Final Project Report :

Vehicle Capacity Routing Problem

with Genetic Algorithm

Group : TK and friends

Jirayu	Janraluek	60070501008
Prakasit	Nuchkamnerd	60070501032
Jariwat	Phansri	60070501076
Nathaphop	Sundarabhogin	60070503420

Due date : 6 December 2020

CPE341 Optimization Design and Reliability

Computer Engineering at King Mongkut's University of Technology Thonburi

Content

1. Project name : Vehicle Capacity Routing Problem with Genetic Algorithm

2. Student name(s)

1. Jirayu	Janraluek	60070501008
2. Prakasit	Nuchkamnerd	60070501032
3. Jariwat	Phansri	60070501076
4. Nathaphop	Sundarabhogin	60070503420

3. Project Description and Scope

The Vehicle Capacity Routing Problem (VRP) is a combinatorial optimization and integer programming problem which asks "What is the optimal set of routes for a fleet of vehicles to traverse in order to deliver to a given set of customers?". It generalises the well-known travelling salesman problem (TSP) [1] .

In this project we can summarize into 2 phases which is

1st Phase: To Minimize the traveling distance from a starting location to visit all customers and return back to the starting location (node 0). Each customer is visited only once.

2nd Phase: To Consider customers' demand, minimize the traveling distance with 2 vehicles (trucks, each with a capacity = 5.0 Max). The routing starts as node 0 (as the depot) to visit customers and returns back to the starting node 0. Two trucks (each with a round trip) are required to serve all 14 customers.

Scopes of the project

- Using Genetic Algorithm to solve the problem
- Consider at least 10 customers to visit
- The least of all possible solution is 1×10^6

4. Model Formulation

4.1. 1st Phase: Find shortest path with no constraint

Objective(s)

$$Min\ D = \left(\sum_{i=0}^{n-1} d(X_i, X_{i+1})\right) + d(X_n, X_0)$$

Decision Variable(s)

X_i = Customer number that the truck travel at i^{th} node

$d(X_i, X_{i+1})$ = Distance between 2 customers.

N = Number of all customer

n = Number of customer that truck visit

i = Iterate number

Constraint(s)

$i = 0, 1, 2, ..., n$

$10 \leq n \leq N$

$X_i = 0,\ 1,\ 2,\ 3, ..., N$

$X_0 = 0$; Start from node 0

$X_i \notin X - \{X_i\}$; Each customer is visited only once

4.2. 2nd Phase: Find shortest path with constraints of demand and capacity

Objective(s)

$$Min\ D = \sum_{k=1}^{M} ((\sum_{i=0}^{n_k - 1} d(X_{k,\,i}, X_{k,\,i+1}) + d(X_{k,\,n}, X_{k,\,0}))$$

Decision Variable(s)

i = Iterate number of customer of first truck

j = Iterate number of customer of second truck

k = Truck number

$X_{k,\,i}$ = Customer number that the truck k^{th} travel at i^{th} node

$d(X_{k,i}, X_{k,i+1})$ = Distance between 2 customers.

$w(X_{k,i})$ = Demand of each customer

C = Capacity of each truck

N = Number of all customer

M = Number of truck

n_k = Number of customer that k^{th} truck visit

Constraint(s)

$i = 0, 1, 2, \ldots, n_1$, $\quad j = 0, 1, 2, \ldots, n_2$, $\quad k = 1, 2, \ldots, M$

$0 \leq n_k \leq N$

$X_{k,\,i} = 0,\ 1,\ 2,\ 3, \ldots, N$

$X_{k,\,0} = 0$; Start from node 0

$X_{k,\,i} \notin (X - \{X_{k,\,i}\})$; Each customer is visited only once

$\sum_{i=1}^{M} n_i \leq N$; Number of k truck travel node do not exceed number of customers

$\sum_{j=1}^{N} w(j) = \sum_{k=1}^{M} \sum_{i=1}^{n_k} w(X_{k,i})$; Must serve all customer demand

$\sum_{i=1}^{n_k} w(X_{k,i}) \leq C$; Do not serve customer demand exceed each truck capacity

5. Input Data: Description and Source of Data

The information is about traveling distance between each two customers and sets of the demand.

Dataset : Distance between each two customers (TermProject CPE341-2020 Optimization.pdf).

