Received: May 8, 2021. Revised: July 9, 2021.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Hybrid Metaheuristics for Solving Vehicle Routing Problem in Multi Bulk Product Shipments with Limited Undedicated Compartments

Antono Adhi[1,2] **Budi Santosa[1]** **Nurhadi Siswanto[1]***

[1]*Industrial and Systems Engineering, Institut Teknologi Sepuluh Nopember, Indonesia*
[2]*Industrial Engineering, Universitas Stikubank, Indonesia*
* Corresponding author's Email: siswanto@ie.its.ac.id

Abstract: Studies and research related to the Vehicle Routing Problem (VRP) are mostly carried out on simple constraints. The study was conducted to find the shortest route that the carrier will take. Effective algorithms are sought to find better solutions. Another study will develop VRP cases to be closer to real conditions. In actual conditions, the VRP cases are very complex. Several solutions that are considered good can be implemented to solve this problem. VRP in shipping bulk products, such as oil, cements, and fertilizers is one of the daily logistical cases carried out in an archipelago country. Bulk products are shipped from the depot port city to other islands. The optimum solution is needed to make the delivery shorter distance and the products can be sent faster. The solution to the VRP case with a single product can no longer solve the problem of bulk product shipment. Bulk product shipment involves more than one product with more than one ship compartment. The number of ship compartments that are smaller than the number of products will further increase the complexity of the VRP of bulk product shipment. The combination of several metaheuristic methods and strategies of product selection to fill undedicated compartment of ship is used to find the shortest route. Modified Nawas Enscore Ham (M-NEH), Particle Swarm Optimization (PSO), and 3-Opt algorithms are combined to solve VRP. This hybrid algorithm called Hybrid Particle Swarm Optimization (HPSO) is improved and supports each other. HPSO provides a better solution than those algorithms separately and other metaheuristics namely Tabu Search (TS) and Genetic Algorithm (GA). The implementation of the algorithm to solve real data of bulk cement transportation supports the conclusion that HPSO is superior to the other algorithms. It has 0.99% more effective than the second best and about 3.42% more effective than the average total distance of five other compared metaheuristic methods.

Keywords: Metaheuristics, Vehicle routing problem, Bulk product shipment, Bulk cement shipment, Optimization.

1. Introduction

The Vehicle Routing Problem (VRP) is a classic case that is very important in operational research [1]. Dantzig and Ramser were the people who first proposed VRP in transporting products on a route [1]. At first, this VRP case was known as the Truck Dispatching Problem. In his research, Dantzig sought a solution to find the shortest route for the gasoline truck from the terminal to service stations. VRP was developed by Dantzig from the Traveling Salesman Problem (TSP) with limited carrier capacity. Shipments of a certain product will be against the capacity constraint. Carrier must return to the depot to retrieve the products. If the depot and the customers are in the same company, usually it must consider the inventory of the customers. This problem is called as Inventory Routing Problem (IRP). On the other hand, the problem is defined as VRP when the depot delivers products at the demand of customers without inventory consideration.

VRP can be defined as a problem of selecting the shortest route to send products from the depot to several customers [2]. By obtaining the shortest route, other costs can be optimized. The current study is carried out by improving the method and developing variants to bring VRP closer to the actual conditions [2]. During 70 years of research conducted on VRP, VRP cases are increasingly diverse and increasingly

difficult to be solved [3]. According to the current problem, there are many different types of VRP which the most famous problem is Capacitated VRP (CVRP), VRP with Pickup and Delivery (VRPPD), and Distance Constraint VRP (DCVRP) [3]. In the CVRP vehicle are identical and are based at a single central depot, demand may not be split and are known in advance, and only the capacity restrictions for the vehicle are imposed [4]. The majority variants of VRP were mostly related to new constraints to the original VRP. Vehicles are not only restricted in complicates vehicle capacity but further in total time duration of a route or in time-window constraints [5, 6]. In time-window constraints, customer can be served in specific periods during which the location can be reached, or customer is open, due to traffic limitation [7]. The other variants of constraint are characterized by multiple vehicle types, multiple trips of vehicles, multiple depots, and other operational issues of shipping operational as loading constraints [8].

Exact methods were developed to solve VRP [9]. Over the past 50 years VRP have solved by branch-and-bound algorithm [10]. Thus, VRP is a complex an NP-Hard problem [9, 11], exact methods became inefficiency to solve large scale VRP instances [9]. More cities considered in this problem will increase the level of complexity. Possible solutions to solve VRP cases are heuristics or metaheuristics [12]. Classical heuristics were developed mostly between 1960 and 1990, and metaheuristics were developed after that period [9]. Several studies use heuristic and metaheuristic methods to solve VRP problems. Symbiotic organisms search is used to solve the capacitated vehicle routing problem (CVRP) [13]. The combination of Chicken Swarm Optimization (CSO) and Tabu Search (TS) algorithms is also used to solve CVRP [3]. The Simulated Annealing algorithm is used to solve the Hybrid Vehicle Routing Problem (HRVP) developed from the Green Vehicle Routing Problem (G-VRP) [14]. Neighborhood search is used to improve solutions to solve vehicle routing problems with cross-docking [15].

One of the cases developed in VRP is the management of shipments by sea. This routing problem determines the route of the ship to deliver products from a depot city to consumption cities to achieve minimum cost [16]. Product shipment by sea transportation is important in global trade. One of the crucial and important shipping is bulk product shipping. Late delivery of bulk product will increase in other costs. In real conditions, the case of VRP in bulk product shipment will be more complex with more than one type of product and a limited number of ship compartments. The number of compartments is smaller than the product types, therefore it is not possible to carry all types of products. One compartment cannot be filled with more than one type of product. The undedicated compartment can be filled with one desired product and another product after the compartment has been emptied [17]. VRP in bulk product shipment is the selection of optimum routes from depot ports to other ports with variants in the number of products and limitation of ship's compartments. The demand of every city can be more than one type of products. This VRP case can be implemented in other products with the similar conditions. In an undedicated compartment type, one compartment only be filled with a type of product and cannot be mixed with other product. This compartment can be filled with different type of products after it is empty.

This problem condition has interesting problem solving process. It can provide a solution that jumps over the usual problem solving for the original problem of VRP. The optimization considerations are not only in determining the shortest distance but also in selecting the product to be carried, even though the objective of the VRP is to find the shortest route. Therefore, the following city routes may not in the next nearest city, but cities with the same product demand. If it only considers the proximity of the distances of the cities, the ships will repeatedly take products to the depot. However, if the city with the same product demand is very detrimental to the mileage, the search process will consider the product being transported according to the nearest city. These conflicting problems will be resolved with the development of metaheuristics and the most appropriate products selection strategy.

In this study, a combination of several metaheuristic algorithms based on Particle Swarm Optimization (PSO) is developed as a new algorithm to solve VRP problems in bulk product shipment. This combination of algorithms is called Hybrid Particle Swarm Optimization (HPSO). The Nawas Enscore Ham (NEH) algorithm interacts and improves each other with the PSO algorithm. NEH is modified to provide multi results that is used to initialize PSO populations. Standard PSO is modified with quantum process to avoid trapped in local optimum. This quantum process is not use random process as used in general metaheuristics process but use NEH algorithm. This process is more directed than random process. The results of the combination of these two algorithms are refined by the 3-Opt algorithm. The problem of VRP in this research is focus on the limitation of compartments amount to carry many types of products. HPSO algorithms try to improve the optimal length of route by choosing

precise type of product to carry. HPSO is combined with the methods of selecting products type and compartments in the route selection process.

