Travelling salesman problem: Greedy single point crossover in ordinal representation

B Fernandez¹*, A Fanggidae², E S Y Pandie² and A Y Mauko²

¹ Graduate of Computer Science, University of Nusa Cendana, Jl. Adisucipto No 10, Kupang, Nusa Tenggara Timur, Indonesia
² Department of Computer Science, Faculty of Science and Engineering, University of Nusa Cendana, Jl. Adisucipto No 10, Kupang, Nusa Tenggara Timur, Indonesia

*bonaventurafernandez@gmail.com

Abstract. Genetic Algorithm (GA) is a metaheuristic algorithm which has stages; coding, selection, crossover, and mutation. The crossover operator plays an important role in producing offspring to find solutions. Single Point Crossover (SPC) and Two Point Crossover (TPC) are classic crossover operators that are easy to apply to ordinal representation coding rather than path representation coding. The greedy algorithm is applied to the SPC operator to solve the Traveling Salesman Problem (TSP) to improve the performance of the genetic algorithm in the ordinal representation coding scheme. The results show that the proposed Greedy Single Point Crossover (GSPC) has higher performance in finding the global optimal solution for all population sizes and cities used, but it costs computation time.

1. Introduction

The Traveling Salesman Problem (TSP) is one of the optimization problems, which reflects the routing decisions that a salesman must make [1]. The routing decision is the minimum total distance a salesman must travel from the starting point to multiple points exactly once and back to the starting point [1,2]. TSP optimization problem becomes complex if it involves multiple points.

Optimization algorithms are often used in solving complex optimization problems. Optimization algorithms that are of great interest in research are stochastic algorithms. Stochastic algorithms have random components that produce different outputs for certain inputs [3]. Stochastic algorithms with randomization and global exploration are known as metaheuristics [4]. The metaheuristic algorithm finds the global optimal solution from many local solutions obtained from randomization.

The most famous metaheuristic algorithms are Genetic Algorithms (GA), Simulated Annealing (SA), and Tabu Search (TS) [5]. GA was first introduced by John Holland in 1975 with its simple, universal, and robust advantages [6]. GA performance depends on the coding scheme and the choice of genetic operators especially, selection, crossover and mutation [7]. The crossover operator is very important in GA, because it is used to exchange information during the search for a solution [8].

Single Point Crossover (SPC) is one of the simplest classical crossover operators. Path representation is the representation most often used to solve TSP problems using Genetic Algorithms, the reason lies in its intuitive representation and good results [9]. Unfortunately, this representation cannot be applied to classical crossover operators, because the resulting children may have duplicate alleles, which means
the loss of another point, and this violates the TSP concept. The following examples are children who
have duplicate alleles that result from the selected parents.

Parent 1: 1 2 4 | 3 5 6
Parent 2: 5 1 6 | 4 2 3
Child 1: 1 2 4 | 4 2 3
Child 2: 5 1 6 | 3 5 6

Classics crossover and mutation operators can use ordinal representation coding but the experimental
results have been generally poor [9]. A greedy algorithm is an algorithm that when solving a problem
always makes the choice that seems best at that moment, the choice is optimal locally in the hope that it
will lead to a globally optimal solution [10].

The performance of GA in the ordinal representation coding scheme needs to be improved in finding
the global optimal solution using the greedy algorithm on the SPC operator.

2. Methods

2.1. Coding

The seller travels to \( n \) cities, \( n \) cities are arranged sequentially in the set \( C \), the tour are represented in
the set \( L \), and chromosome encodings are represented in set \( P \).

Encode:

- Set \( P = \{ \} \). Example the tour = 3-8-5-1-6-4-7-2, then \( L = \{3,8,5,1,6,4,7,2\} \) and \( C = \{1,2,3,4,5,6,7,8\} \)
- Take the first city \( (L_1) \) in \( L \), take the index \( L_1 \) in \( C \) and save it in \( P \), and remove \( L_1 \) from \( C \).
- Take the second city \( (L_2) \) in \( L \), take the index \( L_2 \) in \( C \) and save it in \( P \), and remove \( L_2 \) from \( C \).
- Continue the same way until the set \( C = \{\} \), ordinal representation = \( P \)

Decode:

- Set \( L = \{\} \). Example code \( P = \{3,7,4,1,3,2,2,1\} \) and \( C = \{1,2,3,4,5,6,7,8\} \)
- Take the first code \( (P_1) \) in \( P \), take the city on the \( P_1 \)th index in \( C \) and save it in \( L \), and remove \( P_1 \)
  from \( C \).
- Take the second code \( (P_2) \) in \( P \), take the city on the \( P_2 \)th index in \( C \) and save it in \( L \), and remove the
  \( P_2 \) from \( C \).
- Continue the same way until the set \( C = \{\} \), tour = \( L \)

