Grey Wolf Optimizer algorithm for solving the multi depot vehicle routing problem and its implementation

D Diastivena, S Wahyuningsih, and D Satyananda *)
Department of Mathematics, Universitas Negeri Malang, Indonesia

*Corresponding author: darmawan.satyananda.fmipa@um.ac.id
E-mail: diastivena@gmail.com, sapti.wahyuningsih.fmipa@um.ac.id

Abstract. Multi Depot Vehicle Routing Problem (MDVRP) is one of Vehicle Routing Problem (VRP) variants. MDVRP is a VRP that uses more than one depot for the distribution process. In this paper, the MDVRP will be solved using the Grey Wolf Optimizer (GWO). This algorithm is inspired by hunting technique and hierarchy of grey wolves. The completion of MDVRP consists of two stages, they are grouping and routing. In the grouping stage, customers are grouped to nearest depot. In the routing stage, the route is determined at each depot using “route-first, cluster-second”. In the route-first phase a giant tour is formed by forming population array. In the cluster-second phase, a split will be conducted on the giant tour, so that a number of routes are formed with the total demand for each route does not exceed the vehicle capacity. The implementation of GWO algorithm for MDVRP was designed in Borland Delphi 7.0 programming language. The algorithm was tested using 2 depots and 9 customers formed 2 routes at each depot with a total distance of 833 km. Another test was carried out by comparing the GWO with the ACO using the Cordeau dataset. GWO gives the best result (shortest distance) than ACO.

1. Introduction
Distribution is an activity of sending goods to consumers. The length of the distribution route affects the total distribution cost, where the shorter the route, the smaller the costs incurred, and vice versa. Hence it needs the right design to optimize the distribution process. One application of graph theory that can be used in solving optimization problems is the Vehicle Routing Problem (VRP). The VRP problem is the search for the optimal vehicle route provided that it starts and ends at the same location [3]. VRP problems are developed into various variants, one of them is the Multi Depot Vehicle Routing Problem (MDVRP). According to [13] MDVRP is an expansion of the VRP problem, that determines routes to serve a group of customers from several depots, and each route starts and ends at the same depot without violating vehicle capacity constraints. Basically, it is set of multiple VRP (or CVRP, exactly).

MDVRP is a VRP problem with more than one depot, namely the depot j ∈ {0, 1, ..., n}. The objective of MDVRP is to find a minimum number of routes for each depot with vehicles departing and returning to the same depot. Each customer i ∈ {n + 1, n + 2, ..., n + m} is served exactly once by a vehicle, where n is the last depot and m represents the number of customers. Total customer demands on each route must not exceed the vehicle capacity, with each vehicle is assumed having the same capacity. According to [13] MDVRP completion consists of two stages: grouping and routing. At the grouping stage, customers are grouped into the depot with the closest distance. Hence from the grouping stage solution, the MDVRP problem will become a VRP problem for each depot. In [8], a "route-first,
“cluster-second” strategy was developed in solving VRP problems. In the route-first phase, a giant tour will be formed, while in the cluster-second phase there will be a split on the giant tour by taking into account capacity constraints and customer demand.

One of the methods used in solving MDVRP problems is metaheuristic. One of the metaheuristic algorithms that has been used in solving MDVRP problems is the Greedy Randomized Adaptive Search Procedure (GRASP) [10]. This algorithm forms an initial solution by randomly selecting customers based on the Restricted Candidate List (RCL). The GRASP algorithm has been reviewed by [14] in finding a solution to the MDVRP problem by combining the Variable Neighborhood Descent (VND) algorithm. In addition, the Ant Colony Optimization (ACO) algorithm has also been applied in the search for solutions to the MDVRP problem studied by [2].

Another algorithm which is one of the metaheuristic methods is the Grey Wolf Optimizer (GWO) algorithm. The first GWO algorithm introduced by [7]. This algorithm is inspired by the hunting method and the gray wolf hierarchy. There are four stages of behavior of the gray wolf in hunting, namely chasing, surrounding prey, hunting and attacking prey. At the chasing stage, this algorithm states that α wolf to be the best solution, β wolf as the second-best solution, and δ wolf as the third best solution. At the stage of circling the prey, the gray wolf limits the prey by calculating the distance between the prey. Furthermore, the hunting stage is led by α wolves, while β and δ wolves contribute to helping α. In the last stage, the gray wolf will attack the prey, which means a solution has been found. The GWO algorithm has been applied in solving optimization problems, such as the Traveling Salesman Problem (TSP) by [11]. Shaheen et al. explained that the GWO algorithm solves TSP problems with good quality solutions and computational processes, because the search for solutions is done by forming possible solutions globally. The GWO algorithm is also used in solving the Capacitated Vehicle Routing Problem (CVRP) which has been studied by [5].