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

This table shows the distance between each two customers' nodes.

Note Number 1 - 14 from the table represents the Customer name below.

No.	Customer name	No.	Customer name	No.	Customer name
1	Peigou	6	Chaohua	11	Zhenxing
2	Daping	7	Gaocheng	12	Cuimiao
3	Zhanggou	8	Laojuntang	13	Zhaojiazhai
4	Baiping	9	Sanlimeiye	14	Sanlimeiye
5	Micun	10	Jinlong		

Dataset : Four sets of the demands (TermProject CPE341-2020 Optimization.pdf).

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

This table shows sets of the demand

6. Problem size: number of possible solutions

All possible solutions from 14 node to travel and each can travel once

$14! = 87,178,291,200$ Solutions

7. Algorithm and Parameter Setting

Algorithm : Genetic Algorithm

1. Random possible paths start from 0, for example

0	2	1	4	5	3

2. Evaluate the distance and sort the distance from lowest to highest.
3. Select parent for next generation
4. Crossover path and get off-spring to be new path
5. Mutation path by random swap city
6. Do over from 2nd step for n generation

Generating Path

Generate the random path at least possible minimum node to n node. Also, split the path depends on demand constraint.

1st Phase

1. Generate the random from 1 to N-nodes distance

1	2	3	4	5

2. Shuffle path

2	1	4	5	3

3. Remove the path according to the number of nodes that want to travel. For example, want to travel at least 3 nodes

2	1	4

4. Add node 0 in to the first of path to be start location

0	2	1	4

2nd Phase

1. Generate path from 1 to N-nodes distance

1	2	3	4	5

2. Shuffle path

2	1	4	5	3

3. Remove the path according to the number of customers who have demand

   Demand = [0,1,1,1,0] -> number of customer who have demand = 3

2	1	4

4. If in path, there are nodes that have a customer's demand more than 0, append nodes to the list

   3 has demand 1 but does not contain in path, so append 3 in path.

2	1	4	3

5. Shuffle path

1	4	3	2

6. Check that 1 truck can travel to serve customers until which node (depends on demand capacity). Then cut the path that 1 truck can travel from the big path to be the path of that truck. The rest of the path will be served by another truck

Assume each truck has capacity = 2

1st truck

1	4

2nd truck

3	2

7. Repeat 6. according to the number of cities remaining in the path
8. Insert node 0 in to the first node of each truck's path to be start location

1st truck

0	1	4

2nd truck

0	3	2

<u>Selection</u>

Survival Selection is executed by sorting the list of solutions from minimum distance and selecting top rank of paths to use in the next step. Number of selections depends on the number of populations subtracting with the number of crossover.

For parent selection we are using a roulette wheel selection to random by considering the lower distance result the more probability to be selected

<u>Crossover</u>

1. In each parent, cut start node (0) in path and concatenate path of trucks

   Before : 1st truck 2nd truck

0	1	4		0	3	2

   After :

1	4	3	2

2. Crossover by using ordered crossover to generate child

   Parent :

1	4	3	2		4	2	1	3

   Child (Assume cutting position is 2 and 3) :

2	4	3	1

3. Check that 1 truck can travel to serve customers until which node (depends on demand capacity). Then cut the path that 1 truck can travel from the big path to be the path of that truck. The rest of the path will be served by another truck.

   Assume each truck has capacity = 2

2	4		3	1

4. Repeat 3. according to the number of cities remaining in the path
5. Insert node 0 in to the first node of each truck's path to be start location

   1st truck 2nd truck

0	2	4		0	3	1

<u>Mutation</u>

1. Random number if that number is less than mutation rate, mutate that solution

   Assume : Random number = 0.1 & mutation rate = 0.2

   Random number < mutation rate

   Solution 1

0	2	4

0	3	1

2. Random select path in solution

   Solution 1

0	2	4

0	3	1

3. Random 2 positions in that path and 2 positions must not same

   Assume :

   Random first position : 1

   Random second position : 2

4. Swap node of that 2 positions

   Before :

0	2	4

0	3	1

   After :

0	4	2

0	3	1

5. Repeat from 1. to 4. until complete every solution

Solution 1

0	4	2		0	3	1

Solution 2

0	1	4		0	3	2

<u>Parameter</u> :

1. List of customer’s node and distance to other nodes
2. List of customer’s demands
3. Number of population
4. Generation
5. Capacity
6. Number of truck
7. Crossover rate
8. Mutation rate

8. Experiment Result

8.1. Compare Genetic Algorithm with Brute Force Algorithm

In this section we will use brute force algorithms to compare results with GA and because of a long execution time of brute force we decide to compare GA and brute force only from 8 nodes to 12 nodes.

