Capacitated Vehicle Routing Problem Using Genetic Algorithm

Pham	Thi Mai Phuong	56070503447
Siwat	Kaolueng	56070503434
Rak	Suwannarak	56070501044

Project description and scope

The main objective of the project is to write a program that answer the question: “What is the optimal set of routes for a set of capacitated vehicles to traverse in order to visit a given set of cities ?”

- ❏ the constraint weight for each vehicle is 2
- ❏ set of vehicles must visit a group of cities specifically once and come to the place to begin with minimal distance. (ie. given the cities to visit are from 1 to 10, if vehicle 1 has visited cities 1,2,3,4, vehicle 2 has to visit cities 5,6,7,8,9,10,11,12. Both vehicle has to start from city 0 and come back to city 0 so that their travelled distance in total is minimum)
- ❏ Genetic Algorithm is used to solve the problem. New solutions are generated by using crossover and mutation operators that simulates the reproduction.

Assumption

- ❏ The constraint weight for each vehicle is 2
- ❏ The number of vehicles used is 2
- ❏ The number of visiting cities is 10,11,12
- ❏ Both vehicle starts from city 0 and come back to city 0
- ❏ The distance among cities is given in the table below or the distance among cities can be randomly generated
- ❏ The capacity for each city is given in the table below or can be randomly generated

Model formulation with notation description

· Objective(s)

$$Z = Min\left\{ Max \sum_{i \in S} \sum_{j \in S} \sum_{k \in V} X_{ijk} d_{ij} \right\} \qquad (1)$$

· Decision Variable(s)

$$\sum_{k \in V} Y_{ik} = 1, \quad i \in H. \qquad (2)$$

$$\sum_{i \in H} \sum_{j \in S} q_i X_{ijk} \leq W_k, \quad k \in V. \qquad (3)$$

$$\sum_{i \in S} X_{ijk} = Y_{ik}, \quad j \in S, \quad k \in V. \qquad (4)$$

· Constraint(s)

$$\sum_{j \in S} X_{ijk} = Y_{ik}, \quad i \in S, \quad k \in V. \qquad (5)$$

$$\sum_{i \in S} \sum_{j \in S} x_{ijk} \leq |m| - 1, \quad \forall m \subseteq \{2,3 \ldots, n\}, \quad k \in V. \qquad (6)$$

In the formula : { r = ,...,1 RrgG } is a series of aggregations of distribution centre in the place R (this essay only has one); { NRRihH } i ,...1 ++= is a series of clients' aggregations in the place N ; {} } {HGS is the combination of all distribution centers and clients. { k = ,...1 KkvV } is travel vehicle k 's aggregation; qi is the demand amount of client ( ) ∈ Hii ; Wk is travel vehicle k 's loading capacity; dij is the linear distance from clienti to client j ; Dk is the travel vehicle k 's maximum travel mileage.

Input data

Table 3 Distances between each two customers

Distance (km)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	10	28	23	65	16	16	42	16	21	85	23	35	70	23
1	10	0	76	72	150	52	52	104	42	62	190	60	90	110	65
2	28	76	0	106	74	48	83	28	88	58	186	106	86	146	106
3	23	72	106	0	180	82	82	134	18	92	220	18	120	110	10
4	65	150	74	180	0	122	157	56	162	132	172	180	120	215	180
5	16	52	48	82	122	0	59	76	64	34	162	82	62	117	82
6	16	52	83	82	157	59	0	111	64	69	197	82	97	74	82
7	42	104	28	134	56	76	111	0	116	106	126	134	114	283	134
8	16	42	88	18	162	64	64	116	0	74	202	18	102	112	18
9	21	62	58	92	132	34	69	106	74	0	172	92	72	127	92
10	85	190	186	220	172	162	197	126	202	172	0	220	110	255	220
11	23	60	106	18	180	82	82	134	18	92	220	0	120	130	10
12	35	90	86	120	120	97	97	114	102	72	120	120	0	155	120
13	70	110	146	110	215	74	74	283	112	127	130	130	155	0	120
14	23	65	106	10	180	82	82	134	18	92	220	10	120	120	0

Notes : 1~14 denote the corresponding fourteen coal mines; 1 denotes Peigou; 2 denotes Daping; 3 denotes Zhanggou; 4 denotes Baiping; 5 denotes Micun; 6 denotes Chaohua; 7 denotes Gaocheng; 8 denotes Lugou; 9 denotes Laojuntang; 10 denotes Jinlong; 11 denotes Zhenxing; 12 denotes Cuimiao; 13 denotes Zhaojiazhai; 14 denotes Sanlimeiye; so do the following 1~14 in other figures and tables.

