Malaysian Journal of Mathematical Sciences 13(3): 311–328 (2019)

MALAYSIAN JOURNAL OF MATHEMATICAL SCIENCES

Journal homepage: http://einspem.upm.edu.my/journal

Harmony Search for Multi-depot Vehicle Routing Problem

Misni, F.[2,3] and Lee, L.S.*[1,2]

[1]*Laboratory of Computational Statistics and Operations Research, Institute for Mathematical Research, Universiti Putra Malaysia, Malaysia*

[2]*Department of Mathematics, Faculty of Science, Universiti Putra Malaysia, Malaysia*

[3]*Faculty of Industrial Sciences & Technology, Universiti Malaysia Pahang, Malaysia*

E-mail: lls@upm.edu.my

* *Corresponding author*

Received: 31 August 2018

Accepted: 29 July 2019

ABSTRACT

This study considers the multi-depot vehicle routing problem in supply chain network design. Vehicle routing is one of the important system in supply chain. The objective is to find the minimum distance travelled by the vehicles, from the depot to the customers. We proposed an improved harmony search algorithm for solving this problem. Firstly, the Clarke & Wright saving algorithm is used for the initialisation of a solution vector in harmony search. Three simple heuristics approaches; swapping, insertion and relocation are proposed as the local optimisation techniques during the implementation of the harmony search. The roulette wheel approach is implemented for the selection procedure. Computational experiments are conducted using the benchmark dataset of Cordeau's

problem instances. Computational results show that the proposed harmony search algorithm is comparable to other metaheuristic approaches from the literature.

Keywords: Harmony search, supply chain network design, multi-depot vehicle routing.

1. Introduction

In a supply chain network design problem, vehicle routing of the goods and services is a major concern for the decision maker. The main problem is to coordinate the product flows between the facilities. The network flows of the supply chain can be a forward or reverse flows that involves several facilities such as suppliers, manufacturers, distribution centres and customers. In forward flow, the materials and products are delivered from the point of origin to the end of user. Whereas, reverse flow of materials and products are delivered starting from the point of consumer to the secondary user through the collection center and processing center. At the processing center, the used product will be either recycled, re-manufactured, repaired or reused based on the condition of the product.

One of the concern issues of vehicle routing in supply chain network design is the multi-depot vehicle routing problem (MDVRP). The problem consists of several depots and a set of customers to be served by each depot. Each vehicle will starts and ends at the same depot, and each customer will be served once by only one vehicle. The objective is to optimise the route travelled by the vehicles from the depot to the customers that satisfy the constraints in term of customer's demands and the capacity limit of the vehicles and the depots. Figure 1 shows an example of MDVRP.

The MDVRP is a NP-hard combinatorial optimisation problem. Many algorithms were developed to tackle the problem such as exact method (branch and bound algorithm (Christofides et al., 1981)), heuristic method (Clarke & Wright saving algorithm (Clarke and Wright, 1964)) as well as metaheuristic approaches (simulated annealing (Kirkpatrick, 1984), tabu search (Glover, 1977), genetic algorithm (Holland, 1992), ant colony optimisation (Dorigo et al., 1996), and particle swarm optimisation (Poli and Kennedy, 2007)). Since the problem is NP-hard, exact methods are not suitable to obtain the optimal solution within a reasonable time especially for the large problem instances. Hence, the metaheuristic approaches are more practical to be used.

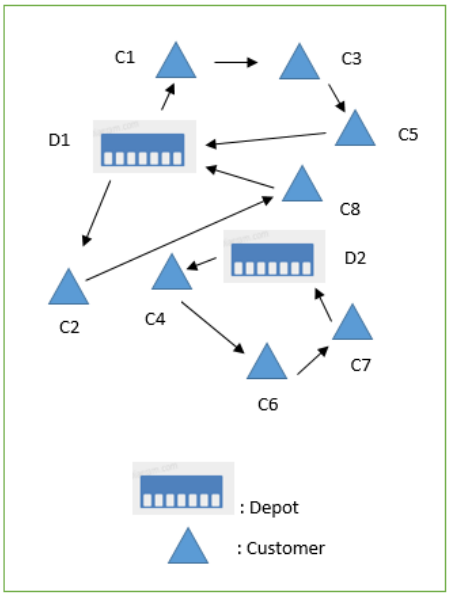

Figure 1: An example of MDVRP with 2 depots, 8 customers and 3 vehicles

In this paper, a modified harmony search (MHS) is proposed to solve the MDVRP and the benchmark dataset from Cordeau is used for validation of the method. The HS is a population-based metaheuristic approach that mimics the music improvisation of music orchestra, first proposed by Geem et al. (2001).