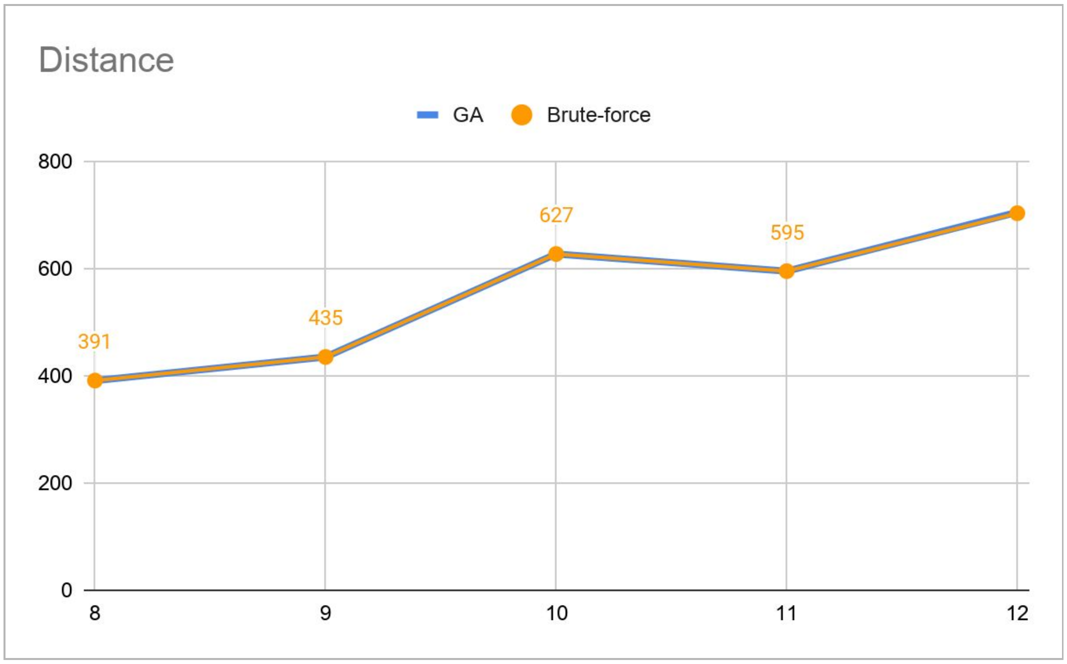
1st Phase: <u>Shortest path with no constraint</u>

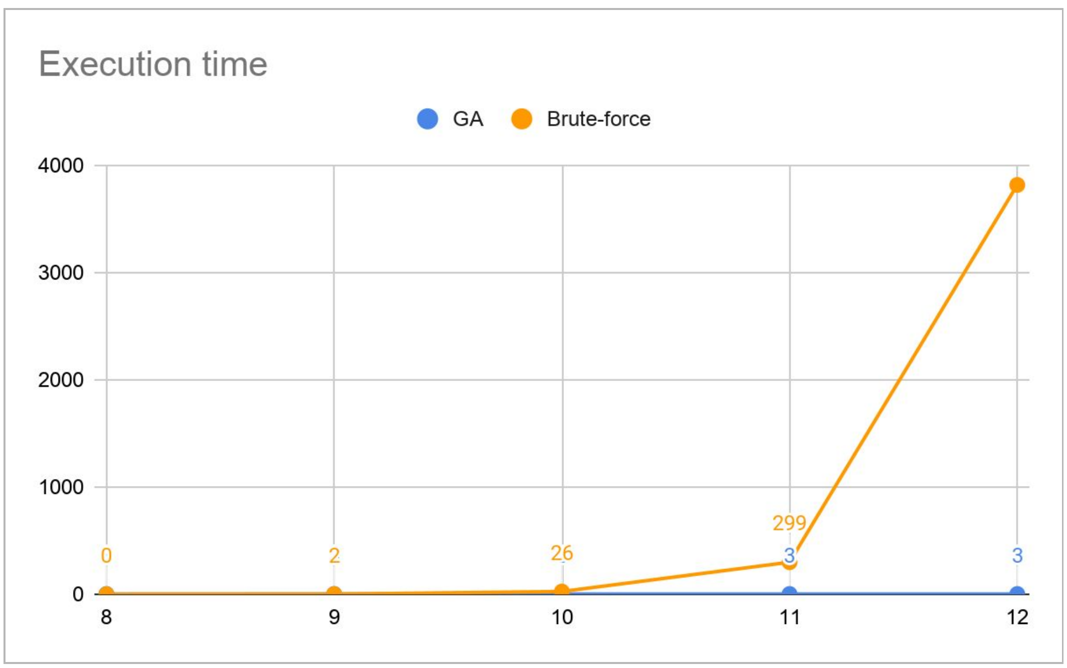
- Parameter setting
    - Genetic Algorithm
        - Number of nodes : 8-12 nodes
        - Fix start node : at node 0
        - Population: 100
        - Generations : 300
        - Crossover rate : 0.7
        - Mutation rate : 0.1
        - Round of Process : 5 times
    - Brute force
        - Number of nodes : 8-12 nodes
        - Fix start node : at node 0
- Result : Table