We only select 1-10 visited cities.

Table 4 Four sets of the demands

sets	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	0.8	0	0.9	1.3	1.5	0	0	0.3	0.2	1	0	0.6	0.4
2	0.2	0.8	0.3	0.7	0.5	1	1	1	2	0.1	0.1	0.2	0.2	0.3
3	2	0.2	0.5	0.1	1.3	0.1	1.5	1.8	0	0	0.2	0.3	0.4	0.4
4	0.5	1	1.5	0	0.2	0	0.8	0.2	0	1	0	0.1	0.5	1.2

We only select set #4.

Problem size

All possible route is 10 x 9 x 8 x 7 x 6 x 5 x 4 x 3 x 2 x 1 (10!) = 3,628,800 routes

All possible route is 11 x 10 x 9 x 8 x 7 x 6 x 5 x 4 x 3 x 2 x 1 (11!) = 39,916,800routes

All possible route is 12 x 11 x 10 x 9 x 8 x 7 x 6 x 5 x 4 x 3 x 2 x 1 (11!) = 479,001,600 routes

Algorithms and parameters setting

Algorithm

Using Genetic Algorithm to solve the problem.

Parameters setting

- ❏ In the first generation, the paths (chromosome) are randomly generated. The population for each generation is 100. This number can be adjusted.
- ❏ Each chromosome’s fitness value will be evaluated.
- ❏ We will keep track of the best solution at each generation and continue the algorithm when we still see the improvement in this solution throughout generations.
- ❏ In the second generations, 30% of the best solutions from the previous generation will be kept, 70% will be generated using crossover (parents will be randomly chosen).
- ❏ At this stage after all individuals are done being chosen and generated using crossover, the population will be ranked again based on their fitness values.
- ❏ In the next generations, we will continue to choose 30% of the best solutions from the previous generation and generate the rest using cross over, then rank them based on their fitness values.
- ❏ This process will go on until their is fairly no improvement observed in the best solution for a few generations

Some essential parts in the programs

Generated Chromosome(Create Routes)
- generate the paths randomly and keep them in a routes ArrayList<ArrayList<Integer>> routes

Find Fitness Values(Find Routes costs)
- evaluate the costs values of each path, put the result in an array: ArrayList<Integer> costs

Elitism (Top Ranks to Next Population)
- put top or best chromosomes to next population. depending on cross rate

Chromosome Ranking(Sorting Chromosome)
- sorting population in order to easily bring current chromosomes to next populations (automatic sorted by TreeMap of Java Collection, Key of map is the cost or fitness value and value of map is the routes, It meaning that if we get same key or cost from the crossover or random generate, some will be missed. Then, we need fill up population)

Generation Loop
- Absolutely, we need to iterate each generation.

Fill up Generation
- Sometimes, the next generation created is missed some chromosome. Therefore, this function is to generated randomly chromosomes to fill up the generation.

Mutation
- Mutate by randomly shuffle the chromosomes, It always depends on mutation rate. After chromosomes are generated:child1 and child2. They have the probability to be mutated.

