```
procedure GeneticAlgorithm(populationSize)
     Create initial population (random feasible solutions)
     with the size populationSize
     While stopping criteria not met
          Calculate fitness of all individuals
          Select the best 2 individuals as parents
          Apply crossover, create 2 offspring
          Replace the worst individual with new offspring
          If rand(0, 1) < mutationProbability
               Select 1 random individual, apply mutation
     operation
          end
     end
end GeneticAlgorithm
```

Figure 3 - 2: Pseudo-code for the basic Genetic Algorithm

Detailed explanations for each stage are stated below:

Initial population:

In the basic GA, individuals in the initial population are randomly created, but only the feasible individuals are picked up.

Selection:

After calculating the fitness value of all individuals, the two best individuals are chosen as parents to perform the crossover operators.

Crossover:

Cycle Crossover Operation and Order 1 Crossover Operation are selected for this stage. One of the two operations is singled out for this stage by using roulette wheel mechanism with equal probability for both operations.

- Cycle Crossover Operation:

The Cycle Crossover operator selects a number of cycles between two parent individuals. Then, to create Offspring 1, cycle 1 is copied from parent 1, cycle 2 from parent 2, cycle 3 from parent 1, and so on. It will be started with the first value in Parent 1, then drop down to find the equivalent in Parent 2, then look up to find the equivalent in Parent 1. Then restate the process until the first value is found, and finish the cycle. An example is stated below:

Parent 1	8	4	7	3	6	2	5	1	9	0
Parent 2	0	1	2	3	4	5	6	7	8	9

Cycle 1: 8 goes to 0, 0 goes to 9, 9 goes to 8 => stop. Cycle 1 values are 8, 9, 0.

Parent 1	~~**8**~~	~~**4**~~	~~**7**~~	3	6	2	5	1	~~**9**~~	~~**0**~~
Parent 2	~~**0**~~	1	2	3	4	5	6	7	~~**8**~~	~~**9**~~
Cycle	1								1	1

Cycle 2: 4 goes to 1, 1 goes to 7, 7 goes to 2, 2 goes to 5, 5 goes to 6, 6 goes to 4 => stop. Cycle 2 values are 4, 1, 7, 2, 5, 6.

Parent 1	~~8~~	~~**4**~~	~~**7**~~	3	~~**6**~~	~~**2**~~	~~**5**~~	~~**1**~~	~~9~~	~~0~~
Parent 2	~~0~~	~~**1**~~	~~**2**~~	3	~~**4**~~	~~**5**~~	~~**6**~~	~~**7**~~	~~8~~	~~9~~
Cycle	1	2	2	3	2	2	2	2	1	1

The only remaining value is 3, so the value that cycle 3 has is 3.

Filling in the offspring:

Values of Cycle 1 that belong to Parent 1 will be in Offspring 1, value from Parent 2 will be in Offspring 2

Values of Cycle 2 that belong to Parent 1 will be in Offspring 2, value from Parent 2 will be in Offspring 1

Values of Cycle 3 that belong to Parent 1 will be in Offspring 1, value from Parent 2 will be in Offspring 2

Parent 1	8	4	7	3	6	2	5	1	9	0
Parent 2	0	1	2	3	4	5	6	7	8	9
Cycle	**1**	**2**	**2**	**3**	**2**	**2**	**2**	**2**	**1**	**1**
Offspring 1	8	1	2	3	4	5	6	7	9	0
Offspring 2	0	4	7	3	6	2	5	1	8	9

- Order 1 Crossover Operation

Order 1 Crossover is permutation crossover. Basically, a group of consecutive nodes from Parent 1 drops down, and remaining values are placed in the offspring in the order which they appear in Parent 2. An example is stated below:

Parent 1	8	4	7	3	6	2	5	1	9	0
Parent 2	0	1	2	3	4	5	6	7	8	9

Choose randomly a group of nodes from Parent 1, then removes those values from Parent 2. In this case, the group includes nodes 3, 6, 2, 5, 1.

Parent 1	8	4	7	**3**	**6**	**2**	**5**	**1**	9	0
Parent 2	0	~~1~~	~~2~~	~~3~~	4	~~5~~	~~6~~	7	8	9

Insert the group into Offspring 1 while still keeping it in its relative position in Parent 1:

Offspring 1	-	-	-	3	6	2	5	1	-	-

Insert the remaining values of Parent 2 (0, 4, 7, 8, 9) into the gaps in Offspring 1:

Offspring 1	**0**	**4**	**7**	3	6	2	5	1	**8**	**9**

To obtain Offspring 2, swap the positions of the two parents, then repeat the procedure.

Offspring 2	8	7	2	3	4	5	6	1	9	0

Mutation:

In each generation, there is a possibility that one of the individuals will encounter a mutation. In this process, 2 cut points will be selected randomly, and the nodes in-between those cut points are scrambled.

Before	8	7	2	**3**	**4**	**5**	**6**	**1**	9	0
After	8	7	2	**1**	**4**	**6**	**3**	**5**	9	0

The basic GA is a straightforward process which gives a certain advantage. It is straight forward, so the algorithm does not require much processing power. Besides, it can find a large area of the solution space by having many individuals in a population, reducing the ability of getting stuck in local optima.

3.5. MATLAB Software

MATLAB is a software which uses high-level languages such as C, C++, C#, Java, Fortran and Python. It integrates computing, visualization, and programming in an environment where users can easily use it. MATLAB is used for:

- Math and computing
- Development of algorithms
- Model, simulate, create prototypes
- Analyze and explore data visualization
- Scientific and technical graphics

CHAPTER 4 DATA COLLECTION

4.1. Types of data

- Customer's information: Name, location, demand, and time windows were taken from the company. Customers were divided into 4 groups based on the distance. Group 1 was a set of customers that did not have same criteria. Group 2 included customers who had the large distances from the depot. Group 3 had medium distances from the depot. Group 4 was a set of customers who located in District 12 – same place with the depot.

Table 4 - 1: Customers' location

Group	Location
1 (15 customers)	District 5, Binh Thanh District, Binh Tan District, Binh Duong, Binh Phuoc
2 (13 customers)	Binh Phuoc, Long An
3 (10 customers)	District 6, District 10, District 11
4 (12 customers)	District 12

- Customers' demand: The order of customers was collected from the Business Department.
- Distance between customers and depot: It was calculated from the depot to customers, from customers to customers and from customers return to the depot. The customers' location used to establish distances was given by the company.

