Solving NTHU Bus Problem Using Genetic Algorithm to

108062226 蘇志翔

***Abstract*—In this term paper, I will introduce what is NTHU Bus Problem, and talk about the way to solve it and the solution. 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. Next, I will introduce the NTHU Bus Problem, and describe how I collect the dataset been used. After that, I will talk about the problem figured out by teacher Ting during the presentation and the method to solve it. Then, the fitness function and GA designment will be introduces. Last but not least, result of this term project and the conclusion will be told in the end of the term project.**

I. 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.

II. 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 (v_1, v_2, … , v_M), each vehicle v_i has capacity Q_i. There are N customers (c_1, c_2 … c_n), each has demand D_i. To emulate VRP, there must have costs or time spent between a vehicle v_i and a customer c_j. 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.

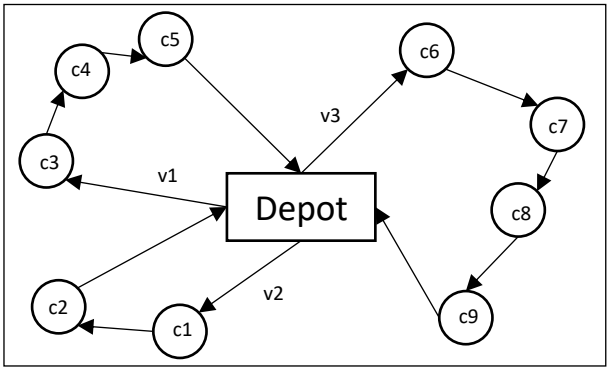

Fig. 1 A kind to solution to VRP.

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.

III. 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.

IV. Problem description

A. Introduction of NTHU Bus Problem

To describe the "NTHU Bus Problem" (NBP), I first assume that there are 10 stations s_1, s_2, …, s_{10}, 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 s_1, and the end is s_{10}. Note that for NTHU school buses, there are two routes: red route and green route, which has different order of stations. In NBP, I assume that all buses follow the red route, which is of the order s_1, s_2, …, s_{10}.

For each station s_i, there are d_i students that want to go to that station, which is the demand of that station. As for the vehicles in NBP, we defined b_1, b_2…b_k to represent bus 1, bus 2…bus k, where k is the number of buses. All the buses have the same capacity, and each bus b_i has its route r_i to send students. In each r_i, the student sent to station s_j by b_i will be d_{ij}. Capacity may be be unlimited or limited according to its GA type, which will talk about later.

B. *Demand calculation*

Since s_1 is the start, the demand of s_1 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:

$$demand = \frac{1}{2}\sum_{i:\ the\ classes\ in\ that\ building} N_i \qquad (1)$$

N: the number of students for the class starting at 10:10 am

By following the equation, we can obtain $d_2 = 10$, $d_4 = 250$ and $d_5 = 91$, which are the demand of s_2, s_4 and s_5.

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:

$$demand = \frac{1}{2}\sum_{i\in \text{the classes in that building}} n_i \qquad (2)$$

n: the number of students that end class at 9:50 am

By following the equation, we can obtain $d_8 = 20$, $d_9 = 104$ and $d_{10} = 50$, which are the demand of s_8, s_9 and s_{10}.

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.

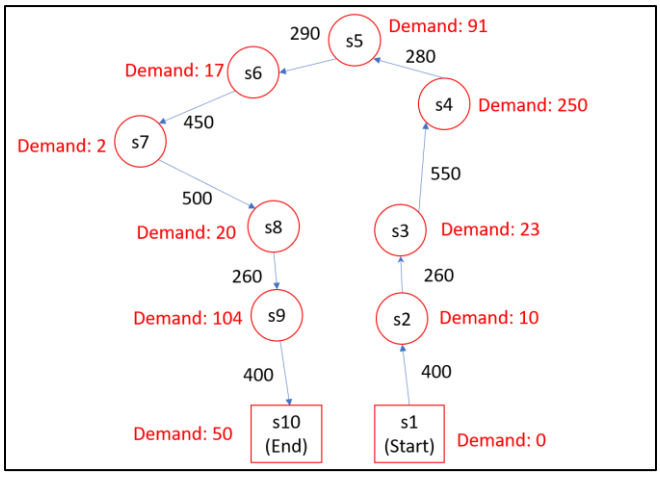

Fig. 2 The graph of the NBP.

