Solving NTHU Bus Problem Using Genetic Algorithm to

*108062226 蘇志翔*

*Abstract*—In this proposal, I will talk about a kind of application of evolution algorithm, vehicle routing problem. First, I will start from my motivation. Then, I will introduce the definition of vehicle routing problem, and show the reason to use genetic algorithm to solve the vehicle routing problems. Finally, I will temporary

# Introduction

In NTHU, there are many school buses that pick students up to Humanities and Social Sciences Building (HSS) or TSMC Building (TSMC), vice versa. During peak hour, the buses come every five minutes.

For my personal experience, since buses come every five minutes, and most of the students go to the station 10 to 15 minutes before the class starts, it always happens that some of the students cannot arrive at the class on time due to limited capacity of the bus. Besides making students go to station earlier, increase the capacity of the bus or decrease the average time the buses pick up students are possible way to deal with this problem. Then, how much capacity should the bus increase or how short should the average time picking up students can be evaluated by vehicle routing problem (VRP), which will be talked about later.

# Vehicle routing problem definition

Vehicle routing problem (VRP) is a problem which was first introduced by George Dantzig and John Ramser in 1959. Its motivation is to find out what is the optimal set of routes for a fleet of vehicles to traverse in order to deliver to a given set of customers.

The description of VRP will be: given a depot with M vehicles (v1, v2, … , vM), each vehicle vi has capacity Qi. There are N customers (c1, c2 … cn), each has demand Di. To emulate VRP, there must have costs or time spent between a vehicle vi and a customer cj. What we need to do is to use these vehicles to satisfy all customers’ requirements. According to the constraints, we may skip some customers’ requirements to reach the optimal solution. Fig. 1 is a solution to a VRP.

VRP can be apply on multiple situation. For example, 1) bus or airline scheduling. 2) Letters or newspaper transmission. 3) Product transportation, and so on. According to its application, the goal of VRP can be 1) to minimize the cost of the total transportation, 2) to minimize the number of vehicles used to satisfy the needs of customers, 3) to maximize the profit.Generally, There are three types of VRP: 1) VRP with one starting point and one end point, both of which are identical. 2) VRP with one starting point and one end point, both of which are different. 3) VRP with multiple starting points and end points.

Fig. 1 A kind to solution to VRP.

# Reasons to use genetic algorithm

As the description, the travelling salesman problem (TSP) is a variation of VRP by allowing to use only one vehicle (the salesman), make the demands of all customers to be zero and the capacity to be ∞. Because TSP is a NP-hard problem and TSP can convert into VRP in polynomial time, VRP is also a NP-hard problem. Since VRP is a NP-hard problem, if using brute-force method or other deterministic algorithm, it will take exponential time to do so. Thus, using some heuristic algorithms to get near-optimal solution is preferable.

There are many kinds of number-optimization algorithm in evolution algorithm (EA) such as genetic algorithm (GA), evolution strategy (ES). In this term project, I’ll use GA to solve VRP. GA is an algorithm used in optimization, and since there is a permutation representation, I can use permutation to represent a solution to VRP because it is based on a graph. Then, by defining the fitness and constraints, I think it’s possible to solve VRP by permutation-based GA.

# Problem description

To describe the “NTHU Bus Problem”, I first assume that there are 10 stations s1, s2, …, s10, which represent 1) North gate (Start), 2) GEN II (To south gate), 3) Maple trail (To south gate), 4) HSS, 5) TSMC, 6) South gate parking lot, 7) Go garden parking lot, 8) Maple trail (To north gate), 9) GEN II (To north gate), 10) North gate (End). The start is s1, and the end is s10. Note that for NTHU school buses, there are two routes: red route and green route, which has different order of stations. In NTHU Bus Problem, I assume that all buses follow the red route, which is of the order s1, s2, …, s10.

For each station si, there are di students that want to go to that station, which is the demand of that station. As for the vehicles in NTHU Bus Problem, we defined b1, b2…bk to represent bus 1, bus 2…bus k, where k is the number of buses. All the buses have the same capacity. Capacity may be be unlimited or limited according to its GA type, which will talk about later.

Since s1 is the start, the demand of s1 should be 0. The demands of other stations come from three parts: 1) students that have classes at 10:10 am, 2) students that end classes at 10:10 am and want to go home, and 3) personal experience. Stations having courses like GEN II (To south gate), HSS, TSMC use the demand from part 1. Those that nears to the north gate and the dormitory follow the demand of part 2, such as GEN II (To north gate), maple trail (To north gate) and North gate (End), and others follow part 3.