The rest of the paper is organized as follows. The problem description is presented in Section 2. The detailed proposed model is described in Section 3. Numerical experiments based on a set of problem instances are presented in Section 4. In this section, HPSO is also tested to solve real problem and compared its result with other algorithms. The last section provides the conclusion of the research result and possibility of next agenda of research.

2. Problem description

This paper studies multi products delivery by a ship with limited compartments number. This problem is adapted from the real condition therefore the research of this paper can be used to solve real VRP in the industrial world. Some conditions of this problem can be defined as follows:

- The number of compartments of the ship is less than the number of product types. If co is the number of compartments and po is the number of product types, then it can be defined as follows:

$$co < po \tag{1}$$

  This condition will cause limitations on the type of products to be delivered. A ship has to return to the city to deliver the rest of the products that have not been delivered yet. This limitation causes the ship to plan more than one routes.
- Compartments have limited capacity. If vji is the volume of product j that is placed in compartment i and CMAXi is the maximum capacity of compartment i, then it can be defined as follows:

$$v_{ji} \leq CMAX_i \tag{2}$$

  This condition will make not all of the product demands from customers that have been determined to be brought can be put in the compartments. This limitation also causes the ship will plan more than one routes. We assume that the ships have the same velocity so that the cost of all ships will be linear with the distance.

All the constraints cause the ship to return to the depot city after all of the compartments are empty. A new strategy will be planned to determine what types of products should be chosen to meet the city demands. A ship can return to the last city if not all of the demands from the city are fulfilled. On the other side, the objective of the VRP has to be considered.

The objective of the VRP is to achieve the shortest route although it can be developed to the other goal namely time or cost of delivery. Distance is the original goal of VRP and this paper will search for the shortest distance. The ship will define routes until all the demands of the city are fulfilled. If D denotes distance of route length, r denotes the number of routes, m_i denotes the number of vertexes in route i, and $d_{j(j+1)}$ denotes travel distance from vertex j to vertex j+1, the objective of the VRP can be expressed as follows:

$$D = \min \sum_{i=1}^{r} \left(\left(\sum_{j=1}^{n-1} d_{j(j+1)} \right) + d_{n1} \right) \tag{3}$$

j=1 is the depot city. d_{n1} indicates the ship will return to the depot after the last city (n) in route i. It happens when compartments are empty and continue to deliver to the next route or after all demands of cities are completely fulfilled.

3. Proposed model

NEH is one of the algorithms that are combined in HPSO. NEH is an acronym for its discoverers: Nawaz, Enscore, and Ham. NEH is a heuristic algorithm that was first developed to solve production scheduling problems [18]. The problem was determining the order of jobs to be processed from one machine to another. NEH is specifically designed to solve production scheduling problems. It is used to solve flow shop problems to find the shortest total completion time. The solution is quite efficient. The time it takes to complete the process was very fast. The NEH method has a concept that partially the production schedule will provide a better total turnaround time, then in overall, it will also provide a good total turnaround time. The original NEH algorithm is as follows:

```
Count fitness of every particle and put to sequence S
sort by fitness from bad to good
Get one particle p from S and put in sequence N
For i = 2 to rest of particle p in S
  Get particle p_i from S
    For position pos=1 to number particle in N+1
      If fitness(p_pos) < bestfitness then
        new bestfitness position = pos
      End If
  End For
  Put particle p_i to sequence N in best
  fitness position
End for
```

S and N are sequences (1, 2, 3, ..., p) where particles p denotes the number of cities minus depot city. The sequence of S and N contain the sort of cities

that are served by depot city. First, *S* is set with the sequence of cities sorted by fitness in descending order and *N* is set to empty. The fitness of every city in *S* is calculated partially and they are sorted from the worst fitness to the best fitness. The fitness is the shortest distance from depot city to the other city.

The next process is taking particles one by one from *S* to *N* to determine their best position in the sequence of particles that have been formed in *N*. In this process the particles from S are placed on N in the index position with the best fitness value. This fitness function contains three strategies of selecting products that are brought to the cities from depot city. These strategies will be explained in the next section. The process is repeated until all particles in *S* are taken into *N*. *N* is the result of best route searching. If the process in the NEH method is repeated, it will produce the same series of particles because there is no consideration of probability in the NEH algorithm.

Although NEH gives good results, it will, however, not provide the best results in a big number of particles. NEH is less effective because when a particle is looking for the best position in a series of particles that have already been formed, the sequence of particles that have been formed does not change position. In this research, the NEH algorithm is improved by shifting the position of the particles to look for a better possible position of the particles that have already been formed. The shift of position method uses Metaheuristic Particle Swarm Optimization (PSO) algorithm.

Particle swarm optimization (PSO) is a population-based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995 [19, 20]. It was inspired by the social behavior of flocking birds or fish. PSO shares many similarities with evolutionary computational techniques such as Genetic Algorithms (GA). The system is initialized with a random solution population and searches for optima by updating the generation. However, unlike GA, PSO does not have evolutionary operators such as crosses and mutations. In PSO, a potential solution, called a particle, moves through the problem space by following the optimum particle current. Particles denote cities minus depot city.

PSO is initialized by a random group of particles and then looks for the optimal value by updating the generation. In each iteration, each particle is updated following the two best values. The first called *pbest* is the best solution (fitness) that has been achieved so far. The other best value called *gbest* is a global best. When a particle takes part of the population as its topological neighbor, the best value is the local best value and is called *lbest*. After finding the two best values, the particle updates its velocity and position with the following Eqs. (4) and (5):

$$v_t = w \times v_{t-1} + c1 \times rand \times (pbest - present_{t-1}) + c2 \times rand \times (gbest - present_{t-1}) \quad (4)$$

$$present_t = present_{t-1} + v_t \quad (5)$$

v_t is defined as a sequence (1, 2, ..., c) of particle velocity which c denotes the number of cities without depot city. $present_t$ which is the corresponding particle is a sequence (1, 2, ..., c) of particle's parameter value. These values describe the value of the city. $present_t$ must be converted by sorting it to obtain the sequence of cities. t denotes the current iteration and t-1 denotes the previous iteration. *pbest* and *gbest* are sequence (1, 2, ..., c) that save best value of every population and global population. *rand* is an intermediate random number (0,1). $c1$ and $c2$ which usually have values equal to 2 *are learning factors.* If the values of $present_t$ are out of range, the values of $c1$ and $c2$ can be reduced. w which is set to have value 0.9 is inertia weight. This variable is used to improve the performance of PSO [21]. The pseudo-code of the procedure is as follows:

```
For each particle
  Initialize particle
End For
Do
  For each particle
    Calculate fitness value
    If the fitness value is better than the
    best fitness value (pBest) in history
      set current value as the new pBest
    End If
  End For
  Choose the particle with the best fitness
  value of all the particles as the gBest
  For each particle
    Calculate particle velocity according to Eq. (4)
    Update particle position according to Eq. (5)
  End For
While maximum iterations or minimum error criteria
is not attained
```

The fitness function of the PSO algorithm contain three strategies of selecting products that are brought to the cities from depot city. These strategies will be explained in the next section. PSO is used to improve NEH performance by shifting the sequence of particles that have been formed by NEH. NEH is also used to improve PSO performance by using this in the PSO initialization process. In the original PSO, the initialization process is carried out randomly. NEH will provide the best chromosome sequence in one of

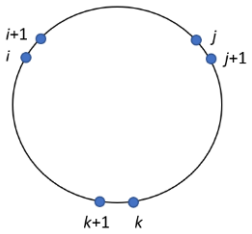

Figure. 1 Three connections of 3-Opt

the PSO populations. Not all populations are initialized by NEH. This process will improve the PSO. It gives a better global value.