Kawtummachai and Shohdohji (2000) hybridised the HS with local improvement process called two phase selection process to solve green VRP. They used roulette wheel to improve the selection process. Huang et al. (2010) solved the VRP with time windows in fourth-party logistics by using a standard HS. They compared the results with the enumeration algorithm and produced satisfactory solutions.

Pichpibul and Kawtummachai (2013) proposed a modified HS for capacitated VRP. They incorporated Clarke & Wright saving algorithm into the initial solution and used roulette wheel selection to improve the selection process. The results show that their method is competitive to the best existing algorithm. Hosseini et al. (2014) hybridised the HS with simulated annealing to solve transportation problem of cross docking and milk run logistics. The solutions outperformed the solutions from the GAMS/CPLEX in term of cost and computational time. Yassen et al. (2015) proposed two HS algorithms

which are HSA-optimiser and HSA-solver. HSA-solver is the hybrid of HS and local search which solved the problem by configured the input generated from the HSA-optimiser.

Recently, Chen et al. (2017) solved a dynamic VRP with time windows using HS algorithm and variable neighborhood descent (VND) algorithm as a method for global exploration and local search capability respectively. Maleki et al. (2017) used hybrid self-adaptive global best HS for solving VRP with time windows. They adapted six local search neighborhood structure to enhance the exploitation capability and showed that their method outperformed the standard HS. To the best of our knowledge, there is no research done in MDVRP that applied the HS algorithm.

2. Mathematical Formulation

The MDVRP consists of a set of customers and depots. The number and location of customers and depots are predetermined. The demands at each customer is also known and it can be either constant or not. The objective and conditions are given below:

Set:
I = sets of all depots $(i = 1, 2, \ldots, I)$
J = sets of all customers $(j = 1, 2, \ldots, J)$
K = sets of all vehicles $(k = 1, 2, \ldots, K)$

Input parameter:
D_j = demand per customer j
d_{ij} = distance from i to j
V_k = capacity of vehicle k
Max_i = capacity limit of depot i
N = number of vehicles

Decision variables:

$$y_{ij} = \begin{cases} 1, & \text{if depot } i \text{ assign to customer } j \\ 0, & \text{otherwise.} \end{cases}$$

$$x_{ijk} = \begin{cases} 1, & \text{if arc } (i,j) \text{ is travelled by vehicle } k \\ 0, & \text{otherwise.} \end{cases}$$

Objective Function:

$$\text{Min } z = \sum_i \sum_j \sum_k d_{ij} x_{ijk} \tag{1}$$

subject to:

$$\sum_i \sum_k x_{ijk} = 1, \forall j \in J \tag{2}$$

$$\sum_i \sum_j x_{ijk} \leq 1, \forall k \in K \tag{3}$$

$$\sum_i \sum_j D_j x_{ijk} \leq V_k, \forall k \in K \tag{4}$$

$$\sum_j x_{ijk} - \sum_j x_{jik} = 0, \forall i \in I, k \in K \tag{5}$$

$$\sum_j D_j y_{ij} \leq Max_i, \forall i \in I \tag{6}$$

$$U_{ik} - U_{jk} + N x_{ijk} \leq N - 1, \forall i \in I, j \in J, k \in K \tag{7}$$

$$\sum_u x_{iuk} + x_{ujk} - y_{ij} \leq 1, \forall i \in I, j \in J, k \in K \tag{8}$$

$$y_{ij} \in \{0, 1\} \tag{9}$$

$$x_{ijk} \in \{0, 1\} \tag{10}$$

$$U_{ik} \geq 0, \forall i \in I, k \in K. \tag{11}$$

The objective of MDVRP is to find the minimum total distance travelled by the vehicles. Constraints in Eqn.(2) and (3) indicate that, each of the customer has to be assigned at a single route and it can be served by only one vehicle. The total demands at each route cannot exceed the vehicle capacity limit and the vehicle must start and end at the same depot. These two constraints are shown in Eqn.(4) and (5), respectively. Besides vehicle capacity limit, the capacity constraint for depot is given in Eqn.(6). Eqn.(7) represents the new sub tour elimination constraint and Eqn.(8) specified that the customer will be assigned to the depot if there is a route from that depot. The binary values on decision variable and the positive values for auxiliary variable are defined in Eqn.(9), (10), and (11), respectively.