2.2. Fitness

The fitness function in TSP is the total distance traveled by the salesman. If each chromosome is
represented in \( X = (X_1, X_2, ..., X_n) \), then the fitness function is defined in equation (1) [11]:

\[
f(X) = \left( \sum_{i=1}^{n-1} D(X_i, X_{i+1}) \right) + D(X_n, X_1) \tag{1}
\]

where \( D(X_i, X_{i+1}) \) is the distance from city \( i \) to \( j \) or otherwise. If \( pop \) represents the number of
populations, then the best fitness in the population is determined by the equation (2).

\[
best = \min \left( f_1(X), f_2(X), ..., f_{pop}(X) \right) \tag{2}
\]
2.3. Parameters
GA performance is measured from the results obtained from testing using parameters; number of competing chromosomes = 2, crossover probability $pc = 0.2$, mutation probability $pm = 0.01$, and trial $= 1000$. In each experiment, several parameters were changed; total population (25, 50, 75 and 100), and the number of cities (20, 30, 50 and 80). The stopping criteria; the number of generations $jg = 1000$, and convergence $pk = 90\%$.

2.4. Population initialization
The initial population is generated randomly based on the size of the population and the number of cities. The same population was tested on three crossover methods. The allele of each chromosome is generated with a random integer [1,50].

2.5. Selection
The selection operator selects the chromosomes in the current population based on their fitness values. In the tournament selection, the number of tournaments is the same as the population. In each tournament, there are several coromosomes that will compete, the winner of the competition is the chromosome with the best fitness, then they are put in the mating pool.

2.6. Crossover
The crossover operator aims to produce child chromosomes that have a randomly selected sequence of genes from the chromosomes of the parents. Figure 1 is a complete graph connecting each city.

![Figure 1. Complete graph for $n = 5$.]

2.6.1. Single Point Crossover (SPC)
One point is chosen randomly, exchange genes between the two parents starting from the first point to the $n^{th}$ gene, as shown in figure 2.

![Figure 2. Single point crossover.]

2.6.2. Two Point Crossover (TPC)
Two points are selected randomly, exchange genes between the two parents starting from the first point to the second point, as shown in figure 3.
2.6.3. Greedy Single Point Crossover (GSPC)

It is the same as SPC, except that one by one the selected genes are checked before being exchanged. Check one by one the selected genes from two parents, if there is an allele that has the shortest path, it is replaced. Figure 4 shows the process of producing children with GSPC, the calculation of the path distance between cities can see figure 1.

![Figure 3. Two point crossover.](image)

| Parent 1 | Parent 2 | Child 1 | Child 2 |
|----------|----------|---------|---------|
| 3 4 3 2 1 | 4 1 2 1 1 | 3 4 2 1 1 | 4 1 3 2 1 |