The 3-Opt algorithm is a local search algorithm to find optimal solutions for combinatorial problems. This algorithm is widely used to solve the Traveling Salesman Problem (TSP). 3-Opt will look for three connections (edges) in a network. These connections are represented by three variables namely *i*, *j*, *k*. These variables cut sequence of the network as depicted in Fig. 1.

The three connections will be changed by deleting and reconnecting to a different connection [22]. With the search for three connections, there are seven possible new connections as shown in Fig. 2 [23]. Three of the seven possible reconnections only consider changes in two connections or what is often called the 2-Opt algorithm. If the new connection provides a better fitness value, the network path will be changed, and the search process starts all over again.

The 3-Opt algorithm usually is initialized by generating a random sequence of particles. Initialization can also be done by fixing a sequence of particles that have been formed from the calculation of other algorithms. The 3-Opt algorithm is as follows:

```
Generate sequence O
isFoundBetter = true
While isFoundBetter
  For i = 1 to n – 5
    For j = i + 2 to n – 3
      For k = j + 2 to n – 1
        For Method=1 to 7
          If ChangeRoute (Method, i, j, k) is better
            Go to While
          End If
        Next Method
      Next k
    Next j
  Next i
End While
```

O denotes a sequence (1, 2, 3, ..., *o*) of cities include depot city. *i*, *j*, and *k* are the index of sequence *O* that denote three edges. The parameters *i*, *j*, *k* denote three connections of sequence *O* that represent three edge connection of sequence *O* as depicted in Fig. 1. Started from these points, the sequence of *O* will be change. 3-Opt algorithm changes the order of the sequence from edges to edges following the seven methods as depicted in Fig. 2. Those methods are implemented from function *Method1* to *Method7*. Usually *O* in 3-Opt algorithm is initiated by random sequence but in HPSO, *O* is initiated with *gbest* resulted from NEH-PSO algorithm with first index is depot city. The better route is searched in function *ChangeRoute*. If a new route searched in a method has better fitness, it changes the last route, and the process is repeated until there is no change. The fitness function of 3-Opt algorithm contains three strategies of selecting products that are brought to the cities from depot city. These strategies will be explained in the next section.

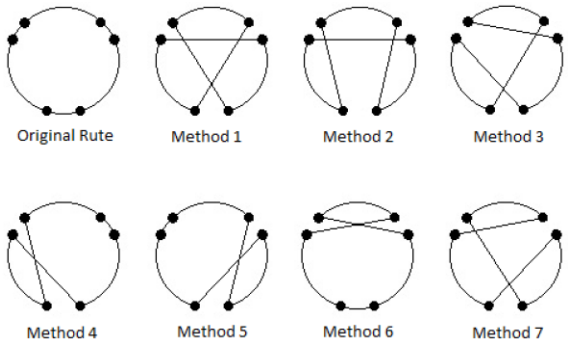

Figure. 2 Seven new connections in the 3-Opt algorithm transferred from the original connection

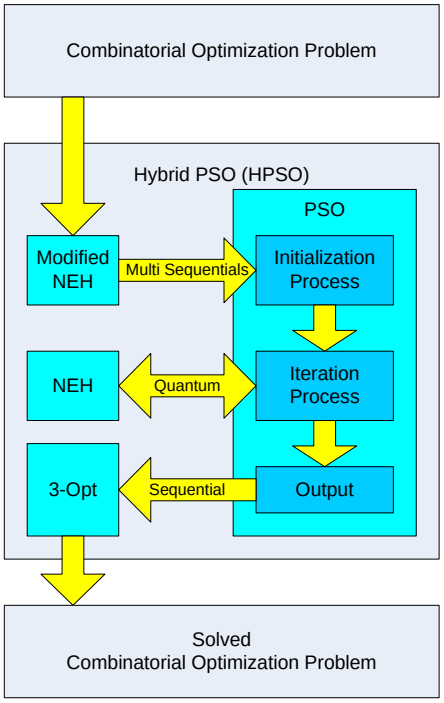

Figure. 3 Combination model of NEH, PSO, 3-Opt algorithm

A combination of the NEH, PSO, and 3-Opt algorithms was developed to find a solution for VRP. The combination of the three algorithms is named hybrid PSO (HPSO). It is a new combination model of hybrid metaheuristics to solve combinatorial optimization such as VRP. The HPSO model is depicted in Fig. 3.

The first process in HPSO is to combine NEH and PSO (NEH-PSO). These two algorithms will work together to improve each other's performance. NEH gives good results but for complex cases, it cannot achieve global optimum. The series of particles in the NEH will be improved by the PSO to produce a better fitness value. On the other hand, the PSO algorithm is improved by the NEH algorithm in the initialization process. Initialization of PSO that was previously carried out randomly will be processed by NEH. NEH is also used as quantum if the current PSO process does not produce better results after several iterations of calculations. If there is no improvement in the fitness value after several iterations, it indicated that the process was being trapped in a local optimum. With a quantum process, the population will escape from the local optimum. The quantum leap, which is usually done randomly [24], in this study will be carried out in a measured manner by the NEH algorithm. A 3-Opt algorithm is used to improve the results obtained from the NEH-PSO.

PSO initialization process is usually obtained randomly for every population. In this study, the PSO initialization process is obtained from the NEH calculation. NEH algorithm was modified and developed to provide more than one population series provided for each PSO population. This new algorithm is called Modified NEH (M-NEH). This modification will change the original NEH algorithm and provide a new algorithm as follows:

```
Count fitness of every particle and put to sequence S
sort by fitness from bad to good
Get one particle p from S and put in sequence N1
For i = 2 to n particle p in S
  Do While Ni exist
    For position pos=1 to number particle in N+1
      If fitness(ppos) < bestfitnessxi
        Clear all Ni+1 with the same Ni
        xi = 1
        Put pi in Ni+1, xi with parent Ni
      End If
      If fitness(ppos) = bestfitnessxi
        xi = xi +1
        Put pi in Ni+1, xi with parent Ni
      End If
    End For
    Put pi to sequence N in position best
      fitnessxi position
  End While
End For

Sort N order by fitness
For i = 1 to population X
  Set population Xi = Ni
End For
```

NEH only considers this condition: fitness(p_{pos}) < $bestfitness_{xi}$. This condition will provide only one solution of NEH is processed. Otherwise, M-NEH by added this condition: fitness(p_{pos}) = $bestfitness_{xi}$ could provide more than one condition because it consider another solution if the new sequential give the same fitness value. The process will generate a new branch of searching. X is sequence (1, 2, ..., x) obtained from N that will be used to initialize the population of PSO. This population setting needs a process to convert the sort of N to X. x particle of X is a real number and the amount of x denotes the order of x compare with another particles.