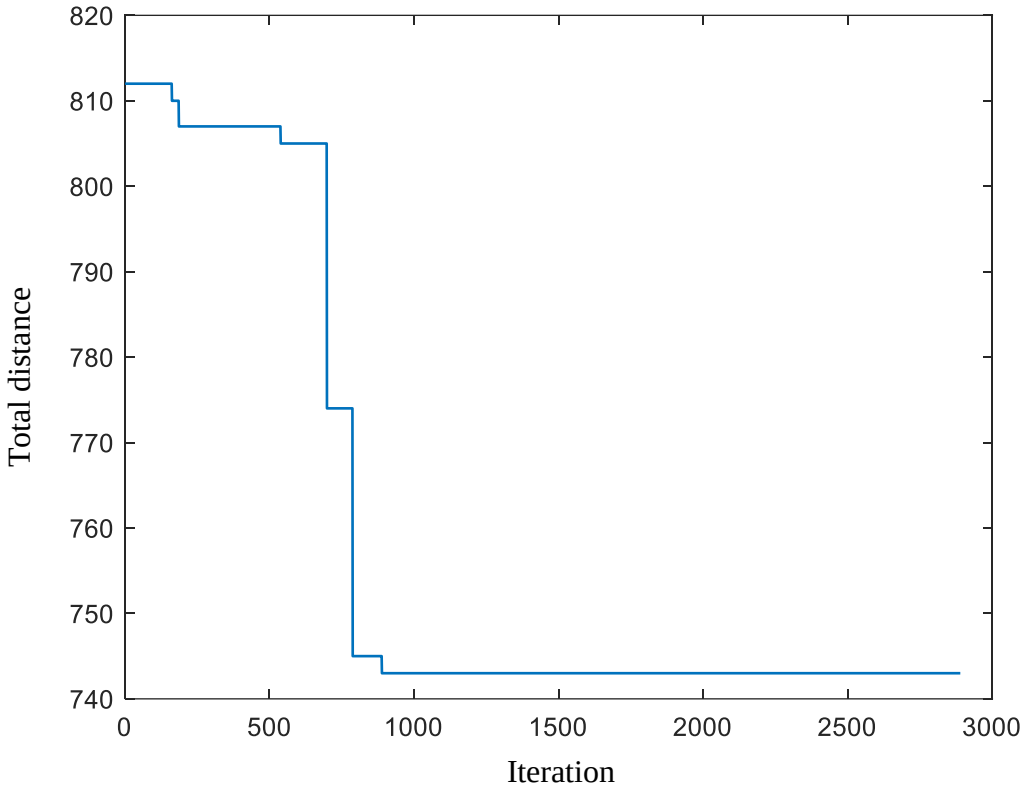

Figure 5 - 1: Total distance and iteration of Group 1

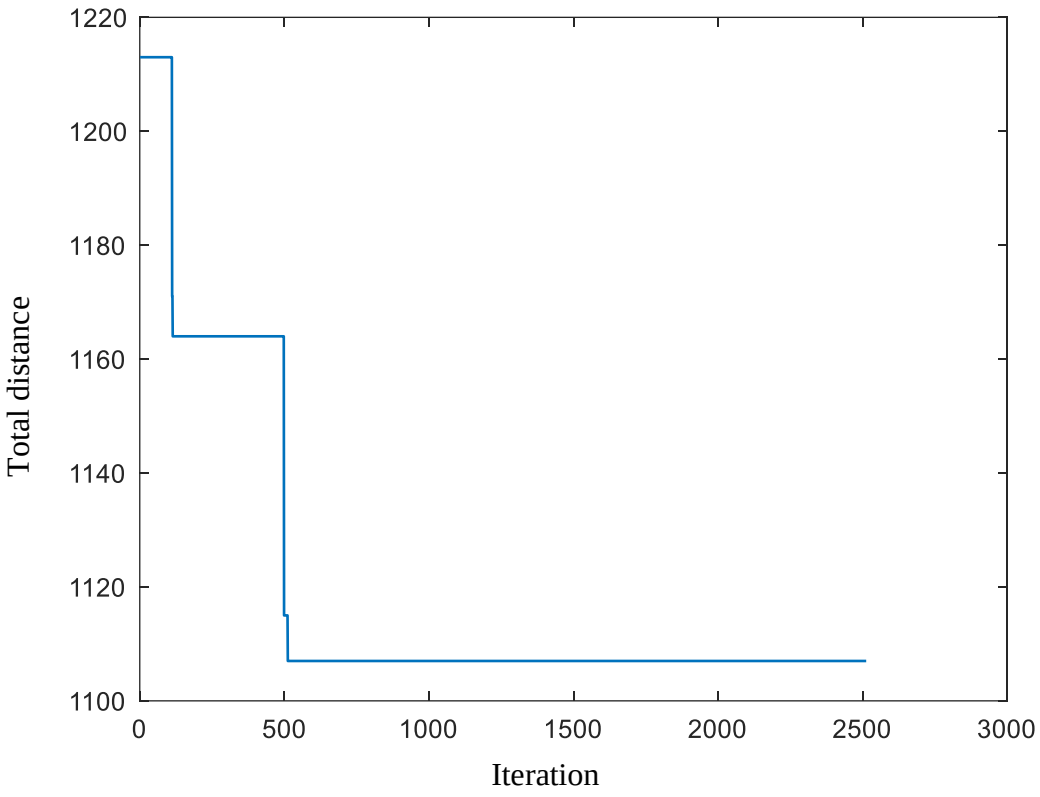

Figure 5 - 2: Total distance and iteration of Group 2

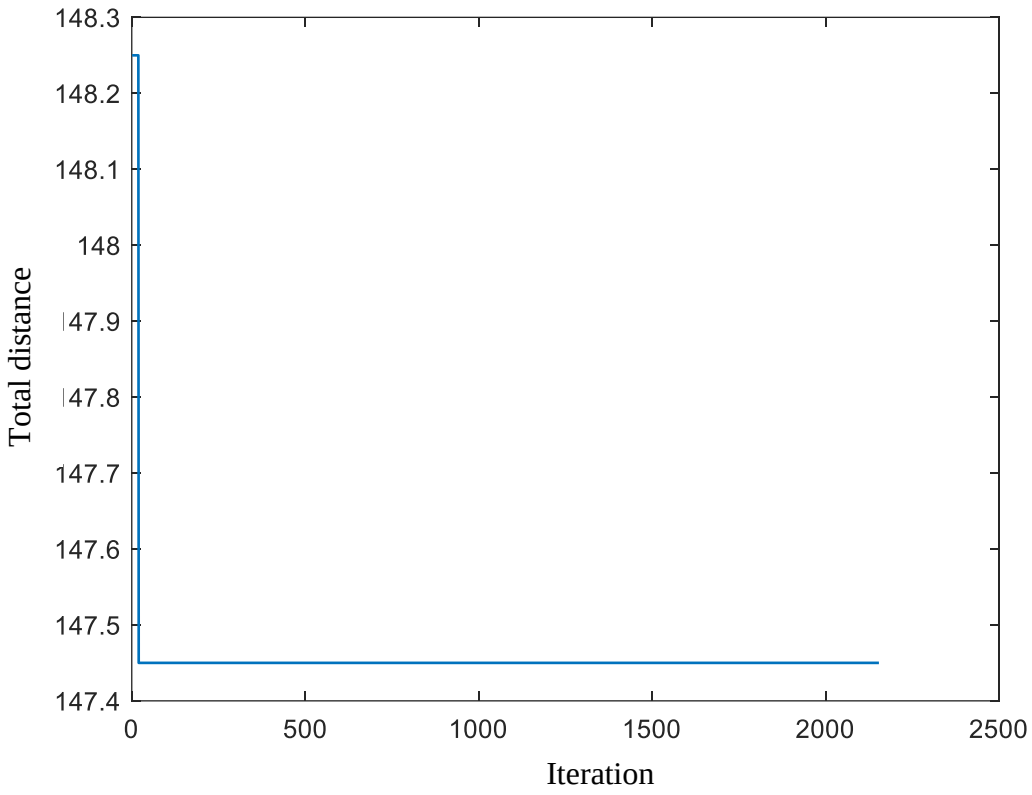

Figure 5 - 3: Total distance and iteration of Group 3

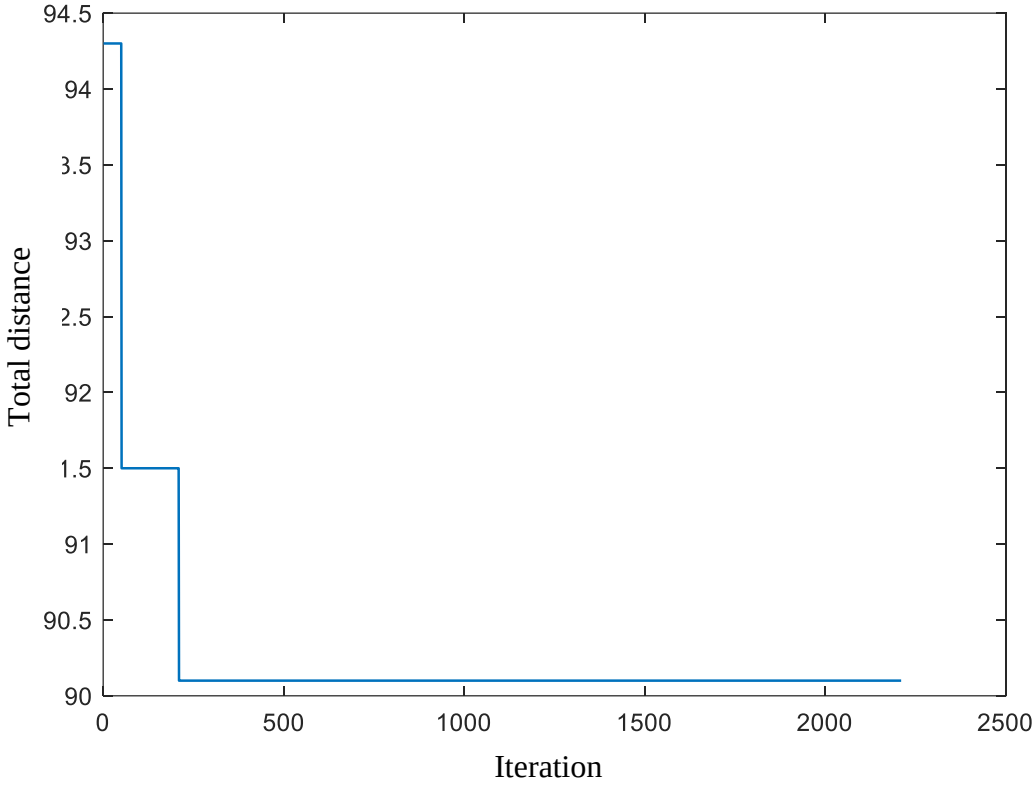

Figure 5 - 4: Total distance and iteration of Group 4

Table 5 - 4: Result of using MATLAB

	Total travelling distance (km)	Total demand (kg)	Total cost (VND)
Group 1	743	6,615	5,111,000
Group 2	1,107	6,558	8,120,000
Group 3	147.45	6,276	554,000
Group 4	90.10	6,059	840,800
TOTAL	4,259.8	25,508	13,135,000