In addition to optimization problems, the GWO algorithm has also been applied to other problems, including the parameter estimation of surface waves [12]. According to Song et al. the application of the GWO algorithm can show a good balance and very fast convergence. They also explained that the GWO algorithm is an algorithm that is simple, flexible, and easy to implement. The GWO algorithm was also reviewed in [15] about economic delivery, presented that GWO provides very competitive results in terms of reducing transmission losses and minimizing total fuel costs. In other area, [4] implemented GWO for feature selection problem, for choosing the correct dataset in a certain situation, and in [9] that explained about forecasting of university students’ weaknesses using neural network.

Based on the description above, this paper introduces the GWO algorithm in solving MDVRP problems and implementing the program. The implementation of the GWO algorithm for MDVRP is designed in the Borland Delphi 7.0 programming language. The program will be tested using the Cordeau dataset. The solution obtained will be compared with the ACO algorithm to see the performance of the GWO algorithm in solving MDVRP problems, where both algorithms are metaheuristic algorithms.

2. Research Method
This paper applies the GWO algorithm for MDVRP problem and is implemented in a computer program designed using the Borland Delphi 7.0 programming language running on a computer with an AMD A9-9425 Radeon R5 processor, with 4.00 GB RAM, and operating system Microsoft Windows 10. To test its performance used 4 datasets. The dataset is available from https://neo.lcc.uma.es/vrp/vrp-instances/multiple-depot-vrp-instances/. The properties of each data set used are presented in Table 1.

| Dataset | Number of depot | Number of customer | Capacity of vehicle |
|---------|-----------------|--------------------|---------------------|
| P01     | 4               | 50                 | 80                  |
| P06     | 3               | 100                | 100                 |

Table 1. Properties of each dataset.
To see the quality of the resulting solution, it will be compared with other metaheuristic algorithms, namely Ant Colony Optimization (ACO) algorithm, which has been studied by Demirel.

### 3. Description of Proposed GWO Algorithm for MDVRP

The solution to MDVRP problems using the GWO algorithm consists of 3 stages. The first stage is grouping. At the grouping stage, each customer is grouped into the depot with the closest distance. The second stage is the formation of a giant tour. In the formation of the giant tour, it is done by forming population arrays. The population array is formed by entering the depot (prey) point as the initial population array. Next, choose 3 customer points (gray wolves) at random and input them into the initial population array. Then find $X_\alpha$, $X_\beta$, and $X_\delta$ in each population ($X_\alpha$, $X_\beta$, $X_\delta$ are the first, second, and third closest customer points from the last point in the population, respectively). Update the population by adding $X_\alpha$, $X_\beta$, and $X_\delta$ to the population array. If all customer points are loaded in the population arrays, add depot points to each population array and calculate the cost of each population array.

GWO uses the same principle a "route-first, cluster-second" strategy as proposed in [8]. The solution at the stage of forming a giant tour is the population array with the smallest cost. The last stage is routing. At the routing stage, a split is carried out on the giant tour that has been formed, so that a number of routes will be formed where the total demand on each route does not exceed the vehicle capacity. Figure 1 is a flow chart for MDVRP completion using the GWO algorithm.

![Grouping giant tour routing flowchart](image)

**Figure 1.** GWO-MDVRP completion flowchart.

The steps of the GWO algorithm in solving MDVRP problems are as follows.

**Step 1: grouping**

Group customers to the nearest depot. The grouping process is done based on the calculation of $d_{nd_i}$ and $d_{nd_j}$, which is the distance between the $n$th customer with the $i$-depot or the $j$-depot.

- If $d_{nd_i} < d_{nd_j}$, then customer $n$ is grouped into the $d_i$ depot
- If $d_{nd_i} > d_{nd_j}$, then customer $n$ is grouped into the $d_j$ depot
- If $d_{nd_i} = d_{nd_j}$, then customer $n$ is grouped into any depot, i.e. $d_i$ or $d_j$ depot.

