Assignment1
Travelling Salesperson Problem

Xiyao Huang
Department of Computer Science
St. Francis Xavier University
Antigonish, Canada
x2023fkv@stfx.ca
202306631

I. Introduction

The problem to be solved in this assignment is: how to develop and improve an effective GA algorithm that can find the approximate optimal solution of TSP for larger data sets within a reasonable time limit. Because the complexity of TSP grows exponentially, traditional methods may not scale well, so using algorithms such as GA may be more applicable. The difficulty lies in how to effectively solve TSP while improving the performance of GA through various enhancements and modifications. And in order to intuitively see the solution, it is necessary to design a reasonable visualization. After implementing an effective TSP algorithm based on GA, we will go deeper to solve a generalized problem of TSP, namely VRP. The difficulty is that although the two problems are similar, the GA algorithm applied to VRP cannot be achieved by simply modifying the parameters, but almost a completely new algorithm must be developed. The first O(2n)-time dynamic programming algorithms for TSP date back to the early 1960s. However, in the next half century, no one was able to break the 2 barrier in terms of running time. [1] Many scientists have made a lot of efforts in this regard. For example, David Eppstein once designed an algorithm that finds Hamiltonian circuits or minimum-weight Hamiltonian circuits in graphs with a degree of at most 3 in time complexity $O(2^{n/3})$, and counts or lists all Hamiltonian circuits in time $O(2^{3n/8})$. [2] Dominic J. Moylett and his team also demonstrated a quadratic quantum speedup when the degree of each vertex is at most 3 by applying a quantum backtracking algorithm to the classical algorithm of Xiao and Nagamochi and used a similar technique to speed up the classical algorithm when the degree of each vertex is at most 4, and then accelerated it for graphs of higher degrees by simplifying these instances in 2017. [3]

II. Problem description

A. Travelling Salesperson Problem

TSP (Traveling Salesperson Problem) is one of the most famous and basic problems in combinatorial optimization, which was first proposed in 1930. The distance between cities satisfies the triangle inequality. [4] The goal of the problem is: given a list of cities and the distance between each pair of cities, find the shortest possible route that visits each city once and returns to the origin city. GA, as a search algorithm, can effectively solve complex combinatorial optimization problems like TSP through genetic variation and crossover operations of the population, explore the solution space of TSP, and gradually optimize the path..

B. Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is one of the three generalizations of TSP. The capacitated vehicle routing problem is one of the most studied combinatorial optimization problems in operations research. [5] In short, VRP is a generalization of the TSP with multiple objectives and multiple values to be optimized. Its purpose is to find the optimal route for a fleet of vehicles to deliver goods to a given group of customers.

III. Algorithm Description

A. Trevelling Salesperson Problem (TSP)

1) Implementation:

- Initialization: In the genetic algorithm, a bunch of parameters are first initialized, such as quantity, population size, number of iterations, and city location information. These parameters are used to control the operation of the genetic algorithm. By using the greedy algorithm to generate the initial population, this method generates a path by selecting the nearest city, thereby forming a better initial solution. It is more efficient than random generation.
- Fitness Evaluation: The fitness evaluation is performed at the beginning of the function, calling the function of calculating the current population to calculate the fitness of self.fruits. The higher the fitness, the better the solution of the individual.
- Selection: The selection operation is divided into two steps and implemented by two methods. First, a certain proportion of parent individuals are selected by fitness, then two individuals are selected from the parent generation for crossover, and two random numbers are generated by selecting weights to select two parent individuals.
- Mutation: The mutation operation is implemented by writing a mutation function, which performs a local reverse transformation on the genes of an individual to maintain the diversity of the population. In this function,

two positions are randomly selected to define a subsequence, and the subsequence is reversed, and then the reversed subsequence is replaced back to the original gene sequence.

- Cross: The crossover operation is implemented by the cross function, which is used to exchange genes between two individuals and generate two new offspring. This function randomly selects a gene segment and exchanges it between the two parent individuals. At the same time, it adjusts the conflicting parts that may appear after the exchange.
- Termination: The termination condition is that during the operation of the genetic algorithm, when the set number of iterations is met, the algorithm will stop searching. After termination, it will check whether the length of the found path is the current optimal path, and then update it.
- Visualization: The visualization part is implemented using the matplotlib library, showing two graphs: the first graph shows the optimal path, and the second graph shows the optimal solution convergence curve of the genetic algorithm

2) Enhancements:

- Elitism: The idea of Elitism is used in the design of GA. Firstly, the fitness of all individuals in the current population is calculated. Secondly, an appropriate proportion of parent individuals is selected, and the best performing individual in the parent generation and its fitness are stored and retained through Elitism. Thirdly, new individuals are generated through crossover and mutation and added to the population. Fourthly, check whether the best individual retained is in the new population. If not, add it directly to the new generation. Finally, the population is updated and the best individual and its fitness are returned.
- Multi-point crossover: The original two-point crossover is changed to a multi-point crossover. This multi-point crossover can increase the diversity of the solution space, allowing the algorithm to explore more potential solutions, thus potentially improving the ability of global optimization.
- Greedy Algorithm: During initialization, a greedy algorithm is used instead of a common random algorithm. The core idea of the greedy algorithm is: start from a certain starting city, and then select the current city to the nearest unvisited city each time until a complete path is built. The advantage is that it can speed up the construction of the optimal solution and reduce the search space.

B. Vehicle Routing Problem (VRP)

- Initialization: Parameters such as vehicle capacity and node information are initialized, and the population is generated by randomly arranging nodes.
- Fitness evaluation: The fitness of the current population is evaluated by calculating the total distance of the vehicle route.
- Selection: Based on fitness, two parent individuals are selected for crossover operation by random extraction and comparison.
- Mutation: Two positions in the individual are randomly selected for exchange, and mutation operation is performed to maintain population diversity.
- Crossover: Gene segments are selected between two parent individuals for exchange to generate new offspring, and correction is performed to prevent duplication and exceeding vehicle capacity.
- Termination: When the set number of iterations is reached, the algorithm terminates and outputs the optimal route and cost.

IV. Results

Because there are a lot of test data, the results are displayed using bier127 as an example. First, the results are shown in two parts: 1. Optimal path: shows the shortest path found after the GA runs. Cities are represented by coordinate points and connected by broken lines to form a path diagram. 2. Convergence curve: shows how the fitness changes with the increase of generations. The downward trend of the curve indicates that the fitness is constantly improving, that is, the path length is decreasing. After each iteration, the best solution in the population is recorded and compared with the best solutions in other iterations. These solutions will gradually converge to a better solution. By observing the changes in these path lengths, you can understand the convergence process of the algorithm. You can compare the optimal path lengths found under different parameter configurations, as well as their convergence speed and stability.

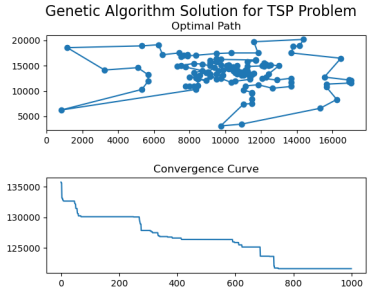

Fig. 1. Result

A. Two-point crossover vs. Multi-point crossover

This time, I used multi-point crossover instead of two-point crossover. Compare the two pictures below: Both generate a reasonable optimal path. The convergence speed of the two is not much different, but it is obvious that there is still a significant drop near 500 on the right, which means that multi-point crossover has more room for exploration and there is still room for improvement after 500 iterations. Therefore, I increased the number of iterations to 1000 in the results, which can further avoid local optimization and converge faster in

the early stage. In terms of potential, multi-point crossover is greater than two-point crossover.

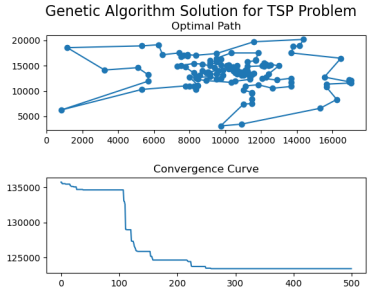

Fig. 2. bier127-2cross

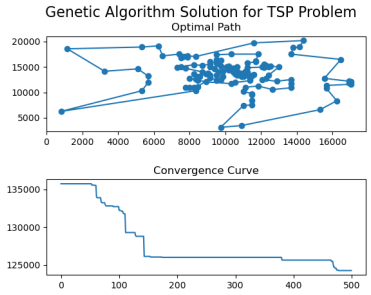

Fig. 3. bier127-multi-cross

B. Greedy Algorithm vs Random Initialization

In this GA, I used the greedy algorithm instead of random initialization. Here is a comparison between the two:

Greedy Alogrithm	Random
High-quality initial solutions	random, may very poor
Reduces the search space, finds better solutions faster	Larger search space, but quality varies
Low diversity in initial solutions, solutions are similar	High diversity, useful for exploring global optima
Quickly finds better solutions	Low efficiency, solutions may be very poor