8.2. Genetic Algorithm (Using MATLAB)

8.2.1. Main program

```
clear
tic
    global noOfCustomers
    global noOfVehicles
    global distance
    global demand
    global timeWindow
    global travelTime
    global capacity

load('15customers.mat');
fprintf('Instance info: \n');
fprintf('- Number of customer is %2.0f\n', noOfCustomers);
fprintf('- Number of vehicles is %2.0f\n', noOfVehicles);
pop = input('Enter number of individuals in a population: ');
convergLimit = input('Enter convergence limit: ');
chromLength = noOfCustomers + noOfVehicles - 1;
solutionArray = zeros(1,chromLength);
```

```
TC = 999999999999999;
TD = 0;
mutationProba = 0.3;

% Two best individuals
S1 = zeros(1, chromLength + 1);
S2 = zeros(1, chromLength + 1);
% Two worst individuals
W1 = zeros(1, chromLength + 1);
W2 = zeros(1, chromLength + 1);

initialPop = zeros(pop, chromLength + 1);
offsprings = zeros(2, chromLength + 1);

% GENERATE INITIAL POPULATION (Generation 1)
for i = 1:pop
    initialPop(i,1:chromLength) = randperm(chromLength);
    initialPop(i,chromLength + 1) =
fitness(initialPop(i,1:chromLength));
    while initialPop(i,chromLength + 1) > 99999999999999
        initialPop(i,1:chromLength) = randperm(chromLength);
        initialPop(i,chromLength + 1) =
fitness(initialPop(i,1:chromLength));
    end
end

gen = 1;
convergCount = 0;

% BIG LOOP
while convergCount < convergLimit
    bestTC(:,:,gen) = min(initialPop(:, chromLength + 1, gen));
    for i = 1:pop
        if initialPop(i, chromLength + 1, gen) == bestTC(:,:,gen)
            bestSolution(:,:,gen) = initialPop(i, 1:chromLength, gen);
        end
    end
    if bestTC(:,:,gen) < TC
        TC = bestTC(:,:,gen);
        solutionArray = bestSolution(:,:,gen);
    end

    if bestTC(:,:,gen) >= min(bestTC(:,:,1:gen-1))
        convergCount = convergCount + 1;
    else
        convergCount = 0;
    end

    % Selecting parents
    best = mink(initialPop(:,chromLength + 1, gen),2);
    for i = 1:pop
        if initialPop(i, chromLength + 1, gen) == best(1) ||
initialPop(i, chromLength + 1, gen) == best(2)
            S1(1,:,gen) = initialPop(i,:, gen);
            for j = i+1:pop
                if initialPop(j, chromLength + 1, gen) == best(1) ||
initialPop(j, chromLength + 1, gen) == best(2)
```

```
                    S2(1,:,gen) = initialPop(j,:, gen);
                    break
                end
            end
            break
        end
    end

    worst = maxk(initialPop(:,chromLength + 1, gen),2);
    for i = 1:pop
        if initialPop(i, chromLength + 1, gen) == worst(1) ||
initialPop(i, chromLength + 1, gen) == worst(2)
            W1(1,:,gen) = initialPop(i,:, gen);
            for j = i+1:pop
                if initialPop(j, chromLength + 1, gen) == worst(1) ||
initialPop(j, chromLength + 1, gen) == worst(2)
                    W2(1,:,gen) = initialPop(j,:, gen);
                    break
                end
            end
            break
        end
    end

    % Crossover: choose a random crossover method. If offspring =
parent, use another method

    coMethod(gen) = rand;
    if coMethod < 0.5
        [offsprings(1,1:chromLength,gen),
offsprings(2,1:chromLength,gen)] =
crossover1(S1(1,1:chromLength,gen),S2(1,1:chromLength,gen));
        offsprings(1,chromLength + 1,gen) =
fitness(offsprings(1,1:chromLength,gen));
        offsprings(2,chromLength + 1,gen) =
fitness(offsprings(2,1:chromLength,gen));
        if isequal(offsprings(1,:,gen), S1(1,:,gen)) == 1 ||
isequal(offsprings(1,:,gen),S2(1,:,gen)) == 1
            [offsprings(1,1:chromLength,gen),
offsprings(2,1:chromLength,gen)] =
crossover2(S1(1,1:chromLength,gen),S2(1,1:chromLength,gen));
        offsprings(1,chromLength + 1,gen) =
fitness(offsprings(1,1:chromLength,gen));
        offsprings(2,chromLength + 1,gen) =
fitness(offsprings(2,1:chromLength,gen));
        end
    else
        [offsprings(1,1:chromLength,gen),
offsprings(2,1:chromLength,gen)] =
crossover2(S1(1,1:chromLength,gen),S2(1,1:chromLength,gen));
        offsprings(1,chromLength + 1,gen) =
fitness(offsprings(1,1:chromLength,gen));
        offsprings(2,chromLength + 1,gen) =
fitness(offsprings(2,1:chromLength,gen));

        if isequal(offsprings(1,:,gen), S1(1,:,gen)) == 1 ||
isequal(offsprings(1,:,gen),S2(1,:,gen)) == 1
```

```
            [offsprings(1,1:chromLength,gen),
offsprings(2,1:chromLength,gen)] =
crossover1(S1(1,1:chromLength,gen),S2(1,1:chromLength,gen));
            offsprings(1,chromLength + 1,gen) =
fitness(offsprings(1,1:chromLength,gen));
            offsprings(2,chromLength + 1,gen) =
fitness(offsprings(2,1:chromLength,gen));
        end
    end

    % Replace the 2 worst individual with new offspring, create new
generation
    gen = gen + 1;
    initialPop(:,:,gen) = initialPop(:,:,gen - 1);
    for i = 1:pop
        if initialPop(i, chromLength + 1, gen) == worst(1) ||
initialPop(i, chromLength + 1, gen) == worst(2)
            initialPop(i,:, gen) = offsprings(1,:,gen - 1);
            for j = i+1:pop
                if initialPop(j, chromLength + 1, gen) == worst(1) ||
initialPop(j, chromLength + 1, gen) == worst(2)
                    initialPop(j,:, gen) = offsprings(2,:,gen - 1);
                    break
                end
            end
            break
        end
    end

    % Mutation
    mutant(:,:,1) = 0;
    if rand < mutationProba
        mutant(:,:,gen) = ceil(1 + rand*(pop - 1));
        while mutant(:,:,gen) ==
findIndex(transpose(initialPop(:,chromLength +
1,gen)),fitness(solutionArray))
            mutant(:,:,gen) = ceil(1 + rand*(pop - 1));
        end
        initialPop(mutant(:,:,gen), 1:chromLength, gen) =
mutation(initialPop(mutant(:,:,gen), 1:chromLength, gen));
        initialPop(mutant(:,:,gen), chromLength + 1, gen) =
fitness(initialPop(mutant(:,:,gen), 1:chromLength, gen));
    else
        mutant(:,:,gen) = 0;
    end

end

% SOLUTION ANALYSIS
    breakPoint = 0;
    index = 1;
    % Assign name and route
    for m = 1:length(solutionArray)
        if solutionArray(m) > noOfCustomers
            vehicleRoutes(index).Name = strcat('Vehicle ',
num2str(index));
```

```
            vehicleRoutes(index).Nodes = solutionArray(breakPoint + 1
:m-1);
            breakPoint = m;
            index = index + 1;
        end
        if m == length(solutionArray)
            vehicleRoutes(index).Name = strcat('Vehicle ',
num2str(index));
            vehicleRoutes(index).Nodes = solutionArray(breakPoint + 1 :
m);
        end
    end


    for i = 1:noOfVehicles
        if isempty(vehicleRoutes(i).Nodes) == 1
            continue
        end

    % Assign starting time + departure time + arrival time
        for j = 1 : length(vehicleRoutes(i).Nodes)
            % Beginning
            if j == 1
                vehicleRoutes(i).startingTime = 0;
                vehicleRoutes(i).departureTime(1) =
vehicleRoutes(i).startingTime;
                vehicleRoutes(i).arrivalTime(1) =
vehicleRoutes(i).startingTime + travelTime(1 , vehicleRoutes(i).Nodes(1)
+ 1);
            else
                vehicleRoutes(i).departureTime(j) =
max(timeWindow(vehicleRoutes(i).Nodes(j-1),1),...
            vehicleRoutes(i).arrivalTime(j-1));
                vehicleRoutes(i).arrivalTime(j) =
vehicleRoutes(i).departureTime(j) + travelTime(vehicleRoutes(i).Nodes(j-
1) + 1,vehicleRoutes(i).Nodes(j) + 1);
            end

            % Return to depot
            if j == length(vehicleRoutes(i).Nodes)
                vehicleRoutes(i).departureTime(j+1) =
max(timeWindow(vehicleRoutes(i).Nodes(j),1),...
        vehicleRoutes(i).arrivalTime(j));
                vehicleRoutes(i).arrivalTime(j+1) =
vehicleRoutes(i).departureTime(j+1) +
travelTime(vehicleRoutes(i).Nodes(j) + 1, 1);
            end
        end

    % Assign carried load before each nodes
        vehicleRoutes(i).carriedLoad(1) = 0;
        for j = 1:length(vehicleRoutes(i).Nodes)
            vehicleRoutes(i).carriedLoad(1) =
vehicleRoutes(i).carriedLoad(1) + demand(vehicleRoutes(i).Nodes(j),1);
        end

        for j = 2:length(vehicleRoutes(i).Nodes)+1
```

```
            vehicleRoutes(i).carriedLoad(j) =
vehicleRoutes(i).carriedLoad(j-1) - demand(vehicleRoutes(i).Nodes(j-
1),1);
        end

    % Assign distance traveled
        for j = 1:length(vehicleRoutes(i).Nodes) + 1
            if j == 1
                vehicleRoutes(i).distanceTraveled(j) = distance(1,
vehicleRoutes(i).Nodes(j) + 1);
                continue
            elseif j == length(vehicleRoutes(i).Nodes) + 1
                vehicleRoutes(i).distanceTraveled(j) =
distance(vehicleRoutes(i).Nodes(j-1) + 1, 1);
                continue
            else
                vehicleRoutes(i).distanceTraveled(j) =
distance(vehicleRoutes(i).Nodes(j-1) +1, vehicleRoutes(i).Nodes(j) +1);
            end
        end

    % Assign travel time
        for j = 1:length(vehicleRoutes(i).departureTime)
            vehicleRoutes(i).timeTraveled(j) =
vehicleRoutes(i).arrivalTime(j) - vehicleRoutes(i).departureTime(j);
        end
    end
toc

fprintf('Total distance traveled (in km) is %4.2f\n', TC);
disp('For more detailed solution, inspect the variable
"vehicleRoutes"');

a = 0;
b = 0;
for i = 1:size(bestSolution,3)
a(i,1) = fitness(bestSolution(:,:,i));
end
figure
plot(a, 'LineWidth', 1)
```