Node	GA		Brute-force	
	Average Distance	Average Execution time (s)	Distance	Execution time (s)
8	391	3	391	0
9	435	3	435	2
10	627	3	627	26
11	595	3	595	299
12	704.8	3	703	3816

Table for comparing GA and Brute-force by distance and execution time

Result : Graph





From the graph above, the distance of GA is very similar to brute force but the execution time of brute force is much longer than GA.

2nd Phase: Shortest path with constraint of demand and capacity

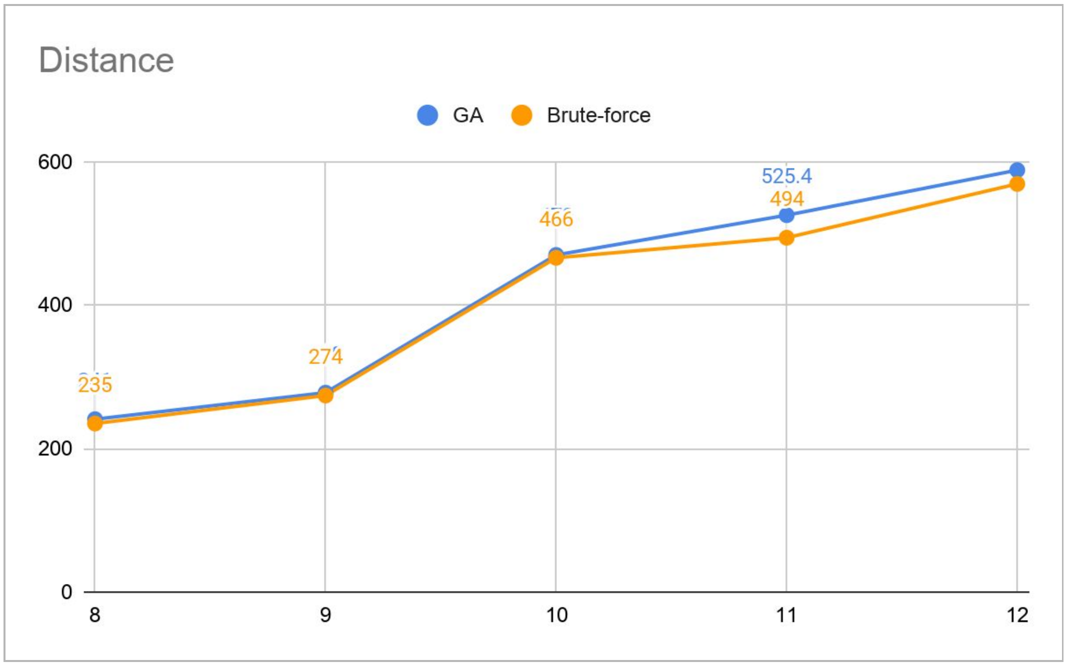
Testing GA

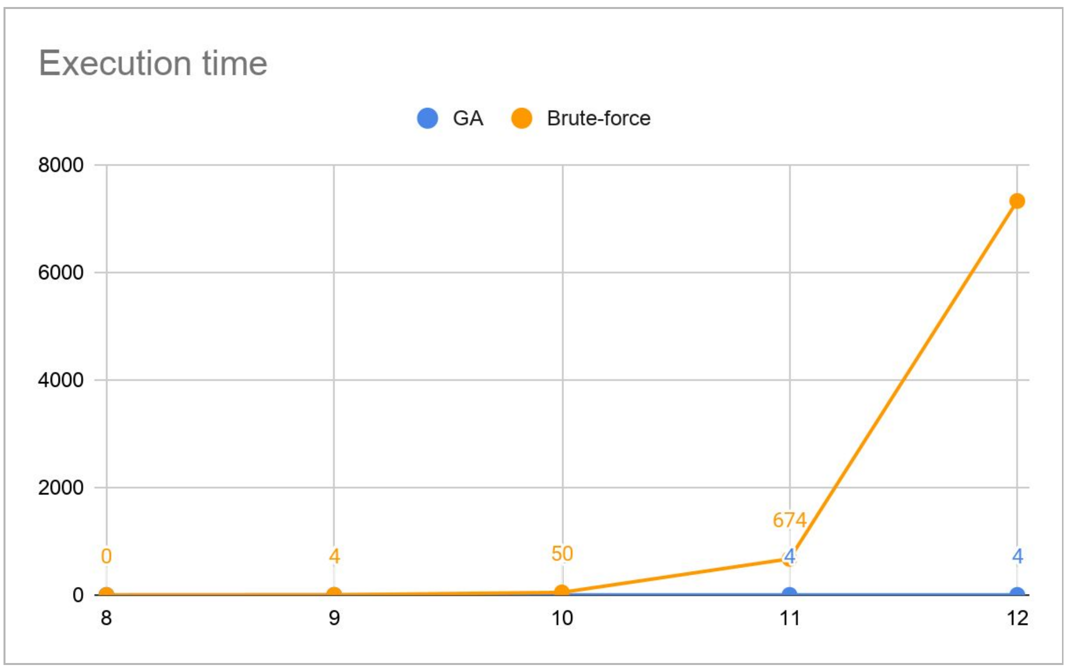
- Parameter setting
    - Genetic Algorithm
        - Number of nodes : 8-12 nodes
        - Fix start node : at node 0
        - Population: 100
        - Generations : 300
        - Crossover rate : 0.7
        - Mutation rate : 0.1
        - Round of Execution : 5
        - number of trucks : 2
        - Capacity of trucks : 4
        - **Demand Set : 1st set**
    - Brute force
        - Number of nodes : 8-12 nodes
        - Fix start node : at node 0
- Result : Table

Node	GA		Brute-force	
	Average Distance	Average Execution time (s)	Distance	Execution time (s)
8	241	4	235	0
9	278	4	274	4
10	470	4	466	50
11	525.4	4	494	602
12	588.2	4	569	7325