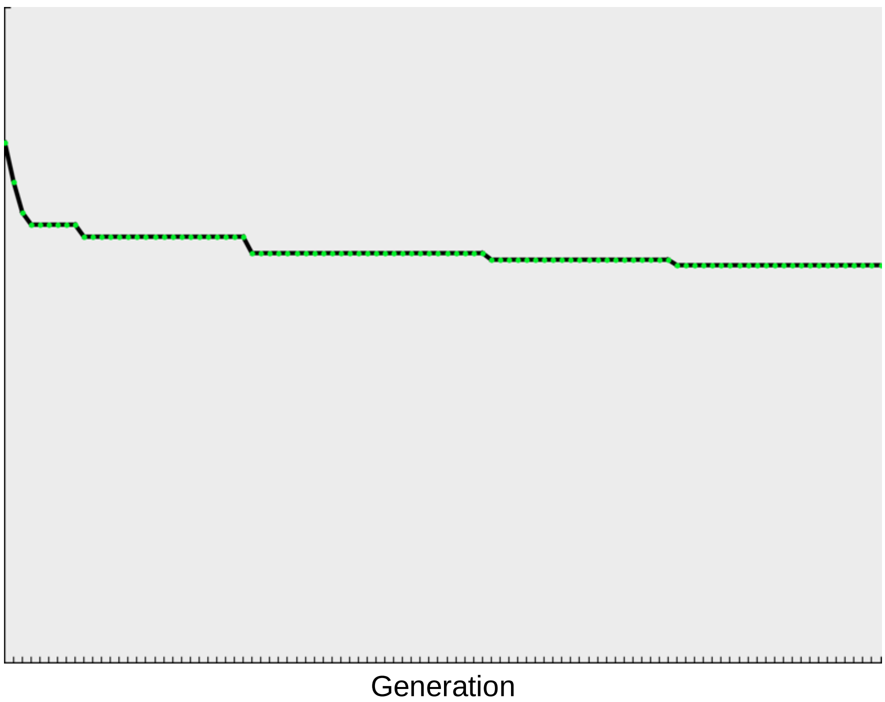
Clear next generation, routes and costs
- to re iterate the computation, the program need to clear all of these.

Crossover
- randomly pick 2 paths from current population and crossover by PMX methods, then put it in the next generation. It also depends on crossrate.

Experimental results : Quality of the Solutions

#	Cost	Route		Time (s)
		Car 1	Car 2	
1	590	[0, 2, 7, 4, 0], [0, 10, 0]	[0, 3, 8, 9, 5, 0], [0, 1, 6, 0]	0.093
2	580	[0, 4, 7, 2, 0], [0, 10, 0]	[0, 3, 8, 0], [0, 1, 6, 5, 9, 0]	0.082
3	604	[0, 2, 5, 9, 6, 0], [0, 1, 0]	[0, 3, 8, 0], [0, 10, 7, 4, 0]	0.079
4	597	[0, 9, 2, 7, 4, 0], [0, 1, 5, 0]	[0, 3, 8, 0, 6], [0, 10, 0]	0.096
5	612	[0, 10, 0] [0, 1, 6, 5, 9]	[0, 3, 8, 0], [0, 7, 2, 4, 0]	0.071

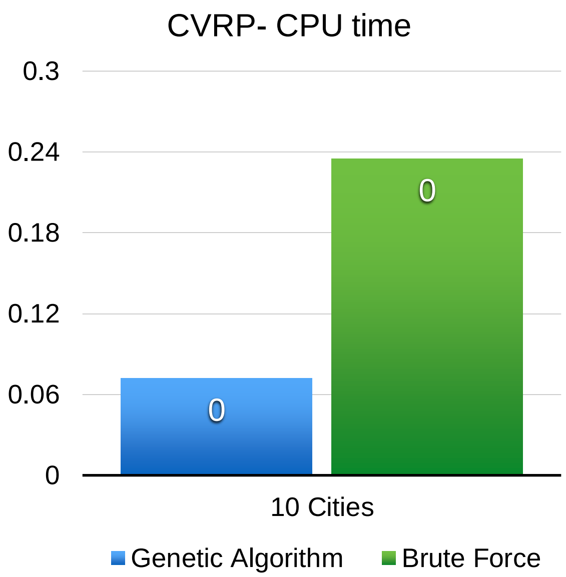
<u>Example</u> Evolution foreach generations



Computation time

- CPU Time of the Genetic Algorithm : 0.072s
- CPU Time of the Brute-force Algorithm : 0.235s