8.2.2. Other functions

Fitness

```
function fitnessScore = fitness(solutionArray)
    global noOfCustomers
    global distance
    global travelTime
    global demand
    global timeWindow
    global capacity

    breakPoint = 0;
    index = 1;
```

```
    noOfVehicles = length(solutionArray) - noOfCustomers + 1;

% Solution analysis
    % Assign name and route
    for m = 1:length(solutionArray)
        if solutionArray(m) > noOfCustomers
            vehicleRoutes(index).Name = strcat('Vehicle ',
num2str(index));
            vehicleRoutes(index).Nodes = solutionArray(breakPoint + 1
:m-1);
            breakPoint = m;
            index = index + 1;
        end
        if m == length(solutionArray)
            vehicleRoutes(index).Name = strcat('Vehicle ',
num2str(index));
            vehicleRoutes(index).Nodes = solutionArray(breakPoint + 1 :
m);
        end
    end


    for i = 1:noOfVehicles
        if isempty(vehicleRoutes(i).Nodes) == 1
            continue
        end

    % Assign starting time + departure time + arrival time
        for j = 1 : length(vehicleRoutes(i).Nodes)
            % Beginning
            if j == 1
                vehicleRoutes(i).startingTime = 0;
                vehicleRoutes(i).departureTime(1) =
vehicleRoutes(i).startingTime;
                vehicleRoutes(i).arrivalTime(1) =
vehicleRoutes(i).startingTime + travelTime(1 , vehicleRoutes(i).Nodes(1)
+ 1);
            else
                vehicleRoutes(i).departureTime(j) =
max(timeWindow(vehicleRoutes(i).Nodes(j-1),1),
vehicleRoutes(i).arrivalTime(j-1));
                vehicleRoutes(i).arrivalTime(j) =
vehicleRoutes(i).departureTime(j) + travelTime(vehicleRoutes(i).Nodes(j-
1) + 1,vehicleRoutes(i).Nodes(j) + 1);
            end

            % Return to depot
            if j == length(vehicleRoutes(i).Nodes)
                vehicleRoutes(i).departureTime(j+1) =
max(timeWindow(vehicleRoutes(i).Nodes(j),1),
vehicleRoutes(i).arrivalTime(j));
                vehicleRoutes(i).arrivalTime(j+1) =
vehicleRoutes(i).departureTime(j+1) +
travelTime(vehicleRoutes(i).Nodes(j) + 1, 1);
            end
        end
```

```
    % Assign carried load before each nodes
        vehicleRoutes(i).carriedLoad(1) = 0;
        for j = 1:length(vehicleRoutes(i).Nodes)
            vehicleRoutes(i).carriedLoad(1) =
vehicleRoutes(i).carriedLoad(1) + demand(vehicleRoutes(i).Nodes(j),1);
        end

        for j = 2:length(vehicleRoutes(i).Nodes)+1
            vehicleRoutes(i).carriedLoad(j) =
vehicleRoutes(i).carriedLoad(j-1) - demand(vehicleRoutes(i).Nodes(j-
1),1);
        end

    % Assign distance traveled
        for j = 1:length(vehicleRoutes(i).Nodes) + 1
            if j == 1
                vehicleRoutes(i).distanceTraveled(j) = distance(1,
vehicleRoutes(i).Nodes(j) + 1);
                continue
            elseif j == length(vehicleRoutes(i).Nodes) + 1
                vehicleRoutes(i).distanceTraveled(j) =
distance(vehicleRoutes(i).Nodes(j-1) + 1, 1);
                continue
            else
                vehicleRoutes(i).distanceTraveled(j) =
distance(vehicleRoutes(i).Nodes(j-1) +1, vehicleRoutes(i).Nodes(j) +1);
            end
        end

    % Assign travel time
        for j = 1:length(vehicleRoutes(i).departureTime)
            vehicleRoutes(i).timeTraveled(j) =
vehicleRoutes(i).arrivalTime(j) - vehicleRoutes(i).departureTime(j);
        end
    end
% Total distance calculation
    fitnessScore = 0;
    for i = 1:noOfVehicles
        for j = 1:length(vehicleRoutes(i).distanceTraveled)
            fitnessScore = fitnessScore +
vehicleRoutes(i).distanceTraveled(j);
        end
    end

% Feasibility
    for i = 1:noOfVehicles
        for j = 1:length(vehicleRoutes(i).Nodes)
            if vehicleRoutes(i).arrivalTime(j) >
timeWindow(vehicleRoutes(i).Nodes(j),2)
                fitnessScore = 999999999999999;
            end
        end

        if max(vehicleRoutes(i).carriedLoad) > capacity
            fitnessScore = 999999999999999;
        end
    end
```

```
end
```