**Step 2: giant tour formation**

- Initialization stage
  Determining population size, i.e. $n + 1$ with $n$ is the number of points (depot and customer). Nextadd a point as a depot to the population. Let $P$ be a depot (start point), so $P$ will be added to population array as Population $[0] = [P]$. Then select three customer points randomly and add as
initial population. For example, chosen points \(Q,R,\) and \(S\). After that, the population will be updated based on the three points that have been selected and two populations will be created. The total will be three populations as:

Population \([0] = [P,Q]\)
Population \([1] = [P,R]\)
Population \([2] = [P,S]\)

Population \([0]\) is the updated population, while population \([1]\) and population \([2]\) are the new population. In this case, Population \([0] = [P,Q]\) means that the population array 0 forms a tour from depot \(P\) to customer \(Q\). Likewise for population \([1]\) and population \([2]\). Then calculate the total distance (cost) of each population.

- **Iteration stage**
  Find \(X_\alpha,X_\beta,\) and \(X_\delta\) in each population (\(X_\alpha,X_\beta,X_\delta\) are the first, second, and third closest points of the last point in the population, respectively). Update population, i.e.
  Population \([i] = \text{Population} \ [i] + X_\alpha\)
  Population \([i + 1] = \text{Population} \ [i + 1] + X_\beta\)
  Population \([i + 2] = \text{Population} \ [i + 2] + X_\delta\)

Calculate the cost in each population array that has been formed. This process is carried out if the number of population (stored in array as array elements) does not exceed the population size. If the number of the population is more than population size, delete the population array that has the largest cost. How many elements are deleted depends on the population size (until the population array does not exceed the population size). The process is carried out until each population array contains all customers.

- **Final stage**
  Add a depot point to each population array. Calculate cost for all population arrays.

Output: population array with the smallest cost.

**Step 3: routing**
Algorithm 1. Split
1) Build a list of \(V_i\) with \(i \in 0, ..., n\), where \(n\) is the number of customer points in the depot \(d\).
2) Set \(V_0 = 0, P_0 = 0,\) and \(V_k = \infty,\) with \(k \in 1, ..., n\).
3) Repeat the following steps until \(i = n,\) with \(i\) starting from \(i = 1\)
   - For \(i = 1\)
     a. Set \(load = 0\) and \(j = i\). Repeat the following steps until \(j > n\) or \(load > Q\).
     b. If \(i = j,\) then \(cost = \text{distance} (\text{depot},T_i) + \text{distance} (T_i, \text{depot})\).
        If \(i \neq j,\) then \(cost = \text{cost} - \text{distance} (T_{i-1}, \text{depot}) + \text{distance} (T_{j-1}, T_j) + \text{distance} (T_j, \text{depot})\).
     c. If \(load \leq Q\) (\(Q\) is vehicle capacity) and \(V_{i-1} + cost < V_j,\) then
        \(V_j = V_{i-1} + cost\)
        \(P_j = i - 1\)

     (\(V_j\) is the minimum cost of the paths connecting the depot to the \(j\)th customer. The role of \(P_j\) is to store the predecessor of node \(j\) on the shortest path leading to \(j\)).
     d. Set \(j = j + 1\)
     e. Back to step a
        - If \(j > n\) or \(load > Q,\) then set \(i = i + 1\).
Algorithm 2. Form a solution after Split.
1) Make a list of solutions \(S,\) with \(S = \emptyset\) and set \(j = n\)
2) Repeat the following steps until \(j = 0\)
   a. \(Route = \emptyset\)
b. For \( k = P_j + 1 \) to \( k = j \), add the \( k \)-customer on the giant tour to the Route

c. Save the Route in the list of solutions \( S \) and set \( j = P_j \)

d. Back to step a.

4. Results and Discussion

4.1 Instance of GWO application for MDVRP

A company will send goods to 9 customers, namely customer 2, 3, 4, 5, 6, 7, 8, 9, and 10. The company has 2 depots used for distribution of goods, namely depot 0 and depot 1. Vehicles used in each depot has the same capacity, which is 70 cardboard boxes [14]. A list of demands for each customer is presented in Table 2 and data on the distance of the depot to the customer and the distance between customers are in the following Table 3.

