A Genetic Algorithm for Prioritized Package Delivery Problem

Zonghui Hu[1,✉] **and Zichen Yang**[1,✉]

[1]Tandon School of Engineering, New York University

Abstract

This project report introduces a new prioritized package delivery problem (PPDP) that given topology, demands, and available postmen with different speeds on different roads, we want to minimize the total delivery cost, taking into account the package priorities. A generic algorithm is proposed to solve the PPDP, which integrates the Dijkstra algorithm and different crossover and mutation methodologies. The Dijkstra algorithm aims to find the shortest path from the source node to each possible destination node, which helps build the cost matrix for each postman. Then, a genetic algorithm is applied to find the optimal solution for this question, and experiments with different crossover and mutations are conducted to compare the efficiency. Computational experiments on various instances demonstrate that the proposed algorithm can effectively solve the PPDP.

Code Repo: https://github.com/Ethan-zc/Genetic-Algorithm-For-Package-Delivery

Project Overview

The sub-tasks of this project could be described as follows:

- **Literature Search & Project Planning(Zonghui Hu, Zichen Yang):** Study related papers and approaches to get the scope of this project, complete the project plan.
- **Problem Propose(Zonghui Hu, Zichen Yang):** Propose the prioritized package delivery problem.
- **Implement Dijkstra's algorithm and helping classes(Zonghui Hu, Zichen Yang):** Implement Dijkstra's algorithm to find the shortest time for each postman to deliver packages to different destinations.
- **Implement the generic algorithm(Zonghui Hu, Zichen Yang):** Implement the general model for the genetic algorithm.
- **Implement different crossover and mutation methods(Zonghui Hu, Zichen Yang):** Implement single point, two points, multi points, and uniform as crossover method; insert, inversion, swap, scramble as mutation method.
- **Further Experiments (Zonghui Hu, Zichen Yang):** Adjusting hyperparameters and procedures to conduct more.
- **Preparing Final Presentation & Final Report(Zonghui Hu, Zichen Yang):** Update from the midterm report with accomplishments, the possible improvement, and experiment results.

Introduction

The increasing demand for package delivery services has led to various challenges, including optimizing delivery routes, reducing delivery time, and improving customer satisfaction. To address these challenges, a new prioritized package delivery problem (PPDP) is proposed in this project report. The PPDP considers the priority of packages in addition to the conventional distance between the source node and all possible destinations. Given a restricted last finishing time, the object is to find an assignment that minimizes the total cost of package delivery.

To solve the PPDP, a pipeline is proposed, which is mainly composed of Dijkstra's algorithm and genetic algorithm. For Dijkstra's algorithm, it is mainly used to find the shortest delivery time for each postman from the source node to all possible destination nodes, which would be built as the cost matrix for each postman. By using the randomly generated assignments as the initial population, a genetic algorithm is used to do evolution to find the optimal assignment. Experiments have been done not only with different crossover and mutation methods but also with different crossover and mutation rates on topologies with different complexity.

The rest of the report is organized as follows. Section 2 presents a literature review on related package delivery problems and algorithms. Section 3 formulates the PPDP and proposes the proposed algorithm. Section 4 reports the computational experiments and performance evaluation of the proposed algorithm. Section 5 concludes the report and discusses future research directions.

Literature Survey

Path planning problem, including its real-scenario version, has been studied extensively in the literature to optimize the delivery routes. As a NP-hard problem, the time complexity of brute-force method is unrealistic when applied to complex situation. To find the solution of large scale problems in a reasonable time, heuristic methods have been widely implemented on related problem. Anuj Kumar et al. (2020) solved Travelling Salesman Problem (TSP) by using improved genetic algorithm.(1) The proposed algorithm generated both