Crossover 1

```
function [child1, child2] = crossover1(p1, p2)
    cycle = zeros(1,length(p1));
    child1 = zeros(1,length(p1));
    child2 = zeros(1,length(p1));
    cycleNumber = 1;
    while length(find(cycle)) < length(cycle)
        index = findIndex(cycle,0);
        cycle(index(1)) = cycleNumber;
        valueOnP1 = p1(index(1));
        firstValue = valueOnP1;
        valueOnP2 = p2(index(1));
        while firstValue ~= valueOnP2
            cycle(findIndex(p1,valueOnP2)) = cycleNumber;
            valueOnP2 = p2(findIndex(p1, valueOnP2));
            valueOnP1 = p1(findIndex(p2, valueOnP2));
        end
        cycleNumber = cycleNumber + 1;
    end
    for i = 1:length(p1)
        if mod(cycle(i),2) == 0
            child1(i) = p1(i);
            child2(i) = p2(i);
        else
            child1(i) = p2(i);
            child2(i) = p1(i);
        end
    end
end
```

Crossover 2

```
function [child1, child2] = crossover2(p1, p2)
    child1 = zeros(1,length(p1));
    child2 = zeros(1,length(p1));
    a = ceil(rand*length(p1));
    b = ceil(rand*length(p1));
    while b == a
            b = ceil(rand*length(p1));
    end

    % Identify 2 cut points
    cp1= min(a,b);
    cp2 = max(a,b);

    child1(cp1:cp2) = p1(cp1:cp2);
    for i = 1:length(child1)
        if child1(i) == 0
            for j = 1:length(p2)
                if isempty(findIndex(child1,p2(j))) == 1
                    child1(i) = p2(j);
```

```
                    break
                end
            end
        end
    end

    child2(cp1:cp2) = p2(cp1:cp2);
    for i = 1:length(child2)
        if child2(i) == 0
            for j = 1:length(p1)
                if isempty(findIndex(child2,p1(j))) == 1
                    child2(i) = p1(j);
                    break
                end
            end
        end
    end

end
```

Find index

```
function index = findIndex(array, value)
    index = find(array < value + 1 & array > value - 1);
end
```

Mutation

```
function mutant = mutation(array)
    a = ceil(rand*length(array));
    b = ceil(rand*length(array));
    while b == a
            b = ceil(rand*length(array));
    end
    % Identify 2 cut points
    cp1 = min(a,b);
    cp2 = max(a,b);

    i = randperm(cp2 - cp1 + 1);
    mutant = array;

    mutant(cp1:cp2) = array(:,i + cp1 - 1);
end
```

Plot result

```
a = 0;
b = 0;
for i = 1:size(bestSolution,3)
a(i,1) = fitness(bestSolution(:,:,i));
end
figure
plot(a, 'LineWidth', 1)
```

8.3. Data collection

Table 8 - 1: List of customer

	No.	Customer	Location
1	1	Công ty TNHH TM & DV Hạnh Phú Hòa	Số 89, Khu phố Phú Hòa, Phường Hòa Lợi, Bến Cát, Bình Dương
	2	Công ty TNHH XD – TM – DV Trung Kiên Phát	Số 135, Bàu Bàng, Lai Uyên, Bàu Bàng, Bình Dương
	3	Cửa Hàng Lệ Hoa	ĐT741, Vĩnh Hoà, Phú Giáo, Bình Dương
	4	Cửa Hàng Vật Tư Cao Su Xuân Hồng	ĐT741, Phước Hoà, Phú Giáo, Bình Dương
	5	Cửa Hàng Năm Chiếm	An Bình, Phú Giáo, Bình Dương
	6	Cửa Hàng Kim Quy	111 Ngô Nhân Tịnh, Phường 13, Quận 5, TP.HCM
	7	Cửa Hàng Hiền	130 Phạm Hữu Chí, Phường 15, Quận 5, TP.HCM
	8	Bến xe Miền Đông	292 Đinh Bộ Lĩnh, phường 26, Quận Bình Thạnh
	9	Bến xe Miền Tây	395 Kinh Dương Vương, Phường An Lạc, Quận Bình Tân
	10	Chành xe Hồng Lợi Nam	203 An Dương Vương, Phường An Lạc, Quận Bình Tân, TP.HCM
	11	Cửa Hàng Quý	Ngã tư Chơn Thành, Chơn Thành, Bình Phước
	12	Cửa Hàng Hồng Quang	Khu Đức Lập, Thị trấn Đức Phong Huyện Bù Đăng, Đức Phong, Bù Đăng, Bình Phước
	13	Cửa Hàng Hoàng Dung	88/3, KP 2, Thị Trấn Chơn Thành, Huyện Chơn Thành, Chơn Thành, Bình Phước
	14	Cửa Hàng Hưng Thịnh	Ngã ba Cây Điệp, ĐT743B, Tân Đông Hiệp, Thị Xã Dĩ An, Bình Dương