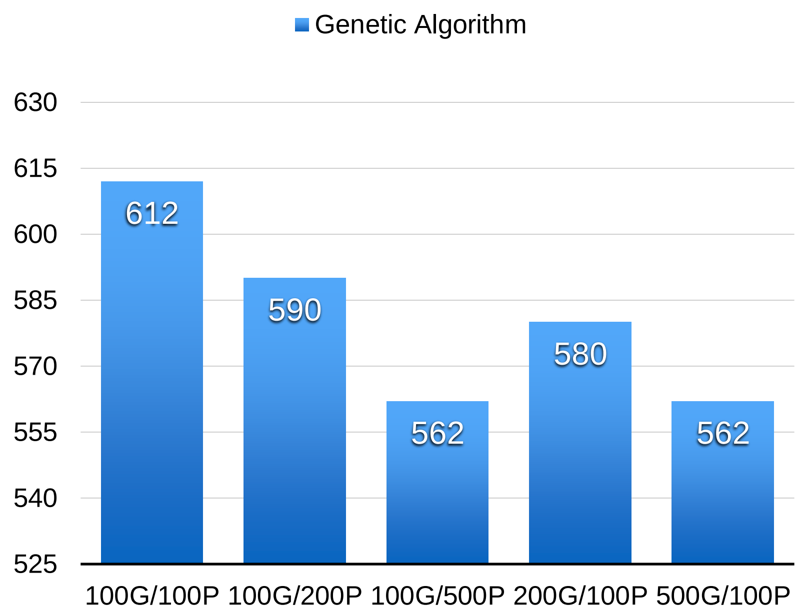
In the aspect of CPU time, the differences are highly significant. The CPU time spent by the Genetic Algorithm is small, so is its variance. The CPU time of the Genetic Algorithm is depend on number of generation and number of population, while the CPU time of the brute-force approach depends on the problem size.



Verification

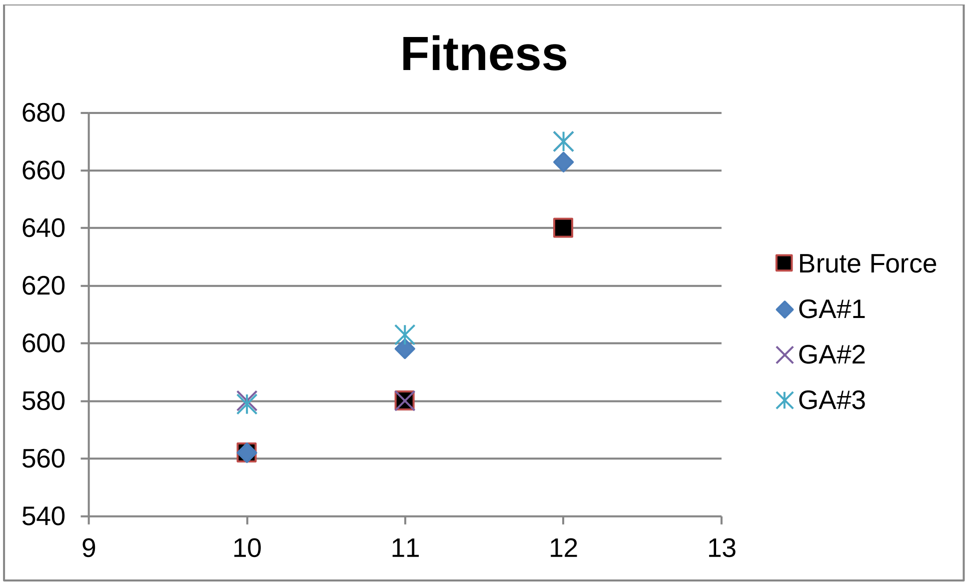
Our group use Brute force to verify the answer. Because Brute force approach can be used to find minimum cost by access all possible solutions. Number of possible solution is determined by problem size equation.