This modification method can be used in other metaheuristics algorithms to achieve an opportunity solution if one solution branch is trapped in the local optimum. In this case, every branch of searching in M-NEH is used to initialize PSO's population.

The quantum process on the PSO is carried out in the middle of the PSO calculation iteration. Quantum is running if there is no change of *gbest* value after q iteration. In this condition, the PSO process may be trapped in the local optimum. Quantum leaps are used in the entire PSO population. If the quantum process provided a better fitness value than before the quantum process, then the current position of the population as a result of the quantum process by NEH was determined as the position of the population to be processed in the next iteration.

The final result of the HPSO calculation is a sequence of particles that provide the best VRP fitness value. The results of this sequence will be refined with the 3-Opt algorithm to produce a better fitness value than the NEH-PSO. The effectiveness of this metaheuristic combination was tested to solve the Vehicle Routing Problem (VRP) with various number of the city, one product type and one transportation compartment. The HPSO is compared with the NEH and PSO algorithms. They will solve the problem with the same parameter values. This experiment is not to search for the optimum solution but to analyze the results after some iterations. Each will process it in just 1000 iterations. The results are shown in Table 1.

NEH-PSO algorithm provides better value than the PSO or NEH algorithm. The 3-Opt algorithm will

Table 1. Result of HPSO

Cities	PSO	NEH	NEH-PSO	HPSO
C76	2152.86	1081.11	982.96	913.74
C51	1135.25	646.49	591.64	569.87
C33	1106.16	904.50	871.04	858.03
C30	718.27	613.48	520.81	517.02
C23	724.02	643.57	612.76	604.84
C22	481.32	460.04	401.54	395.11

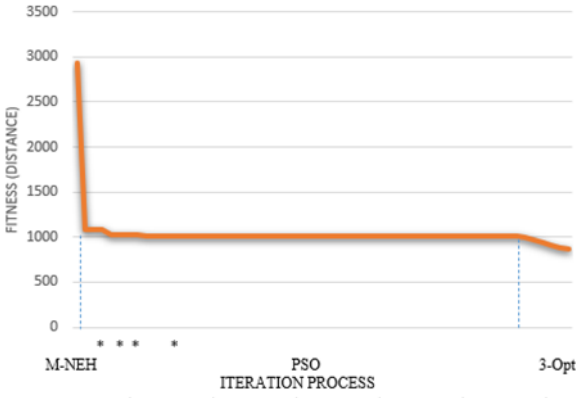

Figure. 4 HPSO processes Eil76

NEH-PSO algorithm provides better value than the PSO or NEH algorithm. The 3-Opt algorithm will improve the performance of the NEH-PSO algorithm, especially on complex data. In the VRP case, HPSO will provide a sequence of particles, namely port cities that must be served in bulk product shipments. The sequence of port cities is the route of delivery. The fitness value is calculated from the distance traveled by ship starting from the depot port to the ports to be served until it returns to the depot port again.

One of the VRP processes is depicted in Fig. 4. HPSO process problem of 76 cities. On the initial process, the M-NEH processes fitness value from 2927.65 to 1081.11. This process fixes up to half of the fitness value. PSO process fitness value from 1081.11 to 1006.66. Some processes are trapped in the local optimum (*) and NEH improve it by jumping from the local optimum. Finally, 3-Opt improves the fitness value to 871.34.

4. Experimental results

VRP in bulk product shipments from cities at depot ports to the cities in other ports is more complex than the original VRP case. VRP in bulk product shipments has more than one type of product to deliver. Each ship has more than one compartment. Each compartment contains only the same product because one product type and another cannot be mixed. But an undedicated compartment can be filled with different product types at another delivery. The VRP completion process does not consider the inventory status of ports. VRP case will become the Inventory Routing Problem (IRP) when considering the inventory status of each port.

The fitness value that determines the effectiveness of product delivery from the depot port to completion and back to the depot port again is the distance traveled by ship. In subsequent calculations, the distance can be converted into the cost of the ship trip or travel time. In this study, fitness is calculated based on the distance traveled by ships to serve ports. A ship in its assignment can serve several port cities. This will make the calculation process to find the closest distance become more complex. Calculations cannot be performed using an exact method. Metaheuristics will solve this optimization problem.

In real conditions, the VRP case is complex. They are more complex than standard VRP cases with one product and one compartment. The VRP level of complexity will be higher if there is more than one bulk product required by port cities. The limited number and capacity of compartments also increases the level of complexity of the VRP. In this study, the number of compartments is smaller than the type of product sent to the ports.

VRP completion process with multiple products and compartments can be achieved by combining a metaheuristics algorithm and strategy selection. Metaheuristics are used to find the best order. Strategies are used to determine the best way of selecting product types and compartments. The HPSO metaheuristics method will provide a sequence of port cities to pass through. In the HPSO calculation process, this sequence will adjust itself, therefore, the fitness value will move towards the optimum point. With the existence of several types of products and compartments, there are several strategies for selecting which product type and how many products that will be brought by the ship. The strategy is calculated in the HPSO fitness calculation process. Several strategies were selected and combined with the HPSO to find the best strategy. At least three strategies will be chosen in the calculation. Each strategy will be calculated and thus the best strategy will be identified. Every algorithm of strategies is divided into two processes. The first process is taking demand of cities and put into the compartments. The second process is putting products in every compartment into the demand of cities.

The first transport strategy (S1) is each compartment of the ship that will bring different product type from the other compartments in one transport journey. The type of product specified in a compartment is selected sequentially from the route

of port cities. This sequent route is obtained from HPSO process. The number of compartments is smaller than the type of product, therefore, the ship will return to the city if some product types have not been served yet. This strategy will even out the product type. The algorithm for the first strategy is shown as follows:

```
Set start = 1
While not stop process
  'First process
  Move to Ci_start and take its demand
  i = start
  While fill compartment
    For o=1 to number of products
      Put De_io in Co with same product
      Put rest of De_io in empty Co
        if De_io not in Co
    End j
    If every Co is full then end while
    Else move i to next city
    If i > number or cities then end while
  End while

  'Second process
  i = start
  While Co is not empty
    For o=1 to number of products
      For m=1 to number of compartments
        Put Co_mo in De_io
      Next m
      If all De_io fulfilled and start=i then
        start = 0
      End if
      If not all De_io fulfilled and start>0 then
        start = i
      End if
    Next o
    Move i to next city that has demand
    If Co empty then end while
  End while
  If there is no demand then stop process
End while
```

Ci is a sequence (1, 2, ..., *c*) of cities. *c* denotes the number of cities without depot city. This sequence of cities is obtained from HPSO process. *Co* denotes sequence (1, 2, ..., *o*) of compartments. These compartments are undedicated therefore every compartment can be filled with different products at different delivery times. De_i denotes product demand of city *i*.