Figure 3. Two point crossover.

```
2.6.3. Greedy Single Point Crossover (GSPC)

It is the same as SPC, except that one by one the selected genes are checked before being exchanged. Check one by one the selected genes from two parents, if there is an allele that has the shortest path, it is replaced. Figure 4 shows the process of producing children with GSPC, the calculation of the path distance between cities can see figure 1.

![Figure 4. Greedy single point crossover.](image)

| Parent 1 | Parent 2 | Child 1 | Child 2 |
|----------|----------|---------|---------|
| 3 4 3 2 1 | 4 1 2 1 1 | 3 4 2 1 1 | 4 1 3 2 1 |

Figure 4. Greedy single point crossover.

2.7. Mutation

Mutation randomizes the allele of the selected gene, usually the selected gene in a small number. Figure 5 shows the allele boundaries for each gene in the chromosome in ordinal representation.

![Figure 5. Allele boundaries for each gene.](image)

If ‘gene 3’ is selected in the mutation, then the allele is replaced with a random integer [1, n-2].
3. Results
The test uses the parameters mentioned in section 2.3. Table 1, table 2, table 3, and table 4 show the best average fitness and computation time. The blue color represents the winners of computation time and the green shows the winners of fitness.

**Table 1. GA results for city = 20.**

| Population | Method SPC | Method TPC | Method GSPC |
|------------|------------|------------|-------------|
|            | Fitness    | Time (seconds) | Fitness    | Time (seconds) | Fitness    | Time (seconds) |
| 25         | 333.8      | 0.02        | 334.56     | 0.02        | 318.53     | 0.22         |
| 50         | 314.49     | 0.04        | 314.29     | 0.02        | 300.18     | 0.24         |
| 75         | 296.86     | 0.05        | 297.68     | 0.03        | 285.16     | 0.35         |
| 100        | 291.02     | 0.05        | 289.54     | 0.04        | 277.44     | 0.40         |

**Table 2. GA results for city = 30.**

| Population | Method SPC | Method TPC | Method GSPC |
|------------|------------|------------|-------------|
|            | Fitness    | Time (seconds) | Fitness    | Time (seconds) | Fitness    | Time (seconds) |
| 25         | 533.91     | 0.03        | 532.8      | 0.02        | 500.62     | 0.28         |
| 50         | 500.05     | 0.05        | 500.84     | 0.03        | 470.86     | 0.50         |
| 75         | 472.75     | 0.08        | 473        | 0.06        | 453.11     | 0.71         |
| 100        | 462.43     | 0.10        | 460.76     | 0.09        | 445.91     | 0.83         |

**Table 3. GA results for city = 50.**

| Population | Method SPC | Method TPC | Method GSPC |
|------------|------------|------------|-------------|
|            | Fitness    | Time (seconds) | Fitness    | Time (seconds) | Fitness    | Time (seconds) |
| 25         | 958.85     | 0.06        | 959.8      | 0.04        | 884.62     | 0.79         |
| 50         | 922.6      | 0.08        | 922.7      | 0.05        | 847.91     | 1.11         |
| 75         | 886.59     | 0.15        | 885.44     | 0.13        | 831.8      | 1.53         |
| 100        | 874.08     | 0.21        | 876.92     | 0.19        | 821.72     | 1.95         |

**Table 4. GA results for city = 80.**

| Population | Method SPC | Method TPC | Method GSPC |
|------------|------------|------------|-------------|
|            | Fitness    | Time (seconds) | Fitness    | Time (seconds) | Fitness    | Time (seconds) |
| 25         | 1629.88    | 0.09        | 1629.03    | 0.06        | 1471.96    | 1.89         |
| 50         | 1607.37    | 0.11        | 1605.72    | 0.09        | 1421       | 1.93         |
| 75         | 1564.01    | 0.29        | 1561.56    | 0.27        | 1399.49    | 3.03         |
| 100        | 1548.79    | 0.41        | 1552.63    | 0.39        | 1385.91    | 3.54         |

The test results show that the GSPC operator wins in the best fitness, but it costs computation time. The fastest computation time is given by TPC.
4. Conclusions

GSPC wins in best fitness because the checking of candidate two alleles from parents is done before allele exchange, this results in long computation time. TPC has a fast computation time because the allele exchange of the two parents starts from the first point to the second point, where the second point is not always the position of the last gene like SPC.

References

[1] Lagos D C, Mancilla R A, Leal P E and Fox F E 2019 Performance measurement of a solution for the travelling salesman problem for routing through the incorporation of service time variability Ing. E Investig. 39 44–9
[2] Macgregor J N and Ormerod T 1996 Human performance on the traveling salesman problem Percept. Psychophys. 58 527–39
[3] Shabani A, Asgarian B, Gharebaghi S A, Salido M A and Giret A 2019 A New Optimization Algorithm Based on Search and Rescue Operations Math. Probl. Eng. 2019 1–23
[4] Gandomi A H, Yang X-S, Talatahari S and Alavi A H 2013 Metaheuristic Algorithms in Modeling and Optimization Metaheuristic Applications in Structures and Infrastructures (Elsevier) pp 1–24
[5] Abd G, M. A and M. E-S 2014 A Comparative Study of Meta-heuristic Algorithms for Solving Quadratic Assignment Problem Int. J. Adv. Comput. Sci. Appl. 5
[6] Holland J H 1975 Adaptation in Natural and Artificial Systems (University of Michigan Press)
[7] Abdoun O, Abouchabaka J and Tajani C Analyzing the Performance of Mutation Operators to Solve the Travelling Salesman Problem 18
[8] Alzyadat T, Yamin M and Chetty G 2020 Genetic algorithms for the travelling salesman problem: a crossover comparison Int. J. Inf. Technol. 12 209–13
[9] Larrañaga P, Kuijpers C M H, Murga R H, Inza I and Dizdarevic S 1999 Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators Artif. Intell. Rev. 13 129–70
[10] Rashid M H and Mosteiro M A 2017 A Greedy-Genetic Local-Search Heuristic for the Traveling Salesman Problem 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC) 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC) (Guangzhou: IEEE) pp 868–72
[11] Wang L-Y, Zhang J and Li H 2007 An Improved Genetic Algorithm for TSP 2007 International Conference on Machine Learning and Cybernetics 2007 International Conference on Machine Learning and Cybernetics (Hong Kong, China: IEEE) pp 925–8