C. *Distance*

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 s_1 should only connect to s_2, s_2 should only connect to s_3, and so on. To make it unidirectional, I make the distance from s_2 to s_1 be infinite. Then, I use Floyd-Warshall algorithm to calculate the distances between all nodes. Thus, the graph of the NBP is complete, as Fig. 2 shows.

V. Problems Found After the Presentation and Its Solution

Originally, I try to solve NBP 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.

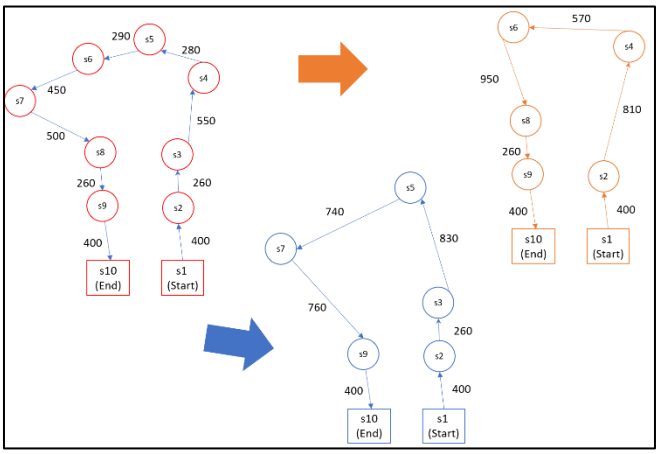

Fig. 3 The division of the graph to deal with the multi-visits of the same node

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.

Since in the NBP, every station will be multi-visited, what I divide the demand of the original graph into the demand for each route r_i of b_i has two ways: 1) Randomly choose an integer representing the students sent for each bus, and eliminate it through evolution. 2) Randomly choose the an integer, and make sure that for every node i, it follows the equation below:

$$demand_i \geq \sum_{j=\,the\ graph\ numbers} demand_{ij}, \quad (3)$$

$$\begin{cases} demand_i = \text{the demand of node i,} \\ demand_{ij} = \text{the demand of node ij in graph j} \end{cases}$$

In my opinion, I choose case 1 since it uses more power of GA and can make initial population distribute more widely than case 2.

VI. Fitness Function

A. Introduction of fitness function

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

$$f(x) = max(vehicle\ time_i) \cdot time\ weight \quad (4)$$
$$+ \sum demand\ not\ satisfied_j \cdot nonsatisfied\ penalty$$

$$\begin{cases} vehicle\ time_i\text{: Time spent on the route of vehicle i} \\ demand\ not\ satisfied_j\text{: } d_j - \text{delivery}_j\ at\ s_j \\ delivery_j\text{: The sum of students sent to station j by buses} \end{cases}$$

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.

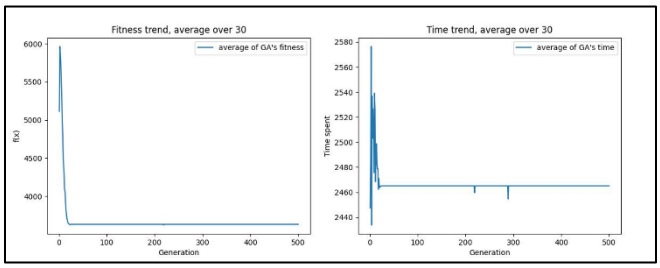

Fig. 4 The lowest fitness might not have the lowest time spent due to the weights of penalty to oversupply and undersupply. Left-hand side is the fitness and right-hand side is the time.

B. Time

First, we can divide the time spent into two parts:

$$vehicle\ time_i = \sum driving\ time + \sum unload\ time_{ij} \quad (5)$$

$$\begin{cases} vehicle\ time_i\text{: time totally spent by } b_i \\ driving\ time\text{: time driving across the route} \\ unload\ time_{ij}\text{: time waiting students to stop by } b_i \text{ at } s_j \end{cases}$$

Since for every bus, its starting point and end point are both s_1 and s_{10}, 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:

$$unload\ time_i = parking\ time + \sum get\ off\ time_{ij} \quad (6)$$

$$\begin{cases} unload\ time_i\text{: the unload time for } s_i \\ parking\ time\text{: time spent when parking} \\ get\ off\ time_{ij}\text{: time for the students to get off from } b_i \text{ at } s_j \end{cases}$$

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 b_i to s_j.