Process will be executed until the demands are fulfilled completely. After the process finish, the algorithm is divided into two processes. The first process is filling the compartment with demand of the cities. It is started from city with index *start* (Ci_{start}). *start* will be changed every time any city is not serviced. *start* is set into the index of this city and filling process will be started from this position. Whenever a product is requested, the process searches the appropriate product type in each compartment. If the same compartment was found, then the product will be added to that compartment. If it is not found, the process will look for an empty compartment. If it is found, the compartment will be added with the product. *cap* is defined as the remainder capacity of compartment that can be filled. If *cap* is bigger or equal than the demand (De_{io}), Co_o will be updated as in Eq. (6). Otherwise, it will be updated as in Eq. (7)

$$Co_o = Co_o + De_{io} \tag{6}$$

$$Co_o = Co_o + cap \tag{7}$$

The Eqs. (6) and (7) indicate the demand of city *i* that bring product *o* (De_{io}) will be reduced as many as *cap* otherwise the compartment of product *o* (Co_o) will be added as many as *cap*. This process will be repeated until all compartments are full or all cities have been traced (command: *While fill compartment*). If there is no more compartments to carry the product, the city will be considered as the city to be visited in the next shipping after ship return to the depot.

The second process is taking the product from a compartment by the demand of cities. This process will be repeated until all compartments are empties. It is started from city with index *start*. On the product taking process, the product *o* of *i's* demand (De_{io}) is placed in compartment *m* (Co_{mo}). If Co_{mo} is bigger or equal than De_{io}, the variables are updated as shown in Eqs. (8) and (9).

$$Co_{mo} = Co_{mo} - De_{io} \tag{8}$$

$$De_{io} = 0 \tag{9}$$

Otherwise, the variables are updated as shown in Eqs. (10) and (11).

$$De_{io} = De_{io} - Co_{mo} \tag{10}$$

$$Co_{mo} = 0 \tag{11}$$

This process is repeated until all or part of demand De_i is fulfilled depending on the availability of *Co*. If there is demand that cannot be fulfilled and *start = 0*, the ship will return to this city (*start = i*). Otherwise, if all demands of the city are fulfilled and *start* is this city then set *start* = 0. Taking process will be continued to the next city from the sequence that has demanded and fulfills the demand of this city

from compartments. If compartments are empty, then return to the first process. If there is no demand from cities, then the process is stopped.

In the second strategy (S2), different compartments can be used to bring the same type of product. The first demand for a type of product will be served first. If one compartment is full to bring that type of product, another compartment can be used for the same product. If there is a type of product that cannot be served, the ship will return to the city after returning to the depot. This second strategy can streamline the transportation of products that are frequently requested. The second strategy algorithm is shown as follows:

```
Set start = 1
While not stop process
  'First process
  Move to Ci_start and take its demand
  i = start
  While fill compartment
    For o=1 to number of products
      Add De_io to Buffer_o
      If Buffer_o>capacity of Buffer_o
        Search empty Co  and put Buffer_o in Co
      End if
    Next o
    If every Co_o is full then end while
    Else move i to next city
    If i > number of cities then end while
  End while
  Put the remainder of Buffer_o in Co_o that can be filled

  'Second process
  i = start
  While Co is not empty
    For o=1 to number of products
      For m=1 to number of compartments
        Put Co_mo in De_io
      Next m
      If all De_io fulfilled and start=i then
        start = 0
      End if
      If not all De_io fulfilled and start>0 then
        start = i
      End if
    Next o
    Move i to next city that has demand
    If Co is empty then end while
  End while
  If there is no demand then stop process
End while
```

The first process is filling the compartment. $Buffer_o$ is used to store the number of demands of city i for the same product o (De_{io}). If the *Buffer* has exceeded the capacity (*cap*) of an empty compartment (Co_{empty}), the compartment will be filled with products from the $Buffer_o$. Therefore, if there is a product that is more frequently ordered by port cities, the compartments will be filled with that product type. This allows a type of product to occupy more than one compartment. The updating variable of filling empty *Co* is shown as in Eqs. (12) and (13). The second process is the same with strategy S1.

$$Co_{empty} = cap \tag{12}$$

$$Buffer_o = Buffer_o - cap \tag{13}$$

The third strategy (S3) occurs as in the second strategy, between one compartment to another can bring the same type of product. In this third strategy, the most requested product from the entire port cities will be served first. The ship will return to the city that requests other types of products that have not been served. This strategy has a benefit if there is a product that was frequently asked to be served first. The algorithm of the third strategy is shown as follows:

```
Set Buffer_o = Sum(De_o)
Sort Buffer_o

Set start = 1
While not stop process
  'First process
  For o=1 to number of products
    While Buffer_o is not empty
      For j=1 to number of compartments
        If Coj is empty
          Move Buffer_o to Co_j
        Endif
      Next j
      If every Co is full then end while
    End while
    If every Co is full then exit for
  Next o

  'Second process
  i = start
  While Co is not empty
    For o=1 to number of products
      For m=1 to number of compartments
        Put Co_mo in De_io
      Next m
      If all De_io fulfilled and start=i then
        start = 0
      End if
      If not all De_io fulfilled and start>0 then
        start = i
      End if
    Next o
    Move i to next city that has demand
    If Co is empty then end while
  End while
```

If there is no demand then stop process
End while

The first process of strategy S3 is filling the compartment. *Buffer* is used to store the number of demands for the same product. Unlike strategy S2, the *Buffer* is filled with all the demands that are grouped based on product type. The updating value of this variable is shown as in Eq. (14).

$$Buffer_o = \sum_{j=1}^{c} De_{jo} \quad (14)$$

c is the number of cities. The *Buffer* is sorted by the number of every demand. It will be moved to the empty compartment (Co_{empty}) one by one from one type of product to the other types. If $Buffer_o$ is greater than a capacity of compartment (*cap*) then the updating values are shown is shown in Eqs. (15) and (16).

$$Co_{empty} = cap \quad (15)$$

$$Buffer_o = Buffer_o - cap \quad (16)$$

Otherwise, it is shown in Eqs. (17) and (18).

$$Co_{empty} = Buffer_o \quad (17)$$

$$Buffer_o = 0 \quad (18)$$

If all of the compartments (*Co*) are full then the process continue to the second process which is the same as strategy S1.

The fitness process of VRP will process all of the three strategies from the sequential cities provided by HPSO. The calculation will select which strategy is better than others.

The ship assigned to deliver bulk products in this study has three undedicated compartments. Each compartment can be filled with a different type of product only in the different shipments. Any compartments besides the last shipment must be filled to avoid wasted costs due to empty compartments. The effect of this is that the city's demand can be partially fulfilled. This can also increase service to the cities by avoiding long shortages of stock. The capacity of each compartment is 100 units. There are 4 types of products (A, B, C, D) that are requested by port cities. Therefore, the number of compartments is smaller than the number of product types. There are ten port cities. City 1 is a depot serving shipping to the other port cities. The city coordinates are shown in Table 2.

Table 2. City coordinates

City	X	Y
1	100	100
2	80	70
3	120	30
4	40	80
5	50	150
6	20	30
7	130	120
8	70	150
9	120	140
10	150	110

Table 3. City demands

City	A	B	C	D
2	10		70	
3	80	20	10	10
4				40
5		20		50
6	70	50	30	
7	30	30	60	
8	10	40	60	40
9	60	40	60	
10	70	40	50	

The product demand for each port city is shown in Table 3. Product A is set to have more demand than others. The goal is to get closer to the real conditions because some products are ordered more than other products. Not all product types are demanded by every port city. The number of demands is also different for each city.