Result discussion



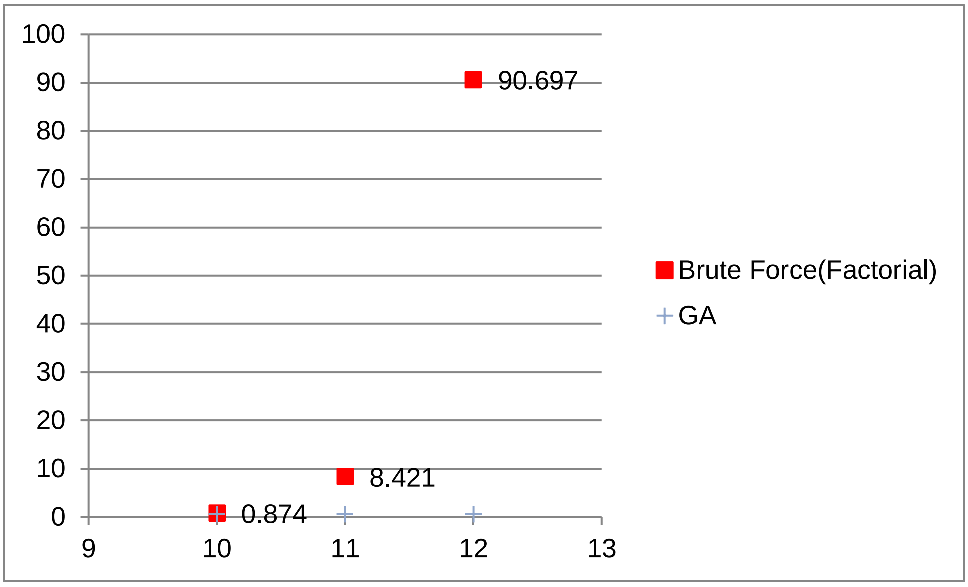
From the result if we increase number of generation or number of population. We will reach to Optimum solution

Method (Problem Size)	Fitness	Route	Time(s)
Brute Force(10!)	562	[0, 4, 7, 2, 9, 5, 0, 10, 0, 3, 8, 0, 1, 6, 0]	0.874
Brute Force(11!)	580	[0, 4, 7, 2, 9, 5, 0, 10, 0, 3, 11, 8, 0, 1, 6, 0]	8.421
Brute Force(12!)	640	[0, 4, 7, 2, 9, 5, 0, 10, 12, 0, 3, 11, 8, 0, 1, 6, 0]	90.697
GA(10!)#1	562	[0, 4, 7, 2, 9, 5, 0, 10, 0, 3, 8, 0, 1, 6, 0]	0.639
GA(10!)#2	580	[0, 9, 5, 2, 4, 7, 0, 10, 0, 3, 8, 0, 1, 6, 0]	0.685
GA(10!)#3	579	[0, 6, 1, 5, 9, 0, 3, 8, 0, 2, 7, 4, 0, 10, 0]	0.623
GA(11!)#1	598	[0, 4, 7, 2, 0, 1, 6, 5, 9, 0, 3, 11, 8, 0, 10, 0]	0.634
GA(11!)#2	580	[0, 4, 7, 2, 9, 5, 0, 10, 0, 3, 11, 8, 0, 1, 6, 0]	0.542
GA(11!)#3	603	[0, 4, 7, 2, 0, 10, 0, 3, 11, 8, 6, 0, 1, 5, 9, 0]	0.539
GA(12!)#1	663	[0, 4, 7, 2, 0, 10, 12, 0, 3, 11, 8, 0, 1, 5, 9, 6, 0]	0.58
GA(12!)#2	670	[0, 4, 7, 2, 6, 10, 12, 0, 3, 11, 8, 0, 1, 5, 9, 0]	0.623
GA(12!)#3	670	[0, 10, 12, 9, 5, 0, 2, 7, 4, 0, 3, 8, 11, 0, 1, 6, 0]	0.6



This figure shows the result that GA gains the near optimum solutions. Also, For GA#1 with problem size 10!, the result gains the optimum solution. For GA#2 problem size 11, the result also gets the optimum solution. This is summarized that GA cannot guarantee the optimum solution. However, near-optimum solution is satisfied for GA.

Compare time and Problem Size of two methods



Y Axis represents CPU time to process in seconds.
X Axis represents Problem Size in Factorial.
The result summarizes that for Brute the time is increasing dramatically. For Genetic Algorithm, the time is still constant around 0.6 seconds. The conclusion of time is GA give more efficiency for computation time.

Conclusion

The cost optimization problem for Capacitated Vehicle Routing Problem is effectively solved by the Genetic Algorithm. The experiment shows that Genetic Algorithm provides results close to those from the brute-force approach which the best solution is guaranteed. On the other hand, the Genetic Algorithm can provide the optimal or near-optimal solutions. The CPU time spent by the Genetic Algorithm is significantly less than those of the brute-force approach, and it is not varied by the problem size, but it depends on size of population and number of generation.

Reference

http://terpconnect.umd.edu/~raghavan/preprints/chap11.pdf
http://www.theprojectspot.com/tutorial-post/applying-a-genetic-algorithm-to-the-travelling-salesman-problem/5