C. Demand not satisfied

Next is to decide "demand not satisfied" in equation (4), which equals to $d_j - delivery_j$. As I mentioned above, there are two ways to divide the original demands into several demands for each buses, and I choose to use method 1, which is to randomly pick an integer for each bus, and eliminate it through evolution. Thus, in fitness function, if the station doesn't satisfy its demand, then the individual should be punish to help deleting those with too biggest delivery. Both oversupply and undersupply should add penalties.

D. Weights for time and demand not satisfied

Last but not least, we need to justify the weight. Since the total demand of the NBP 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.

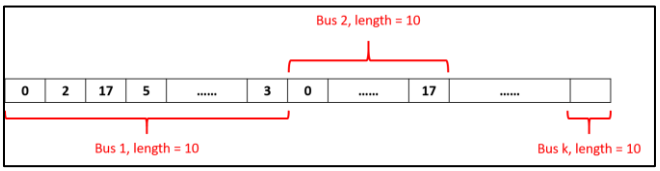

Fig. 5 The integer representation of GA with unlimited-capacity bus

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.

TABLE I. RESULTS OF THE GLOBAL MIMIMUM WHEN SETTING THE WEIGHTS NON-PROPERLY(TIME WEIGHT : NON-SATISFIED PENALTY = 1:100)

Weight			3333.75		**Time spent**			2133.75 s	
$\mathbf{d_1}$	$\mathbf{d_2}$	$\mathbf{d_3}$	$\mathbf{d_4}$	$\mathbf{d_5}$	$\mathbf{d_6}$	$\mathbf{d_7}$	$\mathbf{d_8}$	$\mathbf{d_9}$	$\mathbf{d_{10}}$
0	14	23	246	90	17	2	20	105	52
$\boldsymbol{demand\ not\ satisfied_j}$									
0	4	0	-4	1	0	0	0	1	2

E. *Weights for time and demand not satisfied*

As table I show, the demands of each station aren't satisfied strictly. So how strictly the NBP 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 2000. In this case, demands of each station will be satisfied strictly, and when the time is too large, the bus might give up satisfying all demands to compromise the large time spent.

VII. GA DESIGNMENTS

A. *Introduction of GA designments and selection operators*

To solve the NBP, I design a 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. Initially, I set the time interval be 300 seconds, since the school buses set off every 5 minutes.

Both GA generational population model. For parent selection and survivor selection, I use tournament selection, whose n = 2, to avoid the local minimum. And for survivor selection, I use (μ, λ) to avoid local minimum.

B. *Representations*

Since I split the demand of the station in NBP into several demands, order representation isn't more available, since every station might appear more than once as Fig. 3 shows. To deal with the problem, I use integer representation instead. First of all, the length chromosome of an individual follows equation (7):

$$length\ of\ chromosome\ of\ an\ individual = 10 \cdot k \quad (7)$$

k: the number of buses

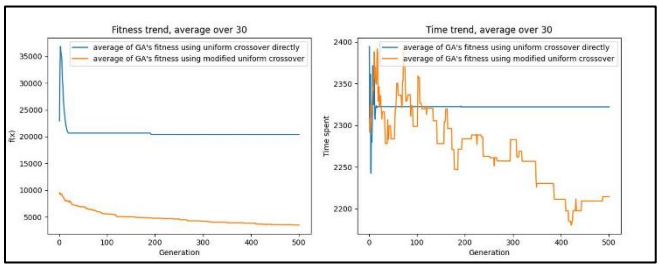

Fig. 6 The differences of fitness value and time spent between GA using uniform crossover directly (Blue line), and GA using modified uniform crossover (Orange line). Left-hand side is the fitness and right-hand side is the time.

In the chromosome, it is divided into k part. Each part has length = 10, and represents the route of a bus. Inside the route of a bus, the first value represents the students sent to s_1 by this bus, and the second value represents the students sent to s_2 by this bus, and so on. The chromosome is illustrated as Fig. 5.

C. *Crossover*

After deciding the representation, I can decide which variation operator will be used. For crossover operator, I initially use 2-point crossover, but found out that premature convergence occurs. So I use uniform crossover to prevent premature convergence. However, another problem occurs: for parent p1 and p2, if p1 and p2 do crossover, then bus b_1 in p1 only crossover to bus b_1 in p2, instead of other buses in p2.

More precisely, we can see the value of b_1 and b_4 of the global minimal solution if using uniform crossover directly, as table II shows.