Table for comparing GA and Brute-force by distance and execution time

Result : Graph



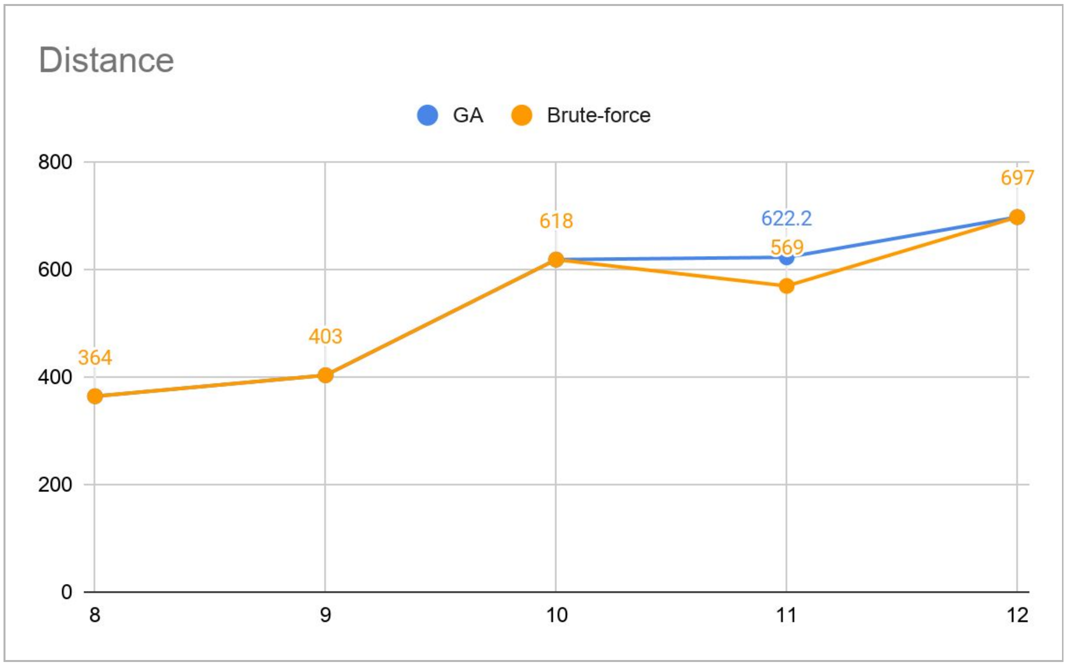


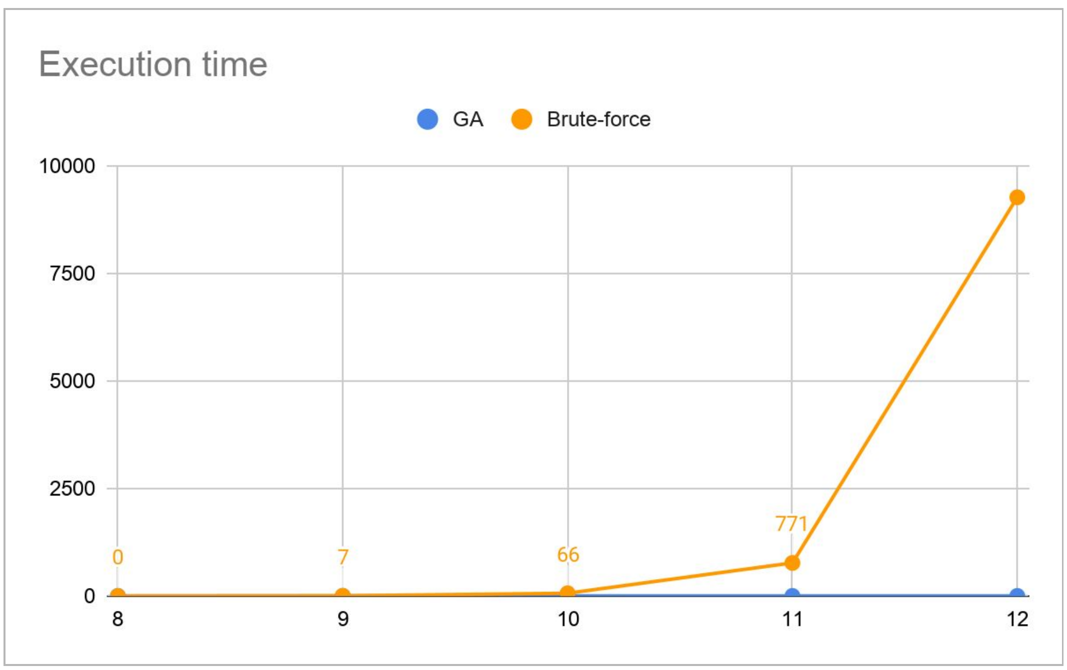
- Parameter setting
  - Genetic Algorithm
    - Number of nodes : 8-12 nodes
    - Fix start node : at node 0
    - Population: 100
    - Generations : 300
    - Crossover rate : 0.7
    - Mutation rate : 0.1
    - Round of Execution : 5
    - number of trucks : 2
    - Capacity of trucks : 4 for node 8, 9
5 for node 10, 11, 12
    - **Demand Set : 2nd set**
  - Brute force
    - Number of nodes : 8-12 nodes
    - Fix start node : at node 0
- Result : Table

Node	GA		Brute-force	
	Average Distance	Average Execution time (s)	Distance	Execution time (s)
8	365	4	365	0
9	438	4	438	7
10	625.2	4	618	66
11	600.4	4	569	745
12	697	4	697	9265

Table for comparing GA and Brute-force by distance and execution time

Result : Graph





From the graph above,for both the 1st set of demands and the 2nd set of demands the distance of GA is slightly more than brute force but the execution time of GA is much less than brute force.