**Table 2.** Customer demands data.

| Customer | Demand |
|----------|--------|
| 2        | 15     |
| 3        | 20     |
| 4        | 10     |
| 5        | 25     |
| 6        | 15     |
| 7        | 25     |
| 8        | 20     |
| 9        | 30     |
| 10       | 25     |

**Table 3.** Depot and customer distance data (2 depot 9 customers).

| \( d \) | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 0  | 160| 50 | 70 | 80 | 165| 143| 154| 93 | 60 | 185|
| 1  | 160| 0  | 185| 154| 87 | 100| 60 | 75 | 90 | 199| 120|
| 2  | 50 | 185| 0  | 38 | 110| 160| 150| 193| 137| 10 | 180|
| 3  | 70 | 154| 38 | 0  | 75 | 125| 115| 176| 130| 48 | 145|
| 4  | 80 | 87 | 110| 75 | 0  | 51 | 36 | 119| 97 | 120| 71 |
| 5  | 165| 100| 160| 125| 51 | 0  | 36 | 140| 140| 170| 20 |
| 6  | 143| 60 | 150| 115| 36 | 36 | 0  | 100| 160| 56 |
| 7  | 154| 75 | 193| 176| 119| 140| 100| 0  | 58 | 203| 160|
| 8  | 93 | 90 | 137| 130| 97 | 140| 100| 58 | 0  | 147| 160|
| 9  | 60 | 199| 10 | 48 | 120| 170| 160| 203| 147| 0  | 55 |
| 10 | 185| 120| 180| 145| 71 | 56 | 56 | 160| 160| 55 | 0  |

The problem is determining the minimum routes from each depot to deliver goods to the 9 customers without violating the capacity of the available vehicles. With each route starting from the depot and ending at the same depot.

MDVRP problem resolution uses the GWO algorithm as follows.

1. Grouping
   - \( d_{20} = 50 < d_{21} = 185 \), customer 2 are grouped into depot 0
   - \( d_{30} = 70 < d_{31} = 154 \), customer 3 is grouped into depot 0
   - \( d_{40} = 80 < d_{41} = 87 \), customer 4 are grouped into depot 0
   - \( d_{50} = 165 > d_{51} = 100 \), customer 5 is grouped into depot 1
   - \( d_{60} = 143 > d_{61} = 60 \), customer 6 is grouped into depot 1
   - \( d_{70} = 154 > d_{71} = 75 \), customer 7 are grouped into depot 1
• $d_{80} = 93 > d_{91} = 90$, customer 8 is grouped into depot 1
• $d_{90} = 60 < d_{91} = 199$, customer 9 is grouped into depot 0
• $d_{100} = 185 > d_{101} = 120$, customer 10 are grouped into depot 1
Depot 0 serves customers 2, 3, 4, and 9, while depot 1 serves customers 5, 6, 7, 8, and 10.

2. Giant tour formation
a. Depot 0

$n = 5$, so the size of the population is $n + 1 = 5 + 1 = 6$
Population array $[0] = [ ]$
Because point 0 is a depot, point 0 is added to the population array.
Population $[0] = [0]$
Choose three randomly customer point, selected points 3, 4, and 2. The population becomes:
Population $[0] = [0, 3]$, cost = 70
Population $[1] = [0, 4]$, cost = 80
Population $[2] = [0, 2]$, cost = 50

Before entering the next iteration, first check whether number of the population is more than population size. Because size of array is only $3 <$ population size, the iteration continues.
Population $[0] = [0, 3, 2]$, cost = 108
Population $[3] = [0, 3, 9]$, cost = 118
Population $[4] = [0, 3, 4]$, cost = 145
Population $[1] = [0, 4, 3]$, cost = 155
Population $[5] = [0, 4, 2]$, cost = 190
Population $[6] = [0, 4, 9]$, cost = 200
Population $[2] = [0, 2, 9]$, cost = 60
Population $[7] = [0, 2, 3]$, cost = 88
Population $[8] = [0, 2, 4]$, cost = 160

Now, the number of population is $9 >$ population size, then delete the population with the largest cost, those are Population $[6]$, Population $[5]$, and Population $[8]$ so that the remaining population is
Population $[0] = [0, 3, 2]$, cost = 108
Population $[1] = [0, 3, 9]$, cost = 118
Population $[2] = [0, 3, 4]$, cost = 145
Population $[3] = [0, 4, 3]$, cost = 155
Population $[4] = [0, 2, 9]$, cost = 60
Population $[5] = [0, 2, 3]$, cost = 88