	15	Cửa Hàng Quang Phú	54 Đường số 1, Tân Bình, Đồng Xoài, Bình Phước
2	1	Cửa Hàng Kim Ngân	KP. Nhơn Hậu 1, P. Tân Khánh, TP. Long An
	2	Hợp Tác Xã Trường Phát	69, Ấp 3, Xã Hưng Hòa, Huyện Bến Cát, Tân Hưng, Bến Cát, Bình Dương
	3	Công ty TNHH MTV SX TM DV Nông Nghiệp Trung Hà	Số 253, Đường ĐH 312, Thôn Phú Cường., Xã Phú Riềng, Huyện Phú Riềng, Tỉnh Bình Phước.
	4	Công Ty TNHH SX TM DV Thắng Thành Lợi	QL 13, Tổ 1, ấp 1, Xã Thành Tâm, Huyện Chơn Thành, Tỉnh Bình Phước.
	5	Công Ty TNHH MTV XD TM Hảo Anh	Phú Hưng, Xã Phú Riềng, Huyện Phú Riềng, Bình Phước
	6	Hợp Tác Xã Sản Xuất Nông Lâm Dịch Vụ Tân Thành	Số 37, ấp Tân Phong, Xã Tân Thành, Huyện Bù Đốp, Bình Phước.
	7	CÔNG TY TNHH MTV TM DV Thanh Tú	219 Phú Riềng Đỏ, Khu phố Phú Thanh, Phường Tân Phú, Thị xã Đồng Xoài, Bình Phước
	8	Công Ty TNHH MTV XD TM DV Tân Hoàng Phương	Tổ 7, ấp Thanh Thiện, Xã Thanh Lương, Thị xã Bình Long, Bình Phước.
	9	HTX Nông Nghiệp - DV - TM Đồng Xê	ấp Đồng Xê, Xã Tân Hoà, Huyện Đồng Phú, Bình Phước
	10	Cửa Hàng Hiền Lệ	Thôn Bình Hiếu, Xã Bình Tân, Huyện Bù Gia Mập, Tỉnh Bình Phước
	11	Cửa Hàng Diệu Kim	ĐT312, Thống Nhất, Bù Đăng, Bình Phước
	12	Cửa Hàng Hưng Thịnh Phát	Tỉnh lộ 830, ấp 4, xã lương bình, Lương Bình, Bến Lức, Long An
	13	Công Ty TNHH MTV TM – DV – SX Khánh An	Hoà Khánh Nam, Đức Hòa, Long An

3	1	Cửa Hàng Kim Hải Nam	3C Đường 3 Tháng 2, Phường 11, Quận 10, Hồ Chí Minh
	2	Cửa Hàng Quyết Thắng	91/6 Hoà Hưng, Phường 12, Quận 10, Hồ Chí Minh
	3	Công Ty TNHH SX – TM – DV Tân Thanh	354/30 Lý Thường Kiệt, Phường 14, Quận 10, Hồ Chí Minh
	4	Cửa Hàng Bảo Lợi	205 Đào Duy Từ, Phường 6, Quận 10
	5	Cửa Hàng Kim Dung	416 Đường Hậu Giang, Phường 12, Quận 6, Hồ Chí Minh 700000, Việt Nam
	6	Cửa Hàng Vĩnh Phúc	227 Bình Tiên, Phường 8, Quận 6, Hồ Chí Minh
	7	Cửa Hàng Lê Hùng	183 Tân Hòa Đông, Phường 14, Quận 6, Hồ Chí Minh
	8	Cửa Hàng Tân Xuân Hoàng	245/2 Lý Thường Kiệt, Phường 15, Quận 11, Hồ Chí Minh, Việt Nam
	9	Công Ty TNHH Vật Tư Tổng Hợp Tân Ái	479, Đường Lạc Long Quân, Phường 5, Quận 11, Hồ Chí Minh
	10	Công Ty TNHH TM – DV – SX Doanh Phát	279 Đường Hàn Hải Nguyên, Phường 2, Quận 11, Hồ Chí Minh
4	1	Cửa Hàng Kim Hải Nam	554 Hà Huy Giáp, Thạnh Lộc, Quận 12, Hồ Chí Minh
	2	Cửa Hàng Thiên Quốc	Số 122 Đường TX22, Khu Phố 5, Phường Thạnh Xuân, Thạnh Xuân, Quận 12, Hồ Chí Minh
	3	Cửa Hàng Quốc Hiếu	314 Lê Thị Riêng, Tân Thới An, Quận 12, Hồ Chí Minh
	4	Cửa Hàng Tân Diệu Hòa	3D QL1A, An Phú Đông, Quận 12, Hồ Chí Minh
	5	Cửa Hàng Mã Xuân Quang	465 Nguyễn Ảnh Thủ, Phường Hiệp Thành, Quận 12, Thành Phố Hồ Chí Minh

	6	Cửa Hàng Trung Hà	1018 Tô Ký, Tân Chánh Hiệp, Quận 12, Hồ Chí Minh
	7	Cửa Hàng Thủy	88D, Khu Phố 2, Phường Thanh Xuân, Quận 12
	8	Cửa Hàng Hòa Phát	250A Đường TX52, Phường, Thạnh Xuân, Quận 12, Hồ Chí Minh
	9	Cửa Hàng Diệp Hồng	183/14 khu phố 4, Tân Chánh Hiệp, Quận 12, Hồ Chí Minh
	10	Cửa Hàng Hoàng Phúc Thái	Đường D6, Hiệp Thành, Quận 12, Hồ Chí Minh
	11	Cửa Hàng Quang Dũng	Lô C, Cụm Công Nghiệp Quang Trung Phường Thành Phố, Hiệp Thành, Quận 12
	12	Cửa Hàng Ánh Kim	3/5 HT19, Phường Hiệp Thành, Quận 12, Tân Thới An, Quận 12, Hồ Chí Minh

o

Table 8 - 2: Distance matrix (Group 1)

	0	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**C11**	**C12**	**C13**	**C14**	**C15**
0	0	40	56	60	52	69	16	15	9	19	20	75	150	73	15	95
C1	40	0	26	25	19	36	49	48	43	52	53	45	116	43	31	61
C2	56	26	0	23	17	34	67	66	62	71	72	19	108	17	50	52
C3	60	25	23	0	11	10	74	73	64	78	78	39	91	37	52	36
C4	52	19	17	11	0	17	65	64	55	69	70	30	98	28	44	43
C5	69	36	34	10	17	0	81	80	71	85	85	37	81	40	59	26
C6	16	49	67	74	65	81	0	0.75	11	6	5	86	160	85	29	105
C7	15	48	66	73	64	80	0.75	0	10	5	6	86	160	84	25	105
C8	9	43	62	64	55	71	11	10	0	15	17	82	153	80	16	98
C9	19	52	71	78	69	85	6	5	15	0	3	91	165	89	29	107
C10	20	53	72	78	70	85	5	6	17	3	0	92	166	90	35	111
C11	75	45	19	39	30	37	86	86	82	91	92	0	91	3	70	36
C12	150	116	108	91	98	81	160	160	153	165	166	91	0	94	140	56
C13	73	43	17	37	28	40	84	84	80	89	90	3	94	0	67	38
C14	15	31	50	52	44	59	25	25	16	29	35	70	140	67	0	84
C15	95	61	52	36	43	26	105	105	98	107	111	36	56	38	84	0