8.2. Genetic Algorithm Result

GA with no constraint

```
The shortest path is:  [[0, 10, 12, 4, 7, 2, 5, 9, 1, 8, 11, 14, 3, 13, 6]]
Distance:  823

[ ] sum_distance(d_big, x[-1][0], True)

0 -> 3: 23.00
Sum Distance:  23
==========
3 -> 14: 10.00
Sum Distance:  33
==========
14 -> 11: 10.00
Sum Distance:  43
==========
11 -> 8: 18.00
Sum Distance:  61
==========
8 -> 1: 42.00
Sum Distance:  103
==========
1 -> 6: 52.00
Sum Distance:  155
==========
6 -> 5: 59.00
Sum Distance:  214
==========
5 -> 9: 34.00
Sum Distance:  248
==========
9 -> 2: 58.00
Sum Distance:  306
==========
2 -> 7: 28.00
Sum Distance:  334
==========
7 -> 0: 42.00
Sum Distance:  376
==========
376
```

Single objectives

Number of nodes : 14 (fixed start node = 0)

Best distance : 823

Best path : [0, 10, 12, 4, 7, 2, 5, 9, 1, 8, 11, 14, 3, 13, 6]

Execution time : 4 s

<u>GA with capacity and weight constraint</u>

```
=========== First Path ============
0 -> 11: 23.00
Sum Distance:  23
=========
11 -> 14: 10.00
Sum Distance:  33
=========
14 -> 8: 18.00
Sum Distance:  51
=========
8 -> 13: 112.00
Sum Distance:  163
=========
13 -> 6: 74.00
Sum Distance:  237
=========
6 -> 5: 59.00
Sum Distance:  296
=========
5 -> 12: 62.00
Sum Distance:  358
=========
12 -> 10: 110.00
Sum Distance:  468
=========
10 -> 0: 85.00
Sum Distance:  553
=========
```

```
============ Second Path =============
0 -> 9: 21.00
Sum Distance:  21
==========
9 -> 2: 58.00
Sum Distance:  79
==========
2 -> 7: 28.00
Sum Distance:  107
==========
7 -> 4: 56.00
Sum Distance:  163
==========
4 -> 0: 65.00
Sum Distance:  228
==========
======================================
Distance: 781.00
======================================
```

Single objectives with constraint

Number of nodes : 14 (fixed start node = 0)

Number of trucks : 2

Demand set : 1st set

Each Truck capacity : 5

Best distance : 718

Best path : [0, 11, 14, 8, 13, 6, 5, 12, 10]

[0, 9, 2, 7, 4]

Execution time : 4 s

```
============ First Path =============
0 -> 3: 23.00
Sum Distance:  23
==========
3 -> 14: 10.00
Sum Distance:  33
==========
14 -> 11: 10.00
Sum Distance:  43
==========
11 -> 8: 18.00
Sum Distance:  61
==========
8 -> 1: 42.00
Sum Distance:  103
==========
1 -> 5: 52.00
Sum Distance:  155
==========
5 -> 9: 34.00
Sum Distance:  189
==========
9 -> 12: 72.00
Sum Distance:  261
==========
12 -> 10: 110.00
Sum Distance:  371
==========
10 -> 0: 85.00
Sum Distance:  456
==========
```

```
============ Second Path =============
0 -> 4: 65.00
Sum Distance:  65
==========
4 -> 7: 56.00
Sum Distance:  121
==========
7 -> 2: 28.00
Sum Distance:  149
==========
2 -> 6: 83.00
Sum Distance:  232
==========
6 -> 13: 74.00
Sum Distance:  306
==========
13 -> 0: 70.00
Sum Distance:  376
==========
======================================
Distance: 832.00
======================================
```

Single objectives with constraint

Number of nodes : 14 (fixed start node = 0)

Number of trucks : 2

Demand set : 2^{nd} set

Each Truck capacity : 5

Best distance : 832

Best path : [0, 3, 14, 11, 8, 1, 9, 12, 10]
[0, 4, 7, 2, 6, 13]