3. Standard Harmony Search

The HS is a population-based metaheuristic inspired by the music improvisation. When musicians improvise the harmony, they usually choose these three options: (1) play any pitch from the memory, (2) play something similar to any other in their memory, or (3) compose new or random notes (Geem et al., 2001). The process of searching for optimal solutions is analogous to

this efficient search for a perfect state of harmony. Figure 2 shows the analogy between the music improvisation and the optimisation problem.

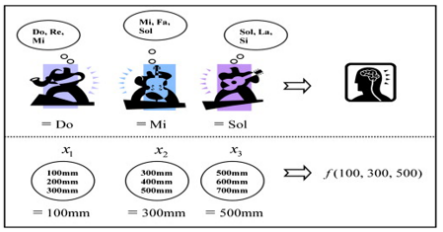

Figure 2: Analogy between music improvisation and optimisation problem (Geem et al., 2001)

Let us consider an orchestra performance consisting of a saxophone, a double bass and a guitar. Each of these music instruments has its own range of pitch. For example, the instrument pitch for saxophone is *Do, Re, Mi*, the double bass is *Mi, Fa, Sol* and the guitar is *Sol, La, Si* respectively. If the saxophonist chooses randomly *Do*, the double bassist chooses *Mi* and the guitarist chooses *Sol*, the new harmony *Do, Mi, Sol* is made. If this new harmony is better than the worst in the harmony memory (HM), then the worst harmony will be replaced by the new harmony. This process is repeated until a perfect harmony is found. In real optimisation, each musician is replaced by a decision making variable and the instrument pitch is the range of each variable that can be chosen. For each variable, they should choose one potential value within the range and combine altogether to make a new solution vector. If the new solution vector is better than the worst vector in the HM, then the new vector replaces the worst one. This process is repeated until a stopping criterion for termination is reached or the optimal solution is found. Below are the basic steps of a standard HS (SHS) algorithm:

Step 1: Problem formulation

To apply the SHS on a problem, it must be formulated as an optimisation problem with the objective function and a set of constraint. The SHS algorithm searches the solution space to find the optimal solution vector $x = (x_1, x_2, ..., x_n)$ that optimise (minimise or maximise) the objective function. If the decision variables have discrete values, the set of possible values is given by $x_i \in X_i = \{x_i(1), x_i(2), ..., x_i(K)\}$ where K_i is the number of different values in the definition space for variable i. If the variables have continuous values, the set of possible values is given by $x_i^L \leq x_i \leq x_i^U$ where L and U is the lower bound and upper bound, respectively.

Step 2: Parameter setting
Once the problem formulation is ready, the parameters of the algorithm must be set with values. The parameters setting used in SHS algorithm are harmony memory considering rate (HMCR), pitch adjusting rate (PAR), harmony memory size (HMS), stopping criterion and bandwidth (bw) that operate altogether with PAR in pitch adjustment.

HMCR is important to ensure that good solutions are considered as element of new solutions. If it is too low, only few good solutions are selected and convergence may be slow and if it is too high, other alternatives may not well explored, resulting in not so good solutions. Therefore, in order to use the memory effectively, the value of HMCR should be in between 0.7 and 0.95 (Yang, 2010).

A small value of PAR together with narrow value of bw can cause the convergence of the SHS algorithm to be slow, given the limitation in exploration to a single portion of the search space. However, a high value of PAR can cause solutions to disperse around a few potential optimal as in random search. For these reasons, usually the value of PAR is around 0.1 and 0.5 and the bw generally bounded between 1% and 10% of all the range of variable values (Yang, 2010).