Table 8 - 3: Travel time matrix (Group 1)

	0	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**C11**	**C12**	**C13**	**C14**	**C15**
0	0.00	1.00	1.40	1.50	1.30	1.73	0.40	0.38	0.23	0.48	0.50	1.88	3.75	1.83	0.38	2.38
C1	1.00	0.00	0.65	0.63	0.48	0.90	1.23	1.20	1.08	1.30	1.33	1.13	2.90	1.08	0.78	1.53
C2	1.40	0.65	0.00	0.58	0.43	0.85	1.68	1.65	1.55	1.78	1.80	0.48	2.70	0.43	1.25	1.30
C3	1.50	0.63	0.58	0.00	0.28	0.25	1.85	1.83	1.60	1.95	1.95	0.98	2.28	0.93	1.30	0.90
C4	1.30	0.48	0.43	0.28	0.00	0.43	1.63	1.60	1.38	1.73	1.75	0.75	2.45	0.70	1.10	1.08
C5	1.73	0.90	0.85	0.25	0.43	0.00	2.03	2.00	1.78	2.13	2.13	0.93	2.03	1.00	1.48	0.65
C6	0.40	1.23	1.68	1.85	1.63	2.03	0.00	0.02	0.28	0.15	0.13	2.15	4.00	2.13	0.73	2.63
C7	0.38	1.20	1.65	1.83	1.60	2.00	0.02	0.00	0.25	0.13	0.15	2.15	4.00	2.10	0.63	2.63
C8	0.23	1.08	1.55	1.60	1.38	1.78	0.28	0.25	0.00	0.38	0.43	2.05	3.83	2.00	0.40	2.45
C9	0.48	1.30	1.78	1.95	1.73	2.13	0.15	0.13	0.38	0.00	0.08	2.28	4.13	2.23	0.73	2.68
C10	0.50	1.33	1.80	1.95	1.75	2.13	0.13	0.15	0.43	0.08	0.00	2.30	4.15	2.25	0.88	2.78
C11	1.88	1.13	0.48	0.98	0.75	0.93	2.15	2.15	2.05	2.28	2.30	0.00	2.28	0.08	1.75	0.90
C12	3.75	2.90	2.70	2.28	2.45	2.03	4.00	4.00	3.83	4.13	4.15	2.28	0.00	2.35	3.50	1.40
C13	1.83	1.08	0.43	0.93	0.70	1.00	2.13	2.10	2.00	2.23	2.25	0.08	2.35	0.00	1.68	0.95
C14	0.38	0.78	1.25	1.30	1.10	1.48	0.73	0.63	0.40	0.73	0.88	1.75	3.50	1.68	0.00	2.10
C15	2.38	1.53	1.30	0.90	1.08	0.65	2.63	2.63	2.45	2.68	2.78	0.90	1.40	0.95	2.10	0.00

Table 8 - 4: Demand and time windows of Group 1

No.	Demand (kg)	Lower time (hr)	Upper time (hr)
0	0	0	24
1	380	9	16
2	360	9	16
3	497	9	16
4	400	9	16
5	500	13	16
6	585	10	16
7	600	11	16
8	394	9	15
9	470	9	15
10	400	9	15
11	482	9	15
12	350	10	15
13	358	10	15
14	368	11	16
15	471	9	16
TOTAL	6,615		

Table 8 - 5: Demand and time windows of Group 2

No.	Demand (kg)	Lower time (hr)	Upper time (hr)
0	0	0	24
1	543	9	16
2	600	9	12
3	550	9	15
4	400	9	15
5	509	10	15
6	567	10	16
7	489	9	16
8	541	9	16
9	520	9	14
10	418	10	16
11	500	10	16
12	477	9	16
13	444	9	16
TOTAL	6,558		

Table 8 - 6: Demand and time windows of Group 3

No.	Demand (kg)	Lower time (hr)	Upper time (hr)
0	0	0	24
1	600	9	16
2	635	9	16
3	684	9	16
4	553	9	16
5	650	9	16
6	569	9	15
7	582	10	15
8	632	10	14
9	671	11	16
10	700	9	15
TOTAL	6,276		

Table 8 - 7: Demand and time windows of Group 4

No.	Demand (kg)	Lower time (hr)	Upper time (hr)
0	0	0	24
1	486	9	16
2	575	9	16
3	600	9	16
4	471	9	16
5	573	9	15
6	450	10	14
7	555	10	14
8	539	10	15
9	494	11	16
10	416	9	16
11	500	9	13
12	400	9	16
TOTAL	6,059		

8.4.2. Result of using MATLAB

Table 8 - 9: Routing, travelling distance, demand and total cost of using MATLAB

Group	Serving nodes	Travelling distance (km)	Demand (kg)	Total cost (VND)
1	[Depot, C5,C11,C13, Depot]	182	1,340	1,519,000
	[Depot, C1, C15, C12, Depot]	307	1,201	1,995,000
	[Depot, C8, C7, C9, Depot]	43	1,464	172,000
	[Depot, C2, C3, C4, Depot]	142	1,257	1,176,000
	[Depot, C14, C6, C10, Depot]	69	1,353	249,000
2	[Depot, C9, C2, Depot]	189	1,120	994,000
	[Depot, C10, C6, C7, Depot]	252	1,474	2,366,000
	[Depot, C5, C11, Depot]	269	1,009	1,708,000
	[Depot, C12, C13, C1, Depot]	129	1,464	931,000
	[Depot, C3, C8, C4, Depot]	268	1,491	2,121,000
3	[Depot, C5, C6, Depot]	34.15	1,219	132,800
	[Depot, C4, C7, Depot]	35.8	1,135	124,400
	[Depot, C1, C2, Depot]	24.3	1,235	93,200
	[Depot, C9, C10, Depot]	27.6	1,371	105,600
	[Depot, C3, C8, Depot]	25.6	1,316	98,000
4	[Depot, C11, C3, Depot]	19.1	1,100	552,000
	[Depot, C2, C8, Depot]	8.8	1114	32,800
	[Depot, C10, C7, C6, Depot]	20.4	1,421	113,600
	[Depot, C9, C12, C4, Depot]	22.6	1,365	96,000

	[Depot, C1, C5, Depot]	19.2	1,059	46,400