For part 1 demand, I use the data from the curriculum of NTHU [1]. I assume 1) the semester to be 2023-spring, 2) the day to be Monday, and 3) the buses start at 9:50 am, which is the end of the second class. Then, I assume the demands are half of the number to the students that have course for next class, which starts at 10:10 am. The relationship between the demands and the number to the students that have course for next class follows the equation below:

(1)

By following the equation, we can obtain d2 = 10, d4 = 250 and d5 = 91, which are the demand of s2, s4 and s5.

Next is the part 2 demand. I also use the data from the curriculum of NTHU and the assumption from part 1 demand. Then, I assume the demands are also half of the number to the students that have course for next class, which starts at 10:10 am. The reason to set the proportion to be half is because the total number of students lived in dormitories is 8300 [2], and the total number of students in NTHU is 18122 [3]. To be convenience, I take floor to the proportion and make the final proportion be 2. The relationship between the demands and the number to the students that end classes at 9:50 am follows the equation below:

(2)

By following the equation, we can obtain d8= 20, d9 = 104 and d10 = 50, which are the demand of s8, s9 and s10.

Finally, for the remaining stations, I take the bus on 5/29 and 6/5, Monday, at 9:50. I calculate the average number of the students that goes to the remaining stations, including s3, s6, s7, and multiply it with 4, since the average number of buses between 9:50 and 10:10 is 4. Thus, we obtain d3 = 23, d6 = 17, and d7 = 2.

Next, I determine the distance between each station by using Google maps. As I mention above, the route of the school bus is unidirectional, which means s1 should only connect to s2, s2 should only connect to s3, and goes on. To make it unidirectional, I make the distance from s2 to s1 be infinite. Thus, the graph of the NTHU Bus Problem is complete, as Fig. 2 shows.