random chromosomes and greedy chromosomes using nearest neighbor greedy approach,which would have better fitness. Starting from an efficient initial population, the algorithm found better solutions of typical instances of TSP problems as compared to other existing algorithms.
Similarly, Edwin Dwi Ahmad et al. (2020) used a hybrid of Tabu Search and Simulated Annealing to solve larger scale TSP problems. (2) Though Hyper-heuristic, the method they used, has no guarantee of success in finding the optimal solution, the tuning parameters in such method are done automatically which further shortening the process of finding solutions. Their results showed that the hybrid is even better than Greate Deluge, another heuristic search algorithm.
For problems that are closer to reality, like vehicle routing problem in pharmaceuticals distribution and express route optimization, Gulsun Nakiboglu et al. (2018), Wu Qianqian et al. (2020) and other researchers also prove the efficiency and improvement brought by genetic algorithm or other heuristic algorithms. (3, 4) However, few existing research consider the problem after path routing: the assignment of packages on different transport methods to finish the route. The priority of packages and cost on transport methods is essential in real problem as well.
Therefore, In this project report, we propose a new prioritized PDP that considers the priority of packages as well as the distance and capacity constraints of delivery vehicles. We develop a generic algorithm that integrates a priority-based insertion heuristic and a local search algorithm to optimize the delivery routes and schedules. Our computational experiments demonstrate that the proposed algorithm can effectively solve the PPDP and outperform the existing algorithms in terms of solution quality and computational time.
Compare to heuristic search algorithm, the greedy algorithm guarantee an optimal solution. (5) To make sure the path route is optimal that wouldn't influence the assignment part, we use Dijkstra's algorithm to obtain optimal delivery time matrix for each postman.

Methodology

A. Prioritized Package Delivery Problem. We define the main problem we want to solve as follows: Given the graph topology G with V nodes and E undirected links, we define the source of all demands as node 0. Initially, all packages and available postmen would be in node 0. A demand list would be given in terms of a list, which contains the destination of a certain package. For example, when a graph with 10 nodes and 15 links is given, the demands could be [1,2,3,4,..., 15], where the first package needs to be delivered to node 1, and the second package needs to be delivered to node 2, etc...At the same time, packages have priority. For example, we have demand $[d_1, d_2, d_3, ..., d_D]$, we require that d_x could only be able to deliver when $d_(x-1)$ is one the way of delivery or has been delivered. Meanwhile, there are P available postmen in node 0. Different postmen have different speeds on different links, they could go out and deliver at most one package at a time. If one postman goes out on time 0 and requires time period T to deliver the package assigned to him, the certain postman would be unavailable in the time period (0, 2T), since it would need one more T for the postman to get back from the destination node to the source node 0. Every postman also has an initial cost every time when they are assigned a package to deliver. A regulated time would be given, it is the time that the company expected that all packages are delivered and all postmen to be returned to node 0. When we find a feasible assignment and the last finish time is larger than the regulated time, for every minute there would be an overtime penalty cost. If the last finish time is smaller than the regulated time, then there would be no penalty cost on time. Therefore, when given an assignment, the total cost would be the possible overtime penalty plus the sum of the initial cost of all postmen in the assignment. The object of this question is to find the assignment that minimizes the total cost.
As for the assignment, we define the assignment as a list of postmen. For example, we have an assignment [1,2,3,1,1,3], which means that at time 0, postmen 1, 2, and 3 would go out and deliver the first, second, and third packages, then when Postman 1 is back, he would be in charge of deliver package 4, and then after he returns, Postman 1 would deliver the fifth one and postman 3 would deliver the sixth one at the same time.

B. Graph. For the graph, we define the data structure as an adjacent matrix. When we have a V nodes and E links graph, the topology would be represented as a VxV matrix M, where M[v1, v2] is the distance between node v1 and v2. For M[v1, v2], where v1 == v2, we define it as 0. If v1 is not equal to v2 and M[v1, v2] = 0, it means that there is no connection between v1 and v2.

C. The Dijkstra's Algorithm. Dijkstra's algorithm implemented here is used to find the cost matrix for each postman. The available postmen would be input as a list, and for each postman, he would have a VxV matrix S, where S[v1, v2] represents the speed of the certain postman on the link. For each postman, we want to find the shortest time for him to reach all other nodes rather than node 0. Therefore, we divide the adjacent matrix of the graph with the speed matrix of a postman to find the cost matrix C of the postman. C[v1, v2] is the time, in terms of minutes, the postman would take when he travels from v1 to v2. Dijkstra's algorithm would be used here to find the shortest path from the source node 0 to all other possible destination nodes for every postman based on his cost matrix C. It would return a list SP with a length of V - 1, where SP[i] is equal to the short time for a certain postman to deliver a package to node i.