Now iteration can be continued, as number of population has not exceeded population size
Population $[0] = [0, 3, 2, 9]$, cost = 118
Population $[6] = [0, 3, 2, 4]$, cost = 218
Population $[1] = [0, 3, 9, 2]$, cost = 128
Population $[7] = [0, 3, 9, 4]$, cost = 238
Population $[2] = [0, 3, 4, 2]$, cost = 255
Population $[8] = [0, 3, 4, 9]$, cost = 265
Population $[3] = [0, 4, 3, 2]$, cost = 193
Population $[9] = [0, 4, 3, 9]$, cost = 203
Population $[4] = [0, 2, 9, 3]$, cost = 108
Population $[10] = [0, 2, 9, 4]$, cost = 180
Population $[5] = [0, 2, 3, 9]$, cost = 136
Population $[11] = [0, 2, 3, 4]$, cost = 163

There are 12 populations which is greater than the population size, then delete 6 population arrays with the largest cost, namely Population $[6]$, Population $[7]$, and Population $[2]$, Population $[8]$, Population $[9]$, and Population $[3]$ so the remaining population is
Population $[0] = [0, 3, 2, 9]$, cost = 118
Population \[1\] = [0, 3, 9, 2] \hspace{1cm} \text{cost} = 128
Population \[2\] = [0, 2, 9, 3] \hspace{1cm} \text{cost} = 108
Population \[3\] = [0, 2, 9, 4] \hspace{1cm} \text{cost} = 180
Population \[4\] = [0, 2, 3, 9] \hspace{1cm} \text{cost} = 136
Population \[5\] = [0, 2, 3, 4] \hspace{1cm} \text{cost} = 163

Because size of the array is not greater than the population size then the iteration can be continued.

Population [0] = [0, 3, 2, 9, 4] \hspace{1cm} \text{cost} = 238
Population \[1\] = [0, 3, 9, 2, 4] \hspace{1cm} \text{cost} = 238
Population \[2\] = [0, 2, 9, 3, 4] \hspace{1cm} \text{cost} = 183
Population \[3\] = [0, 2, 9, 4, 3] \hspace{1cm} \text{cost} = 255
Population \[4\] = [0, 2, 3, 9, 4] \hspace{1cm} \text{cost} = 256
Population \[5\] = [0, 2, 3, 4, 9] \hspace{1cm} \text{cost} = 283

All customer points are contained in a population array. Next add point 0 as a depot into each population array.

Population [0] = [0, 3, 2, 9, 4, 0] \hspace{1cm} \text{cost} = 318
Population \[1\] = [0, 3, 9, 2, 4, 0] \hspace{1cm} \text{cost} = 318
Population \[2\] = [0, 2, 9, 3, 4, 0] \hspace{1cm} \text{cost} = 263
Population \[3\] = [0, 2, 9, 4, 3, 0] \hspace{1cm} \text{cost} = 325
Population \[4\] = [0, 2, 3, 9, 4, 0] \hspace{1cm} \text{cost} = 336
Population \[5\] = [0, 2, 3, 4, 9, 0] \hspace{1cm} \text{cost} = 343

The solution for forming a giant tour for depot 0 is the population with the smallest cost, namely Population \[2\] = [0, 2, 9, 3, 4, 0].

b. Depot 1

\( n = 6 \), so the population size is \( n + 1 = 6 + 1 = 7 \)

Population array \([0] = [\ ]\)

Because point 1 is a depot, point 1 is added to the population array.

Population \([0] = [1]\)

Choose three randomly customer point, selected points 5, 7, and 8. The population becomes:

Population \([0] = [1, 5]\) \hspace{1cm} \text{cost} = 100
Population \([1] = [1, 7]\) \hspace{1cm} \text{cost} = 75
Population \([2] = [1, 8]\) \hspace{1cm} \text{cost} = 90

The next process in forming a giant tour for depot 1 is the same as forming a giant tour in depot 0. In order to obtain the solution for forming a giant tour for depot 1 is \([1, 7, 8, 6, 5, 10, 1]\).

3. Routing

\textbf{Depot 0}

at depot 0 there are 4 customers \((n = 4)\) and the vehicle capacity is 70 cardboard boxes \((Q = 70)\).

\( T_1 = 2, T_2 = 9, T_3 = 3, \) and \( T_4 = 4 \).