Step 3: Initialise the memory
To create the initial memory, several solutions are generated randomly and the number of solutions should be at least equal to the HMS. HM can be described as the following matrix:

$$HM = \left[\begin{array}{cccc|c} x_1^1 & x_2^1 & \cdots & x_n^1 & f(x^1) \\ x_1^2 & x_2^2 & \cdots & x_n^2 & f(x^2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_1^{HMS} & x_2^{HMS} & \cdots & x_n^{HMS} & f(x^{HMS}) \end{array}\right],$$

where
x_i^j = a decision variable for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots,$HMS,
$f(x^j)$ = fitness function for $j = 1, 2, \ldots,$HMS.

Step 4: Improvisation
The following is the process of improvisation in SHS algorithm.

1) *Random selection*: When SHS determine the new solution x^{new}, it randomly chooses a value from the range of all possible values $x_i \in X_i = \{x_i(1), x_i(2), ..., x_i(K)\}$ or $x_i^L \leq x_i \leq x_i^U$ with a probability (1−HMCR).

2) *Memory consideration*: When the probability equals to HMCR, the new solution, x^{new} is chosen randomly from the HM. The random number may be calculated using a uniform distribution $U(0, 1)$.

3) *Pitch adjustment*: After the value of new solution has been randomly chosen from the HM, it may be adjusted to neighbouring values with probability PAR. For discrete variable, $x^{new} = x_i(k+n)$ where $n \in \{-1, 1\}$ and k is the position of a chosen solution in HM. For continuous variable, the new solution vector will be $x^{new} = x_i + \triangle$ where $\triangle = U(-1, 1) \times bw(i)$.

Step 5: Memory update
If the new solution, x^{new} is better than the worst solution in HM in terms of the objective function value, the new solution will replace the worst solution. The HM will be updated and sorted according to the value of the objective function.

Step 6: Termination
If the SHS algorithm meets the stopping criterion such as the maximum iteration or maximum execution time, the process will be terminated.

4. Modified Harmony Search (MHS)

In order to improve the solution quality, a MHS with some modifications is proposed and described as in the following steps.

Step 1: Problem formulation
In MDVRP, a set of available customers to be served is assigned as a variable and each variable in a set of solution vector should not be repeatable and identical to each other. The solution vector represents the sequence of the customers for delivery and separated by the value of zero for different depots. The objective function is the total travelling distance between nodes in the

sequences.

Due to the depot capacity limit, the customers are clustered into the depots. The assigned customers of each depot are based on their distance to the depot, which is calculated using the Euclidean distance formula as in Eqn.(12).

$$dist(d,c) = \sqrt{(x_d - x_c)^2 + (y_d - y_c)^2}, \tag{12}$$

where
d = depot
c = customer
(x_d, y_d) = location of depot
(x_c, y_c) = location of customer.

The process of clustering is done based on the following condition:

if dist(c,d_1) $\leq$ dist(c,d_2)
    assign customer c to depot 1
else
    assign customer c to depot 2
end

Example:
Solution vector: [1 3 5 2 8 0 4 6 7] (0 = depot)
Depot 1: [1 3 5 2 8]
Depot 2: [4 6 7]

Then, at each depot, the customer's routes are formed according to their demand and the vehicle capacity limit.

Example:
Depot 1: [1 3 5 | 2 8]
Route 1: D1 $\rightarrow$ 1 $\rightarrow$ 3 $\rightarrow$ 5 $\rightarrow$ D1
Route 2: D1 $\rightarrow$ 2 $\rightarrow$ 8 $\rightarrow$ D1

Depot 2: [4 6 7]
Route 1: D2 $\rightarrow$ 4 $\rightarrow$ 6 $\rightarrow$ 7 $\rightarrow$ D2

Step 2: Parameter setting
In SHS algorithm, HM represents the population of the solutions and HMS is the size of the solutions in that population. Another parameter which is *new*HMS is added into the proposed MHS as the size of the newly generated solutions. In MHS, the parameter used in solving MDVRP are shown in Table 1:

Table 1: Parameter setting for PHS

Parameter	Setting
HMS	300
*new*HMS	20
HMCR	0.85
PAR	0.25

Step 3: Initialise the memory
For SHS, all the solution vectors in the population are permuted randomly to create the initial HM, and the number of solutions are equal to HMS. In order to increase the chances of finding the optimal solution in a reasonable time, we implemented Clarke & Wright saving algorithm to generate a solution vector in the initial population of the MHS. The procedures of the method are described as follows:

Step 1: For each pair of customers in each depot, calculate the saving (Eqn.(13)) for all possible pairs of customers:

$$S_{ij} = d_{i0} + d_{j0} - d_{ij}, \tag{13}$$

where
S_{ij} = saving of customer i to j
d_{i0} = distance from customer i to depot
d_{j0} = distance from customer j to depot
d_{ij} = distance from customer i to customer j.