D. Chromosome Representation. We define our chromosome, which is an individual in the genetic algorithm's population, as one feasible assignment. Because of the way we define our assignment, and given a list with a length equal to the demand, while its elements are in the range [1,]we could randomly generate an assignment with a length equal to the number of demands. We construct a class Chromosome with the method get_chromosome(), which would take the

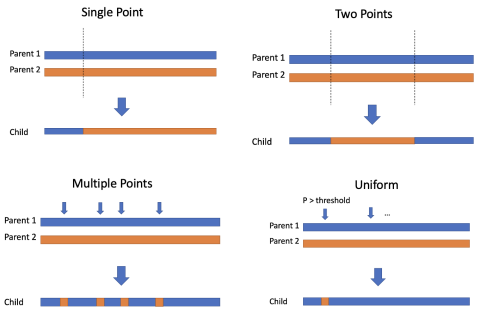

Fig. 1. illustration for four different crossover methods.

randomly generated assignment as input, and return a chromosome object which calculated the last finish time. A list of chromosomes would be used as the initial population of the genetic algorithm.

Genetic Algorithm.

D.1. General Process. The general process of the genetic algorithm could be described as follows: a list of chromosomes would be generated as the initial population, it would then start to evolve. Firstly, two chromosomes would be randomly selected as parents, and after crossover and mutation, a new chromosome would be generated as a new child chromosome, it would be added to the new generation population. Crossover here, also known as recombination, is a genetic operator that is used to combine the genetic information of two parents. Mutation here means that in order to maintain the diversity of a new population, one single individual may have the chance to change the chromosome following certain patterns. This process would continue until the next generation population has the same number of chromosomes as the initial one. When one of the three termination conditions is satisfied, the evolution would be terminated, and the best solution of the last generation would be selected as the output of the genetic algorithm.

D.2. Fitness Evaluation. For Fitness Evaluation, given the chromosome's finish time T, regulated time R_t, overtime penalty $O_penalty$, and the sum of all assigned postmen's initial costs for the chromosome C_p, the fitness for one single individual could be calculated as:

$$Fitness(chromosome) = \frac{1}{(T - R_t) * O_{penalty} + C_p}$$

D.3. Crossover and Mutation. For crossover and mutation methods, we both implemented four different methods. As for crossover, single point, two points, multiple points, and uniform crossover are implemented.

- **Single Point**: A point is picked on one of the parents randomly, assignments on the right-hand side of the point would be swapped, and the new assignment would become the new chromosome for the next generation. These four methods could be illustrated in Figure1.
- **Two Points**: Two points are picked randomly picked on the parent, the chromosome information between these two points would then be exchanged to get the new individual.
- **Multiple Points**: Multiple points are picked randomly picked on the parent, and the chromosome information on these multiple points is exchanged between two parents to produce a new child.
- **Uniform**: Multiple points are picked randomly picked on the parent, and the chromosome information on these multiple points is exchanged between two parents to produce a new child.

Based on the chromosome representation we defined, we picked four mutation operators shown below. (6)

- **Insert Mutation**: Randomly select a gene from a chromosome and insert it at a random position within the same chromosome.
- **Inversion Mutation**: Randomly select a subset of genes (also called a segment) from a chromosome and reverse their order.
- **Swap Mutation**: Randomly select two positions in a chromosome, swap the values of the gene at those two positions.
- **Scramble Mutation**: Randomly select a subset of genes from a chromosome and then scramble their order.

D.4. Termination Conditions. Termination conditions determine when the evolution would come to an end. We have set three different termination conditions, when one of them has been satisfied, the evolution would stop and the solution would be given.

- **Converge Generation Number**: When the evolution's best solution has no change for a certain number, we would say that the current evolution is converged. Therefore the algorithm would be stopped and produce the output.
- **Get solution with finish time less than regulated time**: When the genetic algorithm actually finds out the feasible solution that took less time to deliver all packages than the regulated time, it would terminate the evolution and return the result.
- **Maximum Number of Generations**: When the step number of evolution reaches a certain number, the evolution would stop and return the solution.

Experiments and Results

To obtain the best assignment by genetic algorithm, we experimented with all combinations of crossover methods and mutation operators introduced above. Before the experiments started, we imported eight graphs from SNDlib, a library of

test instances for Survivable fixed telecommunication Network Design, and converted those graphs into the specific Graph class mentioned in the Methodology section. For each graph, We then randomly generated a demand list that has 3 times the size of the graph, and a postmen list that has one-fifth the size of the graph. We keep graphs, demands, and postmen data constant to guarantee results are only impacted by rates we tuned. Detailed constant parameters used throughout experiments are listed in Table 1.

