 500 iterations for each of the 3 (crossover operators) × 30 (instances) = 90 test runs.

Figure 5 shows the average route distance for each crossover operator at 500 iteration intervals. It can be seen that OX has the slowest convergence rate among the three crossovers. PBX is not far ahead. Our proposed crossover NNPX however vastly outperforms the others by vastly reducing the route distance within its first 500 iterations.

Taking every test run into account, OX and PBX may need greater than 10,000 iterations to explore further into the solution space to reduce the route distance obtained to the level that NNPX has reached. On the other hand, NNPX stops making significant improvement to the route distance after 1500 iterations. Thus, we can infer that NNPX would need fewer than 2000 iterations to obtain a good enough solution to a TSP such as this.
5. Conclusion

In this study, we have modelled a goods transportation problem by as a TSP and were able to solve it using Genetic Algorithm. Tackling problems such as these have effect on reducing further damage on the environment, even if just a little, as minimising the route distance taken in the goods transportation operation would directly contribute to lesser fuel consumption and lesser harmful gas emissions. Applying environmentally-friendly solutions to their goods transportation system would bring benefit in terms of both operational cost reduction and increase in brand reputation.

We solved our problem using Genetic Algorithm and proposed a new crossover operator named the Nearest-Node Pairing Crossover (NNPX) that exploits the distance aspect of the TSP. We investigate the performance of NNPX and established crossover operators Order Crossover (OX) and Position-Based Crossover (PBX) in two ways: (1) Explore the effect of varying population size on route distance and computation time, and (2) Compare the convergence rates of all crossover operators using a population size of 10.

In terms of varying the population size, we found that OX and PBX rely heavily on the population size in terms of the quality of the solution obtained, while NNPX consistently yielded good quality solutions regardless of population size. However, NNPX consumed at least twice as much computation time when compared to OX, with PBX in second place. The extra time amounted to an extra 2 seconds (on average) to compute a total of 10,000 iterations. Due to NNPX yielding much better solutions and 2 more seconds being a short amount of time, we claim NNPX still outperforms both OX and PBX.
We found that NNPX converges much faster compared to OX and PBX on average. NNPX was shown on average to vastly reduce the route distance within its first 500 iterations, needing fewer than 2000 iterations to obtain a good solution. We posit that this is because NNPX considers the distance aspect, while OX and PBX are more random in their crossover process causing them to possibly need more than 10,000 iterations to obtain a good solution.

While NNPX does converge early and works well in this particular TSP setup, we do believe there is a possibility that it may lead to premature convergence and become trapped in local optima. This limitation can be tackled in future work by hybridising NNPX with a crossover with a more random process, or by applying mutation operators more aggressively.

Acknowledgements
The research work is supported by the FRGS research grant 203/PMATHS/6711571.

References
[1] Kalczynski, P. 2005. *Journal of Object technology*, 156-163.
[2] Chauhan, C., Gupta, R., & Pathak, K. (2012). *International Journal of Computer Applications*, 12-19.
[3] Kader, R. F. 2013 *International Journal of Advanced Computer Science and Applications*, 987-997.
[4] Al Salibi, B. A., Jelodar, M. B., & Venkat, I. 2013. *International Journal of Computer Science and Electronic Engineering (IJCSEE)*, 34-38.
[5] Brady, R. M. 1985 *Nature*, 317, 804.
[6] Bhagel, A. S., & Rastogi, R. 2011 *International Journal of Advanced Engineering Research*, 1-16.
[7] Khattar, S., & Gosawmi, P. (2014). A solution of genetic algorithm for solving travelling salesman problem. *International Journal for Scientific Research Development*, 341-343.
[8] Mendiratta, K., & Goyal, A. (2014). Implementation of CSP crossover in solving travelling salesman problem using genetic algorithm. *International Journal of Engineering Tredns and Technology (IJETT)*, 455-459.
[9] Hussain, A., Muhammad, Y. S., Sajid, M. N., Hussain, I., Shoukry, A. M., & Gani, S. 2017. *Computational Intelligence and Neuroscience*.
[10] Davis, L. 1985. Applying adaptive algorithms to epistactics domains. *Proceedings of the International Joint Conference on Artificial Intelligence*. Los Angeles.
[11] Syswerda, G. 1991. Schedule optimization using genetic algorithm. *Handbook of genetic algorithm*, 332-349.
Appendix A

Figure A.2. Example of Route Obtained after Solving Goods Distribution Network