Step 2: Sort the saving in descending order.
Step 3: Start merge the route from the top of saving. While merging, the following process should be fulfilled:

a. to merge the route, make sure the nodes are directly connected.
b. two nodes can be combined into a same route if the total demand are not exceeded the vehicle limit capacity.

Step 4: Improvisation
Generally, a single solution is generated at each iteration, but in the MHS, several solutions are produced to create a new HM. The size of the new HM is set to be less than the HMS. In order to generate the new HM, the following processes are conducted.

1) *Random selection*: The new solution vector is produced by randomly permutation of the set of available customers with a probability of (1−HMCR).

2) *Memory consideration*: The proportional selection (or roulette wheel selection) is used to select the new solution vector x^{new} in the HM with a probability equals to HMCR. It depends on the absolute fitness values of any solution compared to the absolute fitness values of other solutions in HM. The generated solution that relies on the survival of the fittest value will increase the chance of getting a good solution for HM and gives the better results (Al-Betar et al., 2012). The selection probability p_i and cumulative probability q_j for the solution i and j is calculated by the Eqn.(14) and (15), respectively:

$$p_i = \frac{f(x^i)}{\sum_{i=1}^{HMS} f(x^i)}, \tag{14}$$

$$q_j = \sum_{i=1}^{j} p_i, \tag{15}$$

where
$f(x^i)$ = objective function.

The selection of solution depends on the generated random number from the range of 0 and 1. If the random number is within the range of the cumulative probability, $q_j \leq rand \leq q_{j+1}$, solution $j+1$ is chosen as the next solution. For minimisation problem, the formulation of calculating p_i is different. Since the fitness values in the HM are sorted in ascending order, the fitness function

should be inverted into $1/f(x^i)$.

3) *Pitch adjustment*: Once the new solution is produced, the adjustment of neighbouring structure in a form of local optimisation is applied to enhance the exploration and exploitation capabilities of the algorithm. In MHS, the local optimisation of swapping, insertion and relocation are used based on the values of PAR.

There are two techniques for swapping; swapping two customers within the depot and between the depots. The techniques of swapping are shown below:

Example:

Depot 1: | 1 | 2 | 3 | 5 | 8 |

Depot 2: | 4 | 6 | 7 |

After swapping:

Swapping customers within the depot:

Depot 1: | ***5*** | 2 | 3 | ***1*** | 8 |

Depot 2: | ***7*** | 6 | ***4*** |

Swapping between the depots:

| 1 | 2 | 3 | 5 | ***6*** |

| 4 | ***8*** | 7 |

The implementation of single adjustment is not capable enough to explore the solution space. Therefore, another adjustment technique is added which is the insertion technique. This technique is applied by inserting the selected customer in the route of the same depot.

Example:

Depot 1: | 1 | 2 | 3 | 5 | 8 |

Depot 2: | 4 | 6 | 7 |

After insertion:

Depot 1: | 1 | ***8*** | 2 | 3 | 5 |

Depot 2: | 4 | ***7*** | 6 |

For relocation, a customer will be relocated from a depot to another depot.

Example:

Depot 1: | 1 | 2 | 3 | 5 | 8 |

Depot 2: | 4 | 6 | 7 |

After relocation:

	Relocation from depot 1 to depot 2:	Relocation from depot 2 to depot 1:
Depot 1:	1 | 2 | 3 | 5	1 | 2 | 3 | 5 | 8 | ***7***
Depot 2:	4 | 6 | 7 | ***8***	4 | 6

Step 5: Memory update
To update the memory, the HM and new HM are combined together and sorted according to the value of objective function. Then, the best solutions with the size of HMS are kept to be used for next iteration.

Step 6: Termination
The stopping criteria of the MHS is 50 non-improving iterations.

5. Results and Discussion

Both the SHS and MHS are coded in MATLAB R2017b software and the Cordeau's problem instances (see Table 2) from VRP database (`http://neo.lcc.umaa.es/vrp`) are used to validate the proposed method. Each of the problem instances has different number of depots, number of customers, vehicle

capacity limit and number of available vehicles at each depot.