Parameters	Value
Maximum Number of Generations	1000
Population Size	500
Mutation Rate	0.05
Converge Generation Number	20
Terminal Fitness	0.05
Regulated Time	30
Overtime Penalty	1.5

Table 1. Parameters throughout experiments (Might Changed in the specific experiment).

E. Get the Best Uniform Crossover Rate. Firstly, we test the influence of the uniform crossover rate and get the best value for it. During this experiment, we used the swap mutation method and tuned the uniform crossover rate from 0.1 to 0.5. Based on the result shown in Figure 2 and Table 2, we set a uniform crossover rate equal to 0.3 for the rest of experiments.

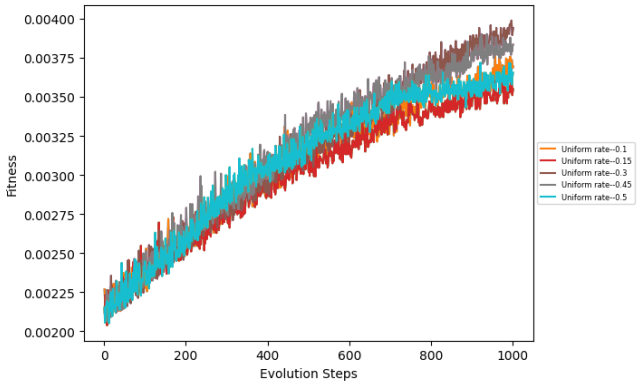

Fig. 2. illustration for four different crossover methods.

Uniform Crossover Rate	Fitness
0.3	0.00394
0.45	0.00384
0.1	0.00370
0.5	0.00366
0.15	0.00355

Table 2. Fitness of Uniform crossover rate in Descending Order.

F. Try All Combinations. In the second experiment, we tried all 16 combinations of crossover methods and mutation operators in one graph to get the best combinations. The result in Figure 3 and Table 3 indicates that the best mutation operator would be swap . But for crossover methods, we could not get the best one. Therefore, we decided to use the best three combinations in the final experiment.

Crossover Method	Mutation Operator	Fitness
two_points	swap	0.00384
single_point	swap	0.00373
multi_points	swap	0.00359
uniform	swap	0.00358
single_point	scramble	0.00354
single_point	inversion	0.00281
two_points	scramble	0.00279
two_points	inversion	0.00272
single_point	insert	0.00269
two_points	insert	0.00258
uniform	scramble	0.00247
uniform	inversion	0.00237
multi_points	scramble	0.00232
multi_points	inversion	0.00226
uniform	insert	0.00209
multi_points	insert	0.00208

Table 3. Fitness of all combinations in descending order.

G. Get the best Mutation Rate. Similarly, in this experiment, we tried to obtain the best mutation rate within in [0.01, 0.02, 0.05, 0.1, 0.2]. The combination we used is a two-point mutation and swap crossover operator. As Figure 4 shows, when the mutation rate equals 0.01, the process has the fastest convergence speed and highest fitness.

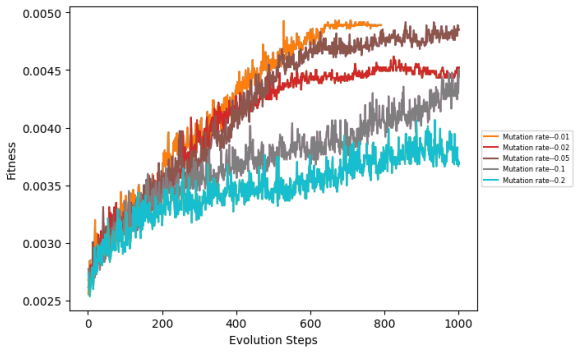

Fig. 4. illustration for four different crossover methods.

H. Test Best-Three combinations. Due to the NP-Hard nature of the problem we are studying and the size of the dataset, it is not possible to obtain an optimal solution. Also, the problem we defined is not the same as the initial problem in SNDlib. Hence, we have no benchmark to evaluate our model's efficiency in complex problems. But for a randomly-generated small scale problem that has 5 nodes, 4 postmen, and 11 demands, the comparison between the solution obtained by the brute-force method and our heuristic method with the parameter above indicates that the latter can obtain optimal solution faster. And as the scale grows up, a brute-force method that has $\mathcal{O}(p^d)$ time complexity, where p is the number of postmen, and d is the number of demands, would perform worse. For larger scale problems, though we sac-

rificed the optimality of the solution for computational efficiency, and may produce a solution that is not guaranteed to be optimal, it is often sufficient for practical purposes.