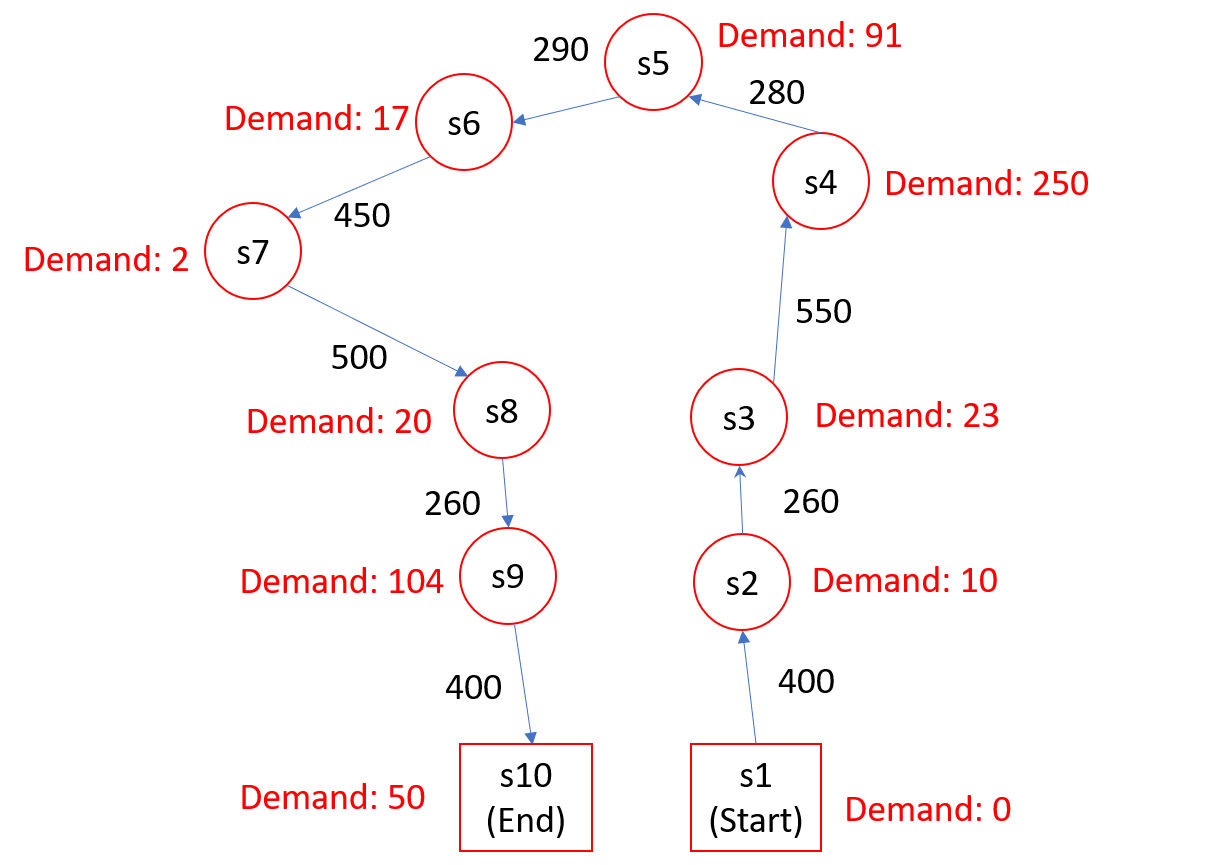


Fig. 2 The graph of the NTHU Bus Problem.

# Problems found after the presentation and its solution

Originally, I try to solve NTHU Bus Problem by using order-based GA and edge-based crossover and mutation. However, after the presentation, teacher Ting figured out some problems, such as every nodes in the graph should only be passed once during the execution, and taking time constraint into consideration might be able to visit the same node several times. So, what I want to do is to find out a solution with teacher’s advices.

I’ve found the discussion about the possibility to visit the same node multiple times during one evolution [4]. The conclusion is that visiting the same node multiple times is not possible, since it should satisfy the basic definition of Hamilton Cycle that it should pass every nodes exactly once, which is the key concept of VRP and TSP.

As an alternative, one solution is to split the same nodes into several nodes on the same graph. However, if using this method, since the nodes will always go to the lowest cost nodes, which is also the nearest adjacent node, there’s nothing changed.

Another solution is to divide the original graph into several graphs [5]. After the division, each graph represents the route of each vehicle. We divide nodes of the graph into 2 types: 1) nodes that want to be visited several times and 2) node that will only visit once.

Suppose there is a node called node 1 that will be visited for twice. Then, node 1 will appear at graph A (node 1A), which is the route of vehicle A, graph B (node 1B), which is the route of vehicle B.

For type 1 nodes, these nodes exist in all graphs after the division. For the demand of node 1A, node 1B, we need to specify a way to divide the demand of original node 1 into the demand of node 1A and the demand of node 1B. As for type 2 nodes, we need to decide which graph it should occurs. Since the graph after the division means the route of the corresponding, which vehicle will go through this node will be important. Fig. 3 shows the idea of the division of the graph to deal with the multi-visits of the same node.

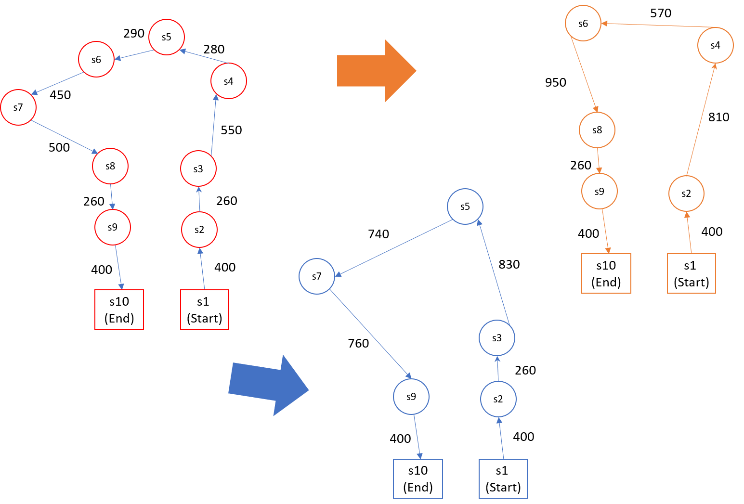


Fig. 3 The division of the graph to deal with the multi-visits of the same node

Since in the NTHU Bus Problem, every station will be multi-visited, what I divide the demand of the original nodes has two ways: 1) Randomly choose an integer representing the students sent by the bus, and eliminate it with the fitness function. 2) Randomly choose the an integer, and make sure that for every node i, it follows the equation below:

(3)

Both of them will have different effect, and will be discussed later.