TABLE II. RESULT OF B_1 AND B_4 IF USING UNIFORM CROSSOVER DIRECTLY

Bus	**s_1**	**s_2**	**s_3**	**s_4**	**s_5**
b_1	0	2	4	160	38
b_4	0	2	1	1	2
Bus	**s_6**	**s_7**	**s_8**	**s_9**	**s_{10}**
b_1	10	1	12	93	50
b_4	1	0	0	2	0
Weight		2453.75	**Time spent**		2453.75

From the table above, we can observe that most deliveries of b_1 are very large, while most deliveries of b4 are small or even zero. Since the fitness function (4) use max(vehicle $time_i$) for every b_i, b_1 will always spend the largest time since it stops at almost all stations, which isn't the result expected.

To solve the problem, I modify the uniform crossover. Originally in the uniform crossover, the demand d_{ij} sent to station s_j by bus b_i always comes from d_{ij} from the same bus and the same station. First, if doing crossover, then I will randomly pair up the buses, each pair of bus will do crossover to each other to create new children. For example, b_1 might crossover with b_4 and b_2 might crossover with b_3. Thus can avoid the problem describe above.

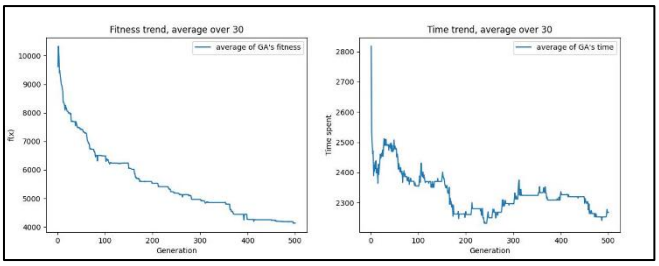

Fig. 7 The fitness and the time spent over the 500 generations taken with the average of 30 runs, left-hand side is the fitness and right-hand side is the time

Fig. 6 shows the differences of fitness value and time spent between GA using uniform crossover directly and GA using modified uniform crossover. We can observe that using modified uniform crossover will gain better result. The side effect, however, is that the time spent will keep changing, although the fitness keeps dropping.

D. Mutation

As for mutation operator, I use creep mutation to avoid the sudden change instead of doing that .

VIII. Results

The final result is shown in table III below, and Fig. 7 shows the fitness and the time spent over the 500 generations taken with the average of 30 runs.

TABLE III. The Global Minimum Solution of the NTHU Bus Problem

	b_1	**b_2**	**b_3**	**b_4**	**total**
s_1	0	0	0	0	0
s_2	4	3	0	3	10
s_3	23	0	0	0	23
s_4	126	122	1	1	250
s_5	68	5	6	12	91
s_6	11	6	0	0	17
s_7	0	2	0	0	2
s_8	10	10	0	0	20
s_9	19	57	5	23	104
s_{10}	24	19	5	2	50
Time spent	2008.75 s	2003.75 s	1188.75 s	1628.75 s	2008.75 s
Weight				2008.75	

We may see that although we use modified uniform crossover to avoid that there is a bus that dominates the total time spent, there's still a distance between the bus spending the largest amount of time and the one spending the least amount of time, i.e. b_1 and b_3. I think the method to deal with the problem is to pair up the bus spending time the most with the one spending time the least, and mixed them through uniform crossover.

IX. Conclusion and Future Works

In this term project, I've learned about what is VRP, and then created a dataset about NTHU buses delivery on 9:50, Mon, at 2023-spring. Besides, I described the mathematical model about NBP, the fitness function, and the solution to solve the problems pointed out by teacher Ting.

Furthermore, I designed the GA modal, including variation operators, selection operators, by using the concept taught by teacher on the courses. And finally get the result of the question.

Originally, I want to design a GA with limited-capacity bus and buses cannot starts simultaneously, which is more close to the real-life NBP. However, the key problem is that every bus has limited capacity, and the crossover and mutation of GA with unlimited-capacity cannot be used anymore, since the children after crossover and mutation might make the delivery of each bus exceed its capacity. On the other hand, if we can deal with the problem of crossover and mutation, I think this simulation GA will be closer to real-life NTHU Bus Problem, and can be further extended with multi-start multi-ends VRP problem.

I'm really happy to take this class. Thanks teacher Ting for having so many kinds of concept during courses and giving me advices about my term project. In the end, I put my source code on GitHub, which is free to use [6].

[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)

[6] suxen1094, Evolutionary-Computation-Term-project, June 2023 (https://github.com/suxen1094/Evolutionary-Computation-Term-project)