Method	Finish Time(second)	Fitness
Exhaustive	31.5	0.04273
Genetic Algorithm	18	0.04273

Table 4. Fitness of all combinations in descending order.

After proving the feasibility of our method, we then tested Best-Three combinations on all eight graphs, with the mutation rate coming out from the last experiment. Figure 5 illustrates that all three combinations can reach a similar suboptimal solution within 1000 generations. It also indicates that single-point crossover has better performance in terms of convergence speed and fitness.

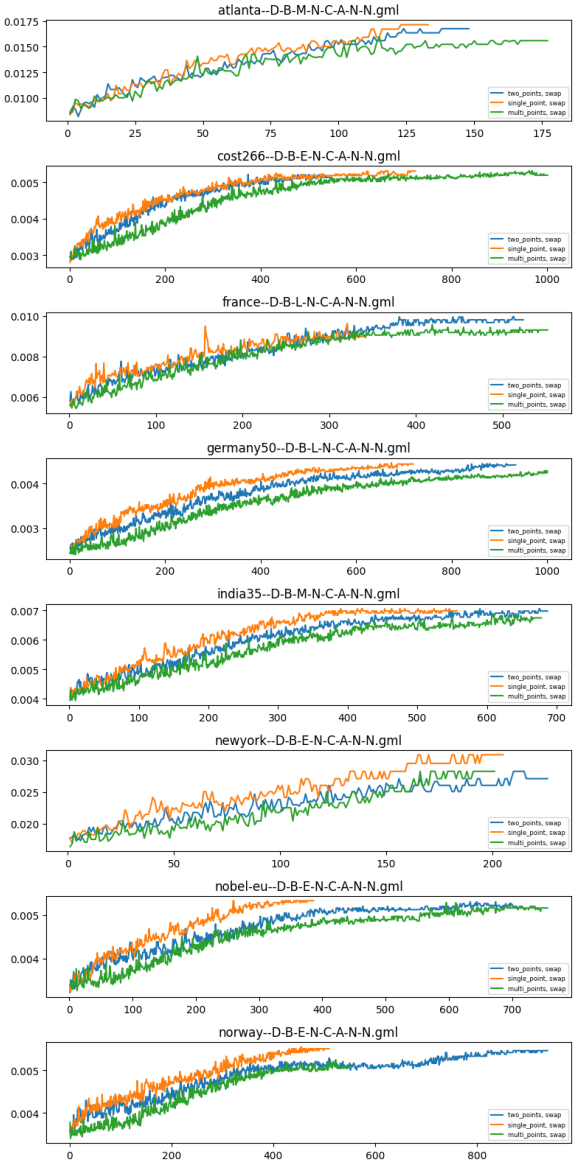

Fig. 5. illustration for four different crossover methods.

Conclusions

In this project report, we proposed a new prioritized package delivery problem (PPDP) that considers the priority of packages in addition to the distance and cost of each postman. We developed a generic algorithm to optimize the delivery routes and schedules. Our computational experiments on various instances showed that the proposed algorithm can solve the PPDP in small and large-scale problems.

The proposed PPDP and algorithm have several practical applications in the package delivery industry, such as e-commerce, logistics, and transportation. By considering the priority of packages, the proposed algorithm can improve delivery efficiency, reduce delivery time, and enhance customer satisfaction. Moreover, the proposed algorithm is generic and can be easily adapted to different real-world scenarios by adjusting the parameters and constraints.

In conclusion, the proposed PPDP and algorithm provide a new perspective on the package delivery problem and contribute to the optimization of package delivery services. Future research directions may include investigating the integration of real-time traffic and weather information into the PPDP and exploring the application of machine learning and reinforcement learning techniques to further improve the solution quality and efficiency.