\textbf{Algorithm 1. \textit{Split}}

\( V_0 = 0, P_0 = 0, \) and \( V_1 = \infty, V_2 = \infty, V_3 = \infty, V_4 = \infty \)

a. For \( i = 1, load = 0 \)

- \( j = i = 1 \)

\hspace{1cm} load = load + q(2) = 0 + 15 = 15 \leq 70
\hspace{1cm} cost = distance(0,2) + distance(2,0) = 50 + 50 = 100

Because \( 15 \leq 70 \) and \( V_0 + cost < V_1 \leftrightarrow 0 + 100 = 100 < \infty \), then \( V_1 = 100 \) and \( P_1 = 1 - 1 = 0 \)

- \( j = 1 + 1 = 2 \)

\hspace{1cm} load = load + q(9) = 15 + 30 = 45 \leq 70
\hspace{1cm} cost = 100 - distance(2,0) + distance(2,9) + distance(9,0)
\hspace{1cm} = 100 - 50 + 10 + 60 = 120
Because $45 \leq 70$ and $V_0 + \text{cost} < V_2 \leftrightarrow 0 + 120 = 120 < \infty$, then $V_2 = 120$ and $P_2 = 1 - 1 = 0$

- $j = 2 + 1 = 3$
  
  $load = load + q(3) = 45 + 20 = 65 \leq 70$
  
  $cost = 120 - \text{distance} (9,0) + \text{distance} (9,3) + \text{distance} (3,0)$
  
  $= 120 - 60 + 48 + 70 = 178$

  Because $65 \leq 70$ and $V_0 + \text{cost} < V_3 \leftrightarrow 0 + 178 = 178 < \infty$, then $V_3 = 178$ and $P_3 = 1 - 1 = 0$

- $j = 4$
  
  $load = load + q(4) = 65 + 10 = 75 > 70$

  Because $load = 75 > 70$, then $i = 1 + 1 = 2$

b. For $i = 2$, $load = 0$

- $j = i = 2$
  
  $load = load + q(9) = 0 + 30 = 30 \leq 70$
  
  $cost = \text{distance} (0,9) + \text{distance} (9,0) = 60 + 60 = 120$
  
  $30 \leq 70$ and $V_1 + \text{cost} < V_2 \leftrightarrow 100 + 120 = 220 > 120$

- $j = 2 + 1 = 3$
  
  $load = load + q(3) = 30 + 20 = 50 \leq 70$
  
  $cost = 120 - \text{distance} (9,0) + \text{distance} (9,3) + \text{distance} (3,0)$
  
  $= 120 - 60 + 48 + 70 = 178$
  
  $30 \leq 70$ and $V_1 + \text{cost} < V_2 \leftrightarrow 100 + 178 = 278 > 178$

- $j = 3 + 1 = 4$
  
  $load = load + q(4) = 50 + 10 = 60 \leq 70$
  
  $cost = 178 - \text{distance} (3,0) + \text{distance} (3,4) + \text{distance} (4,0)$
  
  $= 178 - 70 + 75 + 80 = 263$

  Because $60 \leq 70$ and $V_1 + \text{cost} < V_3 \leftrightarrow 100 + 263 = 363 < \infty$, then $V_4 = 363$ and $P_4 = 2 - 1 = 1$

- $j = 4 + 1 = 5$, because $5 > 4$, then $i = 2 + 1 = 3$

c. For $i = 3$, $load = 0$

- $j = i = 3$
  
  $load = load + q(3) = 0 + 20 = 20 \leq 70$
  
  $cost = \text{distance} (0,3) + \text{distance} (3,0) = 70 + 70 = 140$
  
  $20 \leq 70$ and $V_2 + \text{cost} < V_3 \leftrightarrow 120 + 140 = 260 > 178$

- $j = 3 + 1 = 4$
  
  $load = load + q(4) = 20 + 10 = 30 \leq 70$
  
  $cost = 140 - \text{distance} (3,0) + \text{distance} (3,4) + \text{distance} (4,0)$
  
  $= 140 - 70 + 75 + 80 = 225$

  Because $30 \leq 70$ and $V_2 + \text{cost} < V_4 \leftrightarrow 120 + 225 = 345 < 363$, then $V_4 = 345$
  
  and $P_4 = 3 - 1 = 2$

- $j = 4 + 1 = 5$, because $5 > 4$, then $i = 3 + 1 = 4$

d. For $i = 4$, $load = 0$

- $j = i = 4$
  
  $load = load + q(4) = 0 + 10 = 10 \leq 70$
  
  $cost = \text{distance} (0,4) + \text{distance} (4,0) = 80 + 80 = 160$

  Because $10 \leq 70$ and $V_3 + \text{cost} < V_4 \leftrightarrow 178 + 160 = 338 < 345$, then $V_4 = 338$
  
  and $P_4 = 4 - 1 = 3$

Because $i = n \leftrightarrow 4 = 4$, then Algorithm 1 has been completed.