Fig. 4. Greedy vs. Random

After implementing the TSP problem, I further explored the VRP problem. The VRP solving code you provided uses a genetic algorithm (GA) to find an efficient route for vehicles to visit all required nodes (cities) while taking into account capacity constraints. The results of this algorithm usually show a near-optimal or reasonable route, depending on parameters such as population size, number of iterations, and mutation rate. However, for time reasons, this code only gives a rough visual of the VRP, and the routes are not distinguished in the visualization, so it is impossible to tell when to return to the depot. The following is the result graph (The test data is a self-written file):

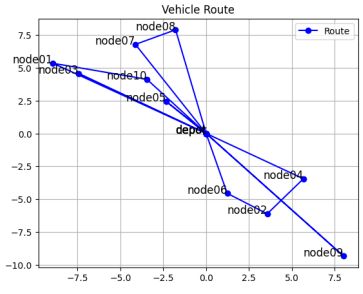

Fig. 5. VRP Result

V. Discussion

In this solution of the Traveling Salesman Problem (TSP), a genetic algorithm (GA) was used. The initial population was initialized by a greedy algorithm, and Elitism and multi-point crossover operations were used for optimization. The greedy algorithm helped generate relatively high-quality initialization. With continuous iterations, the optimal path was explored and optimized through crossover, mutation, and selection. Elitism and multi-point crossover were then optimized to make the search more accurate and extensive. The results show that the path length continues to improve with the increase in the number of iterations. However, after a certain number of iterations, the convergence rate slowed down significantly, indicating that the genetic algorithm may be close to the local optimal solution.

Compared with the random initialization method, the greedy algorithm has a significant advantage in generating high-quality initial solutions. Although the random initialization method can generate more diverse solutions, it also generates solutions of uncertain quality, which requires more iterations for the algorithm to converge to a better path. The greedy algorithm can make the algorithm converge faster. Compared with other optimization algorithms, the genetic algorithm has achieved a good balance between exploration and utilization. The result achieved by this genetic algorithm is competitive with the best known solution to the TSP problem, but it obviously does not reach the level of the best known solution. The reason for this gap may be that the genetic algorithm is prone to fall into local optimality during the iteration process as the population converges.

As a generalization of TSP, the VRP algorithm is much more difficult to implement than TSP. Unlike the expected simple modification of a few parameters or addition of a few functions, completing the VRP algorithm almost requires the implementation of a completely new algorithm. Therefore, this code only roughly describes VRP, and there is still a lot of room for improvement.

VI. Conclusion

The main benefit of this project is that I learned how to use genetic algorithms (GA) to effectively solve complex combinatorial optimization problems such as the Traveling

Salesman Problem (TSP) and its generalized form, the Vehicle Routing Problem (VRP). The performance of the genetic algorithm was significantly improved by introducing Elitism, multi-point crossover, and greedy algorithms. Although the genetic algorithm can stably find a near-optimal solution in the TSP problem, the complexity is greatly increased when dealing with more complex VRP problems, and there is no perfect solution. For the VRP problem, preliminary implementations show that the genetic algorithm can cope with the complexity of multiple vehicles and capacity constraints. However, due to time constraints and the algorithm's imperfect handling of vehicle routing back to the warehouse, the results of the VRP are relatively basic and there is room for further improvement.

REFERENCES

[1] Xiao, Mingyu, and Hiroshi Nagamochi. "An Exact Algorithm for TSP in Degree-3 Graphs Via Circuit Procedure and Amortization on Connectivity Structure." Algorithmica 74, no. 2 (2016): 713–41. https://doi.org/10.1007/s00453-015-9970-4.

[2] Eppstein, David. "The Traveling Salesman Problem for Cubic Graphs." Journal of Graph Algorithms and Applications 11, no. 1 (2007): 61–81. https://doi.org/10.7155/jgaa.00137.

[3] Moylett, Dominic J, Noah Linden, and Ashley Montanaro. "Quantum Speedup of the Traveling-Salesman Problem for Bounded-Degree Graphs." Physical Review. A 95, no. 3 (2017). https://doi.org/10.1103/PhysRevA.95.032323.

[4] Karpinski, Marek, Michael Lampis, and Richard Schmied. "New Inapproximability Bounds for TSP." *Journal of Computer and System Sciences* 81, no. 8 (2015): 1665–77. https://doi.org/10.1016/j.jcss.2015.06.003.

[5] Arnold, Florian, and Kenneth Sörensen. "What Makes a VRP Solution Good? The Generation of Problem-Specific Knowledge for Heuristics." *Computers & Operations Research* 106 (2019): 280–88. https://doi.org/10.1016/j.cor.2018.02.007.