Bibliography

1. Anuj Kumar. Improved genetic algorithm to solve small scale travelling salesman problem. In *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 516–520, 2020. doi: 10.1109/ICICCS48265.2020.9120880.
2. Edwin Dwi Ahmad, Ahmad Muklason, and Ika Nurkasanah. Route optimization of airplane travel plans using the tabu-simulated annealing algorithm to solve the traveling salesman challenge 2.0. In *2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*, pages 217–221, 2020. doi: 10.1109/CENIM51130.2020.9297892.
3. Gulsun Nakiboglu and Pinar Elcicek Gunes. Vehicle routing problem in pharmaceuticals distribution and genetic algorithm application. In *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, pages 1–6, 2018. doi: 10.1109/IDAP.2018.8620875.
4. Wu Qianqian, Li Bin, and Wu Qianqian. Improved genetic algorithm based express delivery route optimization model. In *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, pages 725–729, 2020. doi: 10.1109/ICAICA50127.2020.9182464.
5. Li Wenzheng, Liu Junjun, and Yao Shunli. An improved dijkstra's algorithm for shortest path planning on 2d grid maps. In *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pages 438–441, 2019. doi: 10.1109/ICEIEC.2019.8784487.
6. Nitasha Soni and Tapas Kumar. Study of various mutation operators in genetic algorithms. 2014.

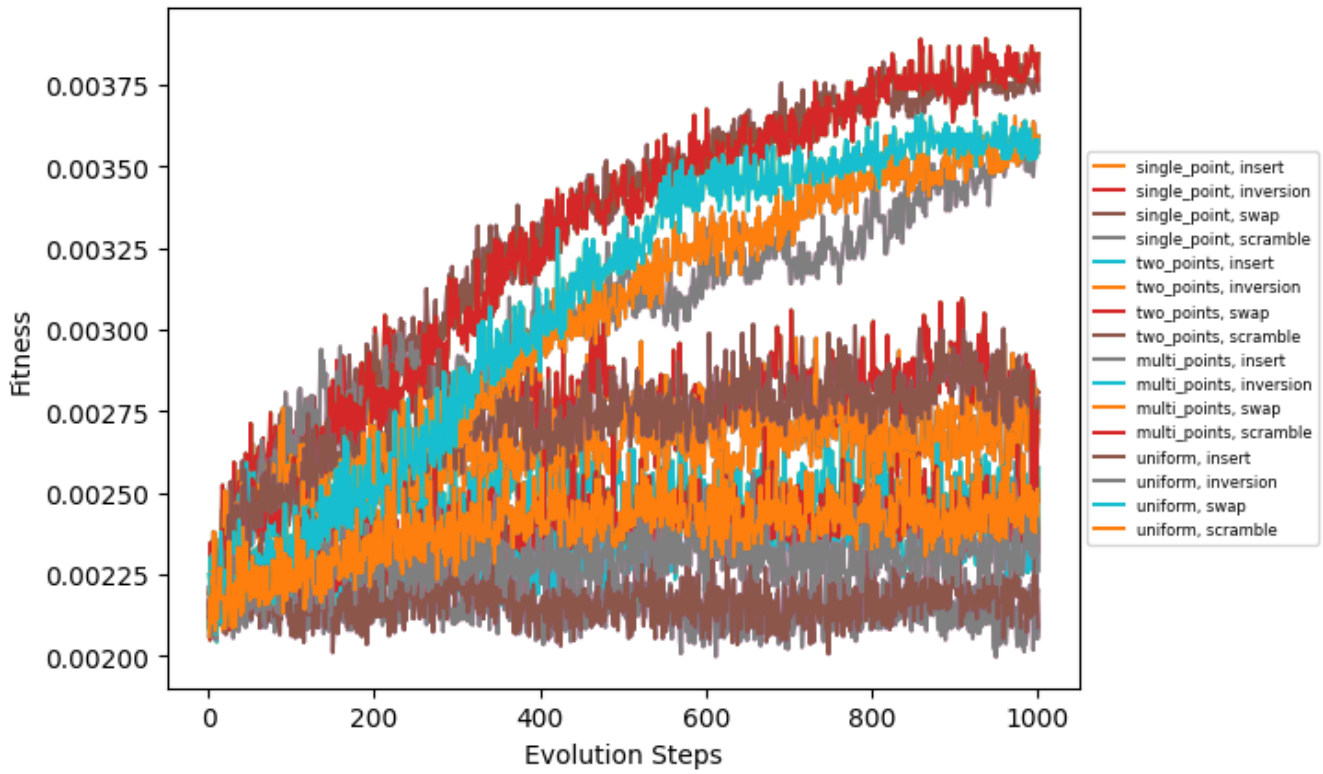

Fig. 3. illustration for four different crossover methods.