# Fitness function

First, we have to decide the fitness function. The fitness function can be expressed by the equation below:

(4)

The fitness function tries to minimize the time spent and take oversupply and undersupply into consideration. Since the goal is to meet the deadline of the next class at 10:10 am, the individual spending less time will get higher scores. First, we can divide the time spent into two parts:

(5)

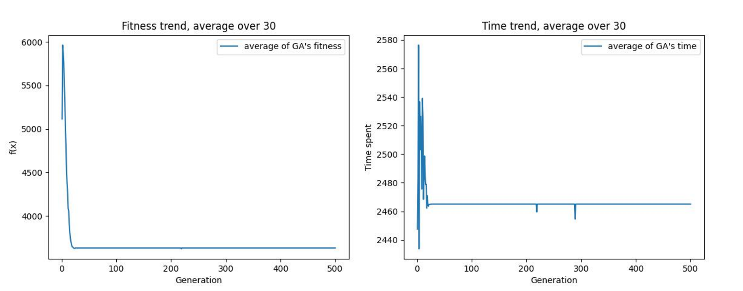


Fig. 4 The lowest fitness might not have the lowest time spent due to the weights of penalty to oversupply and undersupply

Since for every bus, its starting point and end point are both s1 and s10, so the driving time for all buses are identical. Thus the one which might vary for all buses is the unload time. The equation of unload time for bus i is below:

(6)

If the demand of the station within the bus is not zero, then this bus stops at that station. In the equation above, I take two factors into consideration: 1) Parking time, 2) get off time at station j. The parking time is a constant, which means every time the bus stop at a station, it spends a constant time to stop and restart the bus; as for the get off time, it is determined by the students sent by the to .

Next is to decide “demand not satisfied” in equation (4), which equals to . As I mentioned above, there are two ways to divide the original demands into several demands for each buses. Thus, in fitness function, if the station doesn’t satisfy its demand, then the individual should be punish. Both oversupply and undersupply should add penalties. In NTHU Bus Problem, we denote

Last but not least, we need to justify the weight. Since the total demand of the NTHU Bus Problem is 567, if every student spends 4 seconds to get off the bus, then in the worst case, a bus will spend 2268 seconds. If we don’t set the weights properly, we might get the result that the individual with the lowest fitness won’t have the lowest time spent. Fig.4 shows that an individual with lowest fitness might not have the lowest time spent, and table I shows the results of the global minimum, including the fitness and the time spent. In table I, I set the ratio between non-satisfied penalty and time weight to be 100 and cause the demands of stations are not satisfied strictly.

1. Results of the global mimimum when setting the weights non-properly(time weight : non-satisfied penalty = 1:100)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Weight** | | | | | 3333.75 | | | **Time spent** | | | | 2133.75s | | |
| **d1** | **d2** | | **d3** | | **d4** | | **d5** | **d6** | **d7** | **d8** | | **d9** | | **d10** |
| 0 | 14 | | 23 | | 246 | | 90 | 17 | 2 | 20 | | 105 | | 52 |
|  | | | | | | | | | | | | | | |
| 0 | | 4 | | 0 | | -4 | 1 | 0 | 0 | | 0 | | 1 | 2 |

As table I show, the demands of each station aren’t satisfied strictly. So how strictly the NTHU Bus Problem should follow decides the weight balance between time weight, oversupply penalty and undersupply penalty. I set the time weight to be 1, the oversupply penalty and undersupply penalty to be 1000. In this case

# Ga designments

To solve the NTHU Bus problem, I design 3 types of GA: 1) GA with unlimited-capacity bus and buses can starts simultaneously. 2) GA with unlimited-capacity, but buses cannot starts simultaneously. Instead, a bus should wait for a period of time after the previous one sets off. 3) GA with limited-capacity bus.

The reason why I divide unlimited-capacity bus GA into

As for GA design, I will use order-based GA with generational population model. All strategies of crossover and mutation will be tried to find out which has the highest solution quality. As for the selection operators, I will use tournament selection for both parent selection and survivor selection for convenience. If there is enough time, I’ll spend more time to expend this question, such as making the capacity of every bus different or make every station has students that want to arrive that station and students that want to leave. And that’s all for my proposal.

1. NTHU Curriculum (<https://www.ccxp.nthu.edu.tw/ccxp/INQUIRE/JH/6/6.2/6.2.9/JH629001.php>)
2. NTHU Dormitory Introduction (<https://sthousing.site.nthu.edu.tw/p/412-1254-3417.php?Lang=zh-tw>)
3. NTHU charts (<https://www.nthu.edu.tw/about/chart/1>)
4. PierreHamoir, “Multiple Visits of same node VRP #1246,”, May 2019 (<https://github.com/google/or-tools/issues/1246>)
5. shinkisan, March 2018 (<https://github.com/google/or-tools/issues/630>)