The product demand for each port city is shown in Table 3. Product A is set to have more demand than others. The goal is to get closer to the real conditions because some products are ordered more than other products. Not all product types are demanded by every port city. The number of demands is also different for each city.

The process was running 1000 iteration with 20 populations of PSO. Learning factors $c1$ and $c2$ of PSO were set to one with inertia weight was set to 0.9. The steps of the calculation process were started with initiation. These steps are described as follows:

Step 1 – Search S

The initiation process was processed by M-NEH algorithm. Sequence *S* was searched. The first process was to fill *S* with sort of every city according to the distance from depot city to this city in the descending order. The result of this process is shown in Table 4.

Step 2 – Search N

Table 4. Sequence *S*

Index	City	Distance
1	6	106.3
2	3	72.8
3	5	70.71
4	4	63.25
5	8	58.31
6	10	50.99
7	9	44.72
8	7	36.06
9	2	36.06

Table 5. Sequence *N* converse to population of PSO

Index	City	Convert to PSO
1	10	-1
2	7	-0.75
3	9	-0.5
4	5	-0.25
5	8	0
6	4	0.25
7	6	0.5
8	3	0.75
9	2	1

The cities were taken from *S* one by one to *N* and were sorted by index. The city was put in *N* by trying all positions from the sequence of particles that were already formed in N. All fitness was calculated for every position and the position with the best fitness was taken. If there was the same fitness function, then the search process would be branched. The new city occupied that position as a new form. This process was repeated until all cities in the *S* move to *N*. The initiation process from the available data coincidentally produced only one sequence of *N*. Sequence *N* was converted to one of the populations of PSO. This sequence was converted into a double number from -1 to 1. The result of this process is shown in Table 5. The other populations were developed randomly.

The next steps were PSO process. These steps are described as follows:

Step 3 – Searching pbest and gbest

Every population was stored in value and *pbest*. Value of population is sequence *present* (1, 2, ..., *c*). *present* was sorted and its fitness was searched. This fitness function contained 3 strategies. This function selected the best strategy that provided the best objective, namely the shortest route.

Table 6. Fitness of population

Population	Fitness	Population	Fitness
*p*1	1321.07	*p*11	1226.55
*p*2	1512.69	*p*12	1578.00
*p*3	2053.38	*p*13	1607.78
*p*4	1885.56	*p*14	1783.18
*p*5	1658.10	*p*15	1768.92
*p*6	2033.62	*p*16	1577.24
*p*7	1808.42	*p*17	1641.77
*p*8	1504.36	*p*18	1462.85
*p*9	1669.54	*p*19	1902.97
*p*10	1639.81	*p*20	1982.54

pbest was the best fitness of every population. For the first iteration, the fitness of *present* become *pbest*. *gbest* was the best fitness of *pbests*. For the first iteration *gbest* was obtain from population *p*1 because population *p*1 become *gbest*. It could have happened since it was developed from M-NEH algorithm while the others were developed randomly. The fitness of populations is shown in Table 6.

Step 4 – Quantum process

The quantum process is a new method was developed in this research. This process evaluated the value of *gbest*. If in any iterations *gbest* does not change, it is estimated to be at the local optimum. All of the population have to jump from its original position. This quantum process is done by processing all populations with NEH algorithm. The position of populations becomes initiation data for NEH. If there is an improvement in *gbest* value from a population, the new positions of the population were used, otherwise, the last data was. From the available data, there was a quantum going on to improve *gbest* value from 1238.17 to 1124.33 on iteration 66.

Step 5 – Searching v_t and present$_t$

The next step was changing the value of the velocity (v_t) and the position (*present*$_t$) of populations. Eq. (4) calculates that the v_t will move from its origin position according to the reference points namely *pbest* and *gbest*. Eq. (5) calculates the extent to which *present*$_t$ will move from its point of origin based on v_t. The step will return to *Step 3* until a maximum iteration is reached.

The next step is 3-Opt processing. *gbest* produced by NEH-PSO was improved by 3-Opt algorithm. This algorithm is effective enough to improve the big number of cities but for little data as this example data

with 10 cities NEH-PSO has given a good solution.

Step 6 – 3-Opt

O as initiation data of 3-Opt usually is developed randomly if this algorithm was used alone. But the initiation data can be developed from the result of other heuristic or metaheuristic algorithms. In this research, initiation data of 3-Opt was obtained from *gbest* of NEH-PSO. Depot city was added to the sequence *O* on the first index. During the 3-Opt process, the index position of every particle in sequence *O* changed. Sequence *O* was processed with *Method1* until *Method7*. For this available data, 3-Opt did not change the position of sequence *O*. *gbest* produced by NEH-PSO has achieved optimum solution.

The sequence of cities provided by HPSO is (7, 6, 3, 2, 5, 8, 9, 10, 4). The total distance of the route was 1107.71. The fitness value on the iteration process according to optimum searching is depicted in Fig. 5. Case of VRP was processed by 3 steps: initialization by M-NEH, PSO algorithm, and 3-Opt algorithm. Initialization process by M-NEH give fitness reduction from 1550.36 to 1321.07. The optimum solution was achieved after iteration 207 out of 1000 in PSO algorithm. 3-Opt algorithm did not improve fitness value since optimum value had been achieved in PSO algorithm. The result of the calculation was obtained in strategy S2. The schedule of transshipment with the optimum solution is shown in Table 7.

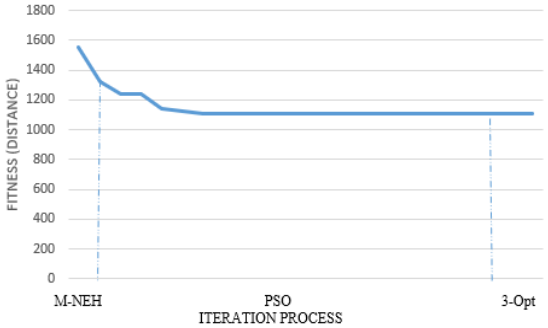

Figure. 5 Iteration process of result value

Table 7. Schedule of transshipment.

No	City	Dist.	Compartment (types of products – amount)			Product							
						A		B		C		D	
			1	2	3	dm	ac	dm	ac	dm	ac	dm	ac
1	7	36.06	(A) 100	(B) 100	(C) 100	30	30	30	30	60	60		
2	6	142.13	(A) 70	(B) 70	(C) 40	70	70	50	50	30	30		
3	3	100	(-)	(B) 20	(C) 10	80		20	20	10	10	10	
4	Depot	72.8	(-)	(-)	(-)								
5	3	72.8	(A) 100	(C) 100	(D) 100	80	80					10	10
6	2	56.57	(A) 20	(C) 100	(D) 90	10	10			70	70		
7	5	85.44	(A) 10	(C) 30	(D) 90			20				50	50
8	8	20	(A) 10	(C) 30	(D) 40	10	10	40		60	30	40	40
9	Depot	58.31	(-)	(-)	(-)								
10	5	70.71	(B) 100	(A) 100	(C) 100			20	20				
11	8	20	(B) 80	(A) 100	(C) 100			40	40	30	30		
12	9	50.99	(B) 40	(A) 100	(C) 70	60	60	40	40	60	60		
13	10	42.43	(-)	(A) 40	(C) 10	70	40	40		50	10		
14	Depot	50.99	(-)	(-)	(-)								
15	10	50.99	(A) 30	(B) 40	(C) 40	30	30	40	40	40	40		
16	Depot	50.99	(-)	(-)	(-)								
17	4	63.25	(D) 40	(-)	(-)							40	40
18	Depot	63.25	(-)	(-)	(-)								
		1107.71											

dm = Demand
ac = Accepted

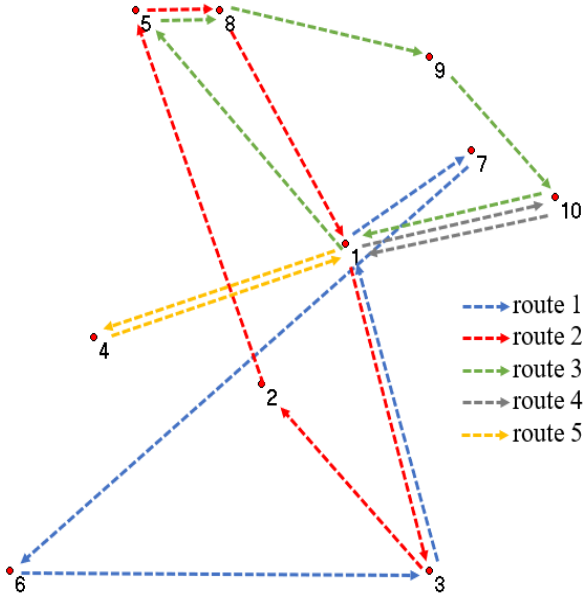

Figure. 6 Routes of shipping

At first, the three-ship compartments only brought products A, B, and C. The demands of cities 7 and 6 could be fulfilled directly. Demand for city 3 has not been fulfilled for product A and D. Demand for city 3 was fulfilled in the second delivery as seen at the fifth line after the ship returned from the depot and brought product A, C, and D. In the second delivery, the ship was fulfilled demands of city 3 and 2. Product B in city 5, product B and C in city 8 that have not been fulfilled were fulfilled after the ship returned to the depot. All of the demand from city 9 and part of the demand from city 10 was fulfilled after this delivery. The rest of the demand of city 10 was fulfilled after the ship returned to the depot. Although not all of the compartments were fulfilled with the product, the demand of city 4 could be fulfilled after returned to the depot because the product type of the demand from city 4 was different from city 10. The ship had to return four times to the city depot to fulfill all demands. The routes of shipping to deliver bulk products is depicted in Fig. 6.

delivered to city 3 first instead of city 6, it would take a longer distance to complete city 6 after the ship returned to the depot. Route 4 and route 5 only served one city since cities 10 and 4 have different types of products.

Data of case was developed until 15 datasets with different demands of cities. All of the cities have the same coordinate but different demand product. All demand data shared a common pattern that some products were ordered more than the other products. The detailed data can be seen in this site: https://notes.its.ac.id/siswanto/. The detailed data shows routes of the ship, the kind of product to be brought and the number of it. The result of calculation of datasets is depict as in Table 8.

Table 8. Calculation of 30 data

Data	Distance	Routes	Strategy
D1	1107.71	5	S2
D2	1100.82	5	S2
D3	1114.10	5	S1
D4	1199.76	5	S1
D5	1273.54	5	S1
D6	1287.91	5	S2
D7	1184.67	4	S2
D8	1118.42	5	S2
D9	1151.00	5	S1
D10	945.49	4	S2
D11	1161.85	5	S2
D12	1090.63	4	S1
D13	1216.72	5	S1
D14	1117.44	5	S1
D15	1135.07	5	S1

Datasets were resolved with 4 or 5 delivery routes. Most of them used strategy S1 to select products that were carried in compartments. It means that in one delivery there were no similar products. The other data used strategy S2 to select products. In one shipping some compartments could bring similar products. Strategy S3 was never used to delivery products. It means that bringing dominant products first cannot give an optimum solution since the demands of cities have such a pattern. In other condition strategy S3 could be used since the objective considered dominant products with special conditions must be delivered first.

HPSO can be used to solve general VRP especially other type of bulk product shipments with the same conditions. One of real instance data to test the algorithm is a maritime shipment in transporting cement products [25]. These kinds of products are produced in a city named of which the node is defined as a depot port and delivered to the other port cities either in the same island or the other islands by several ships. It has two types of products namely Portland Composite Cement (PCC) and Ordinary Portland Cement (OPC) [25]. The demands and distances between ports which are symmetrical are shown as in Table 9.

The cement company has four ships to deliver its products from depot city to the customer cities namely V1, V2, V3, and V4. Every ship has different number and capacity of compartment. All compartments are undedicated compartments which mean it can be used to bring different type of cements

Table 9. Demand of cement industry and distances between cities

City	Demand (Ton)		Distance (Nautical Mile)									
	PCC	OPC	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
C1	0	0	0	744	432	163	274	361	301	634	768	882
C2	719	279	744	0	369	616	617	928	606	382	150	449
C3	811	0	432	369	0	573	680	728	711	336	417	558
C4	527	0	163	616	573	0	153	376	170	774	713	1021
C5	221	0	274	617	680	153	0	352	78	881	715	1023
C6	1030	0	361	928	728	376	352	0	412	930	1025	1178
C7	610	538	301	606	711	170	78	412	0	913	703	1011
C8	204	0	634	382	336	774	881	930	913	0	321	361
C9	302	0	768	150	417	713	715	1025	703	321	0	374
C10	462	0	882	449	558	1021	1023	1178	1011	361	374	0

Table 10. Route distance of shipments

Ship	Compartment Capacity (Ton)		Average Route Distance (Nautical Mile)						Effectiveness (%)
	1	2	NEH	PSO	3-Opt	TS	GA	HPSO	
V1	1500	0	6738.00	6324.57	6526.80	6521.67	6423.17	6324.53	0.00
V2	5200	0	5030.00	4888.13	4908.83	4998.30	4934.80	4878.00	0.21
V3	4000	3500	4237.00	3953.13	3982.80	4149.73	4001.57	3912.00	1.04
V4	3400	2500	3893.00	3688.53	3711.87	3939.00	3791.27	3553.00	3.67
Total distance			19898.00	18854.36	19130.30	19608.70	19150.81	18667.53	0.99

on the other shipments. Several ships have one, and the others have two compartments. HPSO is used to solve this problem and compared its result with its single metaheuristics (NEH, PSO, 3-Opt) as well as other metaheuristics namely TS and GA. The calculation results are shown in Table 10.

The effectiveness is used to compare the performance of the algorithm of which formulation as shown as in Eq. (19).

$$Ef = \frac{D_2 - D_1}{D_2} \times 100\% \quad (19)$$

Ef denotes the effectiveness of HPSO calculation result. D_1 denotes average route distance of HPSO calculation result while D_2 denotes the one of a ship calculated by second-best metaheuristics. The effectiveness is calculated from the deviation between the second-best metaheuristics and HPSO. The effectiveness shows how much the reduction in the route distance obtained by HPSO that is compared to the second-best algorithm. The effectiveness column shows HPSO providing increased effectiveness over PSO. This proves that the addition of the NEH process at the initialization and quantum and 3-Opt at the end of the PSO will improve the performance of this metaheuristic to solve VRP.