**Algorithm 2.** Form the solution $S$

From algorithm 1 is obtained $P_1 = 0$, $P_2 = 0$, $P_3 = 0$, and $P_4 = 3$
$S = \emptyset$ and $j = 4$

a. For $j = 4$

   $Route = 4$

   $k = P_4 + 1 = 3 + 1 = 4$. The 4th customer is 4, then $Route = (4)$

   $S = \{(4)\}$ and $j = P_4 = 3$

b. For $j = 3$

   $Route = \emptyset$

   $k = P_3 + 1 = 2 + 1 = 3$. The 3rd customer is 3, then $Route = (2, 9, 3)$

   $S = \{(4), (2, 9, 3)\}$ and $j = P_3 = 0$

Algorithm 2 has been completed ($j = 0$).

The routing phase for depot 0 has been completed and the following results are obtained.

Route 1: $0 - 2 - 9 - 3 - 0$ a distance of 178 km
Route 2: $0 - 4 - 0$ a distance of 160 km

So that the total distance traveled for each route in serving all customers in depot 0 is 338 km.

**Depot 1**

The routing phase for depot 1 is the same as routing for depot 0 with $n = 5, Q = 70$ and $T_1 = 7, T_2 = 8, T_3 = 6, T_4 = 5$ and $T_5 = 10$. So that the routing stage solution obtained for depot 1 is as follows.

Route 1: $1 - 7 - 8 - 1$ a distance of 223 km
Route 2: $1 - 6 - 5 - 10 - 1$ a distance of 272 km

The total distance traveled for each route in serving all customers in depot 1 is 495 km.

So, the total distance traveled by depot 0 and depot 1 on the MDVRP using the GWO algorithm is 833 km.

### 4.2 Implementation of MDVRP-GWO

The implementation of the GWO algorithm in solving MDVRP problems is designed in the Borland Delphi 7.0 programming language. Figure 2 presents the pseudo-code for proposed MDVRP-GWO algorithm.
Figure 2. The pseudo-code of MDVRP-GWO.

Figures 3 to 6 are examples of application displays for examples of MDVRP problems in the previous section.

Figure 3. Graph instance of 2 depots and 9 customers problem.
Figure 4. Data input of 2 depots and 9 customers problem.

Figure 5. Solution of 2 depots and 9 customers problem.

Figure 6. Route results in graph instance 2 depot 9 customer.
The MDVRP-GWO application is designed in Indonesian. Therefore, the following is a description of the MDVRP-GWO application in English.

- 'Input Titik' means input point. At the input point, there is 'Pilih Titik' which means selecting the point to be inputted, namely the point of the depot and the point of the customer.
- In 'Input Data' there is 'Kapasitas Kendaraan', 'Iterasi', 'Permintaan Customer', and 'Tabel Jarak', which means vehicle capacity, iteration, customer demand, and distance table (the distance between the depot to the customer and the distance between the customer), respectively. In 'Input Data' there are also 5 buttons with different uses. 'Hasil' to start the search for a solution, 'Simpan' to save the data, 'Buka' to reopen the saved data, 'Baru' to start the program from scratch, and 'Keluar' to stop the program.
- 'Hasil' means the solution obtained.
- 'Graph Hasil' is to display the route image formed from each depot point obtained from the search results for the program solution.

To test performance of the GWO algorithm, some datasets from [15] are used. For each dataset, the application is executed 10 times, and it is tested using 5, 10, 50, 100 dan 500 iterations, respectively. From the testing, number of iterations affects the result, as depicted in Figure 7.

![Figure 7. Effect of number of the iteration to distance.](image)

The best result (the shortest distance) of the testing comes from 500 iterations. It is expected that better result would be get if the more iterations are performed, since trends of the graphic declines as number of iterations arises. Another factor is generation of the initial population in GWO is in random order, hence the selected starting vertex is not necessarily the vertex that give the best solution. Thus, if the iteration used is greater, the more combination of solutions will be formed, so that the resulting solution will be better.

The solution of 500 iterations then compared to result of Demirel that used Ant Colony Optimization (ACO) with 2000 iterations on 10 time testing and using parameter $\theta = 45, \alpha = 1, \beta = 1, \rho = 0.10, q_0 = 0.08$ [6], which is the ACO algorithm is also a metaheuristic algorithm. Comparisons are made to see the best solution obtained from the two algorithms in solving MDVRP problems. Table 3 displays result of the comparison:

| Dataset | ACO   | GWO   |
|---------|-------|-------|
| P01     | 629.60| 596.24|
| P06     | 1027.60| 949.67|
| P15     | 3731.60| 2835.77|
From the comparison in the table, GWO gives better result than ACO on all datasets. However, more tests should be held to compare GWO with other algorithm, where the total distance on all datasets using the GWO algorithm is shorter than ACO. More improvement of GWO itself will give better result, such as implements improvement algorithm and combine with other algorithm as a new hybrid algorithm.