MDVRP with the same benchmark instances have been solved by other metaheuristic approaches such as tabu search and genetic algorithm. There have been found in Gillett and Johnson (1976) (GJ), Chao et al. (1993) (CGW), Renaud et al. (1996) (RBL) and Cordeau et al. (1997) (CGL). All of these papers used tabu search algorithm but different approaches in term of parameter setting and local optimisation. The used of GA in MDVRP are found in Genetic Cluster GA (GenClust) by Thangiah and Salhi (2001) and GA using weighted sum fitness evaluation (GA-WS) and GA using Pareto ranking (GA-P) by Ombuki-Berman and Hanshar (2009) .

For each problem instances, both SHS and MHS are ran for 5 times and the best solution is reported in column 9 and 10 of Table 3, respectively. From the results in Table 3, it shows that the MHS is superior than SHS. It can be seen that, the MHS also outperformed the results in GenClust, (GA-WS) and (GA-P) except in P02 instances where MHS produced slightly higher distance than the GenClust GA. However, when compared to the tabu search heuristics, MHS outperformed the solutions in GJ for all the instances, but not in CGW, RBL and CGL. The best known solutions of the Cordeau's instances are found in CGW, RBL and CGL.

Table 2: Cordeau's problem instances

	P01	P02	P03	P04	P05	P06
No. of depots	4	4	5	2	2	3
No. of customers	50	50	75	100	100	100
No. of vehicles per depot	4	2	3	8	5	6
Vehicle capacity	80	160	140	100	200	100

Table 3: The comparison of MHS with SHS, and other metaheuristics

Cordeau's instances	GenClust	GA-WS	GA-P	GJ	CGW	RBL	CGL	SHS	MHS
P01	591.73	622.18	600.63	593.2	576.9	576.87	576.87	627.29	586.12
P02	463.15	480.04	480.04	486.2	474.6	473.53	473.53	528.79	474.71
P03	694.49	706.88	683.15	652.4	641.2	641.19	641.19	705.07	650.13
P04	1062.38	1024.78	1034.59	1066.7	1012.0	1003.87	1001.59	1242.54	1024.10
P05	754.84	785.15	778.01	778.9	756.5	750.26	750.03	973.28	754.57
P06	976.02	908.88	916.71	912.2	879.1	876.50	876.50	1017.34	903.44

The percentage deviation of the obtained solution (OS) in MHS and SHS with the best known solution (BS) are shown in Table 4. The formulation of percentage deviation (P_d) is given in Eqn.(16).

$$P_d = \frac{OS - BS}{BS} \times 100. \tag{16}$$

Table 4: The percentage deviation of MHS and SHS with best known solution

Cordeau's instances	Best known solution	MHS	Percentage deviation MHS (%)	SHS	Percentage deviation SHS (%)
P01	576.87	586.12	1.60	627.29	8.74
P02	463.15	474.71	2.50	528.79	14.17
P03	641.19	650.13	1.39	705.07	9.96
P04	1001.59	1024.10	2.25	1242.54	24.06
P05	750.03	754.57	0.61	973.28	29.77
P06	876.50	903.44	3.07	1017.34	16.07

The value of percentage deviation is to show the range gap between the solution obtained and best known solution. If the percentage is very small, it means that the solution obtained is good since it is closer to the best known solution. The percentage deviation of MHS and best known solution are less than 3.1% for all the problem instances. This shows that the MHS can give satisfactory and competitive results when solving the MDVRP. The implementation of roulette wheel selection, Clarke & Wright saving algorithm and the multi-adjustment in local optimisation gave significant impact to the computational results.

6. Conclusion

In this paper, a MHS algorithm with some modifications on initialisation, selection process and local optimisation is developed. For initial population, a solution vector is created with Clarke & Wright saving algorithm to increase the chances of getting better solution. Roulette wheel approach has been used for selection of new solution in HM. In order to avoid getting trapped in a local optimum, different techniques of adjustment are implemented. It can be applied by swapping the customers either within or between the depots, inserting the customer in the same route, or relocating the customer to a different depot.