The experiments of every metaheuristic are executed 30 times. Results of the experiments are average of final solution. The result shows that HPSO has better solution compared with other metaheuristics either for every ship or total ships belong to the company. It is shown by the distance route solutions that HPSO has 0.99% more effective than the second best and about 3,42% than the average total distance of five other metaheuristics. PSO is in second place while the TS and GA algorithms do not give better results than PSO. And even in some cases 3-Opt give better result than TS and GA.

5. Conclusion

HPSO is quite effective to solve VRP cases with multiple products and compartments. HPSO provides the best sequence of cities that a ship must pass to fulfill the demands of these cities. The fitness value in the HPSO calculation is the distance of the route.

HPSO is combined with the strategy of bulk products selection in the undedicated compartment of the ship. The strategy considers what products are transported and which compartments are filled with what products. There are three strategies proposed in this study to find the best fitness value. Every strategy considers the pattern of products demand. HPSO gives better solution to solve real data of cement product transportation compared with other metaheuristics. It solves more effective by giving shortest route distances for every ship used and the total distance to deliver the product.

Implementation of complex VRP solutions with multiple products and limited compartments should consider more than one strategy. In the next research another strategy can be developed by considering variations of product demands. The number of depot cities also can be considered more than one to pick up products from the nearest depot. This new method is basic solution when the VRP is developed into IRP. In this problem, the inventory status of every city can be considered to avoid the lack of products in every port. The objective of the solution can consider distance and delivery time which may be converted into delivery cost.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Antono Adhi, Budi Santosa, and Nurhadi Siswanto conveyed the idea of the system; Antono Adhi designed the experiment; Antono Adhi and Nurhadi Siswanto conducted formal analysis, investigation, and data preparation; Antono Adhi and Nurhadi Siswanto prepared and wrote the original draft; Antono Adhi, Nurhadi Siswanto and Budi Santosa reviewed and edited the revised draft; Antono Adhi, Nurhadi Siswanto and Budi Santosa analyzed and verified the research results and findings.

Acknowledgments

The authors wish to acknowledge the funding of this research is from the Ministry of Education, Culture, Research and Technology, Republic of Indonesia through the Doctor's Dissertation Research Scheme Grant No. 1264/PKS/ITS/2020.

References

[1] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem", *Manage. Sci*, Vol. 6, pp. 80-91, 1959.

[2] D. Zhang, S. Cai, F. Ye, Y. Si, and T. T. Nguyen, "A hybrid algorithm for a vehicle routing problem with realistic constraints", *Information Science*, 394-395, pp. 167-182, 2017.

[3] N. Niazy, A. E. Sawy, and M. Gadallah, "A hybrid Chicken Swarm Optimization with Tabu Search algorithm for solving capacitated Vehicle Routing Problem", *International Journal of Intelligent Engineering and Systems*, Vol. 13, No. 4, pp. 237-247, 2020.

[4] T. G. Crainic and G. Laporte, "Fleet management and logistics", *Springer Science & Business Media*, 2012.

[5] M. Fisher, "Vehicle routing", *Handbooks in Operations Research and Management Science*, Vol. 8, pp. 1-33, 1995.

[6] G. Laporte, M. Gendreau, J. Y. Potvin, and F. Semet, "Classical and modern heuristics for the vehicle routing problem", *International Transactions in Operational Research*, Vol. 7, pp. 285-300, 2000.

[7] P. Toth and D. Vigo, "An overview of vehicle routing problems", *In The Vehicle Routing Problem*, pp. 1-26, 2001.

[8] B. L. Golden, S. Raghavan, and E. A. Wasil, "The vehicle routing problem: latest advances and new challenges", *Springer Science & Business Media*, 2008.

[9] Y. Marinakis and A. Migdalas, "Annotated bibliography in vehicle routing", *Operational Research*, Vol. 7, pp. 27-46, 2007.

[10] G. Laporte, "Fifty years of vehicle routing", *Transportation Science*, Vol. 43, pp. 408-416, 2009.

[11] J. K. Lenstra and A. Kan, "Complexity of vehicle routing and scheduling problems", *Networks*, Vol. 11, pp. 221-227, 1981.

[12] E. Yakici, "A heuristic approach for solving a rich min-max vehicle routing problem with mixed fleet and mixed demand", *Computer & Industrial Engineering*, Vol. 109, pp. 288-294, 2017.

[13] V. F. Yu, A. A. N. P. Redi, C. Yang, E. Ruskartina, and B. Santosa, "Symbiotic organisms search and two solution representations for solving the capacitated vehicle routing problem", *Applid Soft Computing*, Vol. 52, pp. 657-672, 2017.

[14] V. F. Yu, A. A. N. P. Redi, Y. A. Hidayat, and O. J. Wibowo, "A simulated annealing heuristic for the hybrid vehicle routing problem", *Applied Soft Computing*, Vol. 53, pp. 119-132, 2017.

[15] P. Grangier, M. Gendreau, F. Lehuédé, and L. Rousseau, "A metaheuristic based on large neighborhood search for the vehicle routing

problem with cross-docking", *Computers and Operations Research*, Vol. 84, pp. 116-126, 2017.
[16] B. Santosa, R. Damayanti, and B. Sarkar, "Solving multi-product inventory ship routing with a heterogenous fleet model using a hybrid cross entropy-genetic algorithm: a case study in Indonesia", *Production & Manufacturing Research*, Vol. 4, No. 1, pp. 90-113, 2016.
[17] N. Siswanto, D. Essam, and R. Sarker, "Multi-heuristics based genetic algorithm for solving maritime inventory routing problem", In: *Proc. of IEEE International Conference on Industrial Engineering and Engineering Management*, pp. 116-120, 2011.
[18] M. Nawaz, E. E. Enscore, and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem", *The International Journal of Management Science*, Vol. 11, No. 1, pp. 91-95, 1983.
[19] J. Kennedy and C. Eberhart, "Particle swarm optimization", In: *Proc. of IEEE Int'l Conf. on Neural Networks*, Vol. IV, pp. 1942-1948, 1995.
[20] B. Santosa and P. Willy, "Metoda metaheuristik konsep dan implementasi", *Guna Widya*, 2011.
[21] J. C. Bansal, P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon, and A. Abraham, "Inertia weight strategies in particle swarm optimization", In: *Proc. of Third World Congress on Nature and Biologically Inspired Computing*, pp. 640-647, 2011.
[22] M. Mahi, Ö. K. Baykan, and H. Kodaz, "A new hybrid method based on Particle Swarm Optimization, Ant Colony Optimization and 3-Opt algorithms for Traveling Salesman Problem", *Applied Soft Computing*, Vol. 30, p. 484-490, 2015.
[23] A. S. Alfa, S. S. Heragu, and M. Chen, "A 3-OPT based simulated annealing algorithm for vehicle routing problems", *Computer ind. Engng*, Vol. 21, pp. 635-639, 1991.
[24] M. R. Singh and S. S. Mahapatra, "A quantum behaved particle swarm optimization for flexible job shop scheduling", *Computers & Industrial Engineering*, Vol. 93, pp. 36-44, 2016.
[25] M. Rusdianto and N. Siswanto, "Maritime inventory routing problem with undedicated compartments: a case study from cement industry", *IOP Conf. Series: Materials Science and Engineering*, pp. 1-9, 2020.