5. Conclusion
This paper introduces the GWO algorithm for solving MDVRP. There are three stages at the completion of the MDVRP, namely grouping, forming a giant tour, and routing. At the grouping stage, customers are grouped into the depot with the nearest distance. In the next stage, each customer group at each depot that has been formed will be formed a giant tour by forming a population array. The optimal condition at the stage of forming a giant tour is when a population array has been formed which contains all the customers and ends with a depot point. The solution for this stage is the population array with the smallest cost. At the routing stage, a split on the giant tour is carried out by paying attention to vehicle capacity and customer demands. So that a number of routes will be formed at each depot, with the total demand on each route not exceeding the vehicle capacity. MDVRP-GWO application was test on different size of datasets. The test results show that the GWO algorithm provides a better solution with a shorter total distance when compared to the ACO algorithm on all datasets. In the ACO algorithm, the solution is determined by the parameter \( (\theta, \alpha, \beta, \rho, q_0) \) values used. If the selected parameter value is not correct, the resulting solution is also not optimal.

For future work, the GWO can be improved to better performance by implements improvement algorithm and combine with other algorithm as a new hybrid algorithm. Also, a comparison between GWO algorithm and other metaheuristics algorithm which are used to solve the MDVRP could be investigated.

References
[1] Cordeau J-F, Gendreau M and Laporte G 1997 A tabu search heuristic for periodic and multi-depot vehicle routing problems Networks 30 105–119
[2] Demirel T and Yılmaz S 2012 A new solution approach to multi-depot vehicle routing problem with Ant Colony Optimization Journal of Multiple-Valued Logic and Soft Computing 18 421–439
[3] Domínguez-Martín B, Rodríguez-Martín I, and Salazar-González J-J 2018 The driver and vehicle routing problem Computers and Operations Research 92 56–64
[4] Hu P, Pan J S and Chu S C 2020 Improved Binary Grey Wolf Optimizer and its application for feature selection Knowledge-Based Systems 195 105746
[5] Korayem L, Khorsid M and Kassem S S 2015 Using Grey Wolf algorithm to solve the capacitated vehicle routing problem IOP Conference Series Material Science and Engineering 83 012014
[6] Masruroh A 2012 Algoritma Clark and Wright pada Multi Depot Vehicle Routing Problem (MDVRP) (Malang: FMIPA UM)
[7] Mirjalili S, Mirjalili S M and Lewis A 2014 Grey Wolf Optimizer Advance of Engineering Software 69 46–61
[8] Prins C 2004 A simple and effective evolutionary algorithm for the vehicle routing problem Computers and Operations Research 31 1985–2002
[9] Rashid T A, Abbas D K and Turel Y K 2019 A multi hidden recurrent neural network with a modified grey wolf optimizer PLOS ONE 14 e0213237
[10] Resende M G C and Ribeiro C C 2014 GRASP: greedy randomized adaptive search procedures Search Methodologies 287–312
[11] Shaheen A, Sleit A and Al-Sharaaeh S 2018 A solution for travelling salesman problem using Grey Wolf Optimizer algorithm Journal of Theoretical and Applied Information Technology 96 6256–6266
[12] Song X, Tang L, Zhao S, Zhang X, Li L, Huang J and Cai W 2015 Grey Wolf Optimizer for parameter estimation in surface waves Soil Dynamics and Earthquake Engineering 75 147–157

[13] Surekha P and Sumanthi S 2011 Solution to multi-depot vehicle routing problem using Genetic algorithms World Applied Programming 1 118–131

[14] Villegas J G, Prins C, Prodhon C, Medaglia A L, and Velasco N 2010 GRASP/VND and multi-start evolutionary local search for the single truck and trailer routing problem with satellite depots Engineering Applications of Artificial Intelligence 23 780–794

[15] Wong L I, Sulaiman M H, Mohamed M R and Hong M S 2014 Grey Wolf Optimizer for solving economic dispatch problems 2014 IEEE International Conference Power and Energy (PECon) 150–154