Six Cordeau's problem instances in MDVRP have been used to validate the proposed MHS. The experimental analysis is carried out and the results obtained in MHS are compared with SHS and other metaheuristic approaches from the literature. The results showed that the MHS is superior than the SHS, the three GA methods and tabu search in GJ. However, when compared to the

best known solutions found in CGW, RBL and CGL, there are still some gaps between them. Hence, further research are needed on the MHS algorithm for MDVRP. Although the existing methods in CGW, RBL and CGL performed better than MHS, proposed MHS is said to be significant and comparable to those methods since there is no research found in MDVRP using harmony search and the percentage deviations of MHS with best known solutions are relatively small which is less than 3.1%.

For further investigation, the MHS algorithm can be improved by hybridised it with other metaheuristic approaches such as genetic algorithm, simulated annealing, tabu search or particle swarm optimisation especially in the process of initialisation or local optimisation procedure.

Acknowledgements

This research was supported by the Geran Putra-Inisiatif Putra Siswazah (GP-IPS/2017/9579400) funded by Universiti Putra Malaysia (UPM).

References

Al-Betar, M. A., Doush, I. A., Khader, A. T., and Awadallah, M. A. (2012). Novel selection schemes for harmony search. *Applied Mathematics and Computation*, 218(10):6095–6117.

Chao, I.-M., Golden, B. L., and Wasil, E. (1993). A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. *American Journal of Mathematical and Management Sciences*, 13(3-4):371–406.

Chen, S., Chen, R., and Gao, J. (2017). A modified harmony search algorithm for solving the dynamic vehicle routing problem with time windows. *Scientific Programming*, 2017.

Christofides, N., Mingozzi, A., and Toth, P. (1981). Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20(1):255–282.

Clarke, G. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581.

Cordeau, J.-F., Gendreau, M., and Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks: An International Journal*, 30(2):105–119.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41.

Geem, Z. W., Kim, J. H., and Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search. *Simulation*, 76(2):60–68.

Gillett, B. E. and Johnson, J. G. (1976). Multi-terminal vehicle-dispatch algorithm. *Omega*, 4(6):711–718.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1):156–166.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Hosseini, S. D., Shirazi, M. A., and Karimi, B. (2014). Cross-docking and milk run logistics in a consolidation network: A hybrid of harmony search and simulated annealing approach. *Journal of Manufacturing Systems*, 33(4):567–577.

Huang, M., Bo, G., Wang, X., and Ip, W. (2010). The optimization of routing in fourth-party logistics with soft time windows using harmony search. In *Sixth International Conference on Natural Computation (ICNC2010)*, volume 8, pages 4344–4348. IEEE.

Kawtummachai, R. and Shohdohji, T. (2000). A hybrid harmony search (hhs) algorithm for a green vehicle routing problem (gvrp). In *Proceedings of the 4th International Conference on Engineering Optimization*, pages 573–578.

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34(5-6):975–986.

Maleki, F., Yousefikhoshbakht, M., and Rahati, A. (2017). A hybrid self-adaptive global best harmony search algorithm for the vehicle routing problem with time windows. *BRAIN. Broad Research in Artificial Intelligence and Neuroscience*, 8(4):65–84.

Ombuki-Berman, B. and Hanshar, F. T. (2009). Using genetic algorithms for multi-depot vehicle routing. In *Bio-inspired Algorithms for the Vehicle Routing Problem*, pages 77–99. Springer.

Pichpibul, T. and Kawtummachai, R. (2013). Modified harmony search algorithm for the capacitated vehicle routing problem. In *Proceedings of the International Multi Conference of Engineers and Computer Scientists*, volume 2.

Poli, R. and Kennedy, J.and Blackwell, T. (2007). Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57.

Renaud, J., Laporte, G., and Boctor, F. F. (1996). A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, 23(3):229–235.

Thangiah, S. R. and Salhi, S. (2001). Genetic clustering: an adaptive heuristic for the multidepot vehicle routing problem. *Applied Artificial Intelligence*, 15(4):361–383.

Yang, X.-S. (2010). *Nature-Inspired Metaheuristic Algorithms*. Luniver Press.

Yassen, E. T., Ayob, M., Nazri, M. Z. A., and Sabar, N. R. (2015). Meta-harmony search algorithm for the vehicle routing problem with time windows. *Information Sciences*, 325:140–158.