A Memory Integrated Artificial Bee Colony Algorithm with Local Search for Vehicle Routing Problem with Backhauls and Time Windows

Naritsak Tuntitippawan*
Department of Industrial Engineering, Faculty of Engineering, King Mongkut’s University of Technology North Bangkok, Bangkok, Thailand

Krisada Asawarungsaengkul
Operations Research and Engineering Management Research Center
Department of Industrial Engineering, Faculty of Engineering, King Mongkut’s University of Technology North Bangkok, Bangkok, Thailand

* Corresponding author. E-mail: nst@kmutnb.ac.th DOI: 10.14416/j.ijast.2018.03.001
Received: 1 June 2017; Accepted: 26 June 2017; Published online: 5 March 2018
© 2018 King Mongkut’s University of Technology North Bangkok. All Rights Reserved.

Abstract
The vehicle routing problem is a logistics problem which receives much attention in logistics management. This paper presents a Memory integrated Artificial Bee Colony Algorithm (MABC) to solve the Vehicle Routing Problem with addition of Backhauls and Time Windows, known as the VRPBTW. In VRPBTW, a homogenous fleet of vehicles are utilized to deliver goods to linehaul customer set and pick up goods from backhaul customer set. Vehicle capacity, sequence of linehaul/backhaul and time windows are the three of major constraints for this problem. The VRPBTW’s objective is to determine the optimal routes with minimum of total distance that satisfies all constraints. The proposed algorithm was tested on Gelinas’s VRPBTW benchmark problems. MABC is developed by adding the memory to Artificial Bee Colony (ABC). The local search algorithms including λ-interchange and 2-opt* are utilized to search for the better solutions. The computational results show that MABC significantly yields the good solutions in terms of total travelling distance. Finally, it can be concluded that the performance of the proposed MABC algorithm is superior to the existing studies in term of quality solution.

Keywords: Memory integrated artificial bee colony algorithm, Vehicle routing problem, Backhauls, Time windows, Local search

1 Introduction
Logistics management mainly focuses in cost, time, and quality. The Vehicle Routing Problem (VRP) is a vital operation for logistics. Fully utilization of the fleet of vehicle is a major concern for VRP. Assigning vehicle to pick up goods during backhaul can enhance utilization of the vehicle. In addition, the period for delivery and pick up goods is also the restriction in logistics since warehouse or customer usually schedule a specific period to pick up or receive goods. Vehicle Routing Problem with Backhauls and Time Windows (VRPBTW) is an extension of the classical Vehicle Routing Problem (VRP). The VRPBTW contains two subsets of customers, known as delivery customer (linehaul) and pickup customer (backhaul). Each linehaul customer requires a given quantity of goods from depot, and a given quantity of goods is collected from each backhaul customer and returned to the depot. Because deliveries are usually higher priority

than pickup; therefore, the linehaul customers must be visited before the backhaul customers in each route. Each customer must be visited within a specified time windows. The vehicle is allowed to arrive before the lower bound of time windows, and waits until the customer is available. However, the vehicle is not allowed to arrive after the upper bound of time windows. We consider this restriction as the hard time windows constraint for the vehicle routing problem with backhauls. Due to the finite capacity of each vehicle, the routes must be satisfied the capacity constrain. The objective is to minimize the total travelling distance of vehicles, while all constraints must be satisfied.

Literatures on logistics management can be found in [1], [2]. The decision making model on the excavated soil transportation with transportation cost and time constraint was proposed by [1], [2] presented the location routing problem which intended to minimize the transportation and depreciation cost of collection and distribution in a rubber market. In this