Execution time : 4 s

8.3. Algorithm’s Stability

- Parameter setting
  - Genetic Algorithm
    - Number of nodes : 14 nodes
    - Fix start node : at node 0
    - Population: 100
    - Generations : 300
    - Capacity of trucks : 5
    - Mutation rate : 0.1
    - Crossover rate : 0.7
    - number of trucks : 2
    - **Demand Set** : 1st set
    - **Round of Execution** : 20

Round	Distance
1	722
2	708
3	732
4	738
5	708

Round	Distance
6	778
7	765
8	748
9	708
10	748

Round	Distance
11	718
12	728
13	733
14	738
15	723

Round	Distance
16	786
17	733
18	718
19	718
20	698

AVG	732.4
SD	23.36529
MAX	786
MIN	698

Table shows result of GA 20 round with average, SD, min, max

Calculate Coefficient Variation

(SD/AVG) x 100 = (23.36529 / 732.4) x 100 = 3.19% From running GA algorithm 20 times, it has average distance 732.4 units, maximum distance 786 units, minimum distance is 698 units, and with coefficient variation 3.19% which is not high. It means that the implemented Genetic Algorithm is quite stable.

9. Result discussion and verification

From the comparison between genetic algorithm and brute force, the distance result from genetic algorithm is very close to the brute force method. But the execution time of genetic algorithms is much better than brute force if the number of nodes is big as you can see from the result in 8.1. On the other hand, if the number of nodes is very small, the brute force method is better than a genetic algorithm.

Moreover, we can make a verification that the result from genetic algorithms is correct by comparing it with the result from brute force. The distance result of genetic algorithms is close to the distance result of brute force.

In addition, we have tested the stability of the algorithm by doing an experiment with data set 14 nodes 20 times and calculate coefficient variation to see the variation of distance result in 8.3. From the result in 8.3, coefficient variation is 3.19% which is very low and confirms that the genetic algorithm is very stable.

In conclusion, the genetic algorithm that we create to solve vehicle capacity routing, we can make a verification and check the performance by comparing the genetic algorithm with the brute force method and very stable from calculating coefficient variation.

Reference

[1] Wikipedia. “Vehicle_routing_problem” [online]. Retrieve from https://en.wikipedia.org/wiki/Vehicle_routing_problem. (25 Nov 2020).

APPENDIX

PHASE 1

Distance							
node/round	1	2	3	4	5	averageGA	Bruteforce
8	391	391	391	391	391	391	391
9	435	435	435	435	435	435	435
10	627	627	627	627	627	627	627
11	595	595	595	595	595	595	595
12	703	703	703	703	712	704.8	703

Time							
node/round	1	2	3	4	5	averageGA	Bruteforce
8	3	3	3	3	3	3	0
9	3	3	3	3	3	3	2
10	3	3	3	3	3	3	26
11	3	3	3	3	3	3	299
12	4	4	4	4	4	4	3816

PHASE 2_1

Distance							
node/round	1	2	3	4	5	averageGA	Bruteforce
8	245	245	245	235	235	241	235
9	274	274	284	284	274	278	274
10	466	466	476	466	476	470	466
11	494	494	505	559	575	525.4	494
12	565	608	583	595	590	588.2	569

Time							
node/round	1	2	3	4	5	averageGA	Bruteforce
8	4	4	4	4	4	4	0
9	4	4	4	4	4	4	4
10	4	4	4	4	4	4	50
11	4	4	4	4	4	4	602
12	4	4	4	4	4	4	7325

PHASE 2_2

Distance							
node/round	1	2	3	4	5	averageGA	Bruteforce
8	365	365	365	365	365	365	365
9	438	438	438	438	438	438	438
10	630	618	618	630	630	625.2	618
11	628	613	569	623	569	600.4	569
12	697	697	697	697	697	697	697

Time							
node/round	1	2	3	4	5	averageGA	Bruteforce
8	4	4	4	4	4	4	0
9	4	4	4	4	4	4	7
10	4	4	4	4	4	4	66
11	4	4	4	4	4	4	745
12	4	4	4	4	4	4	9265

-

Node	GA		Brute-force	
	Average Distance	Average Execution time (s)	Distance	Execution time (s)
8	241	4	235	0
9	278	4	274	4
10	470	4	466	50
11	525.4	4	494	602
12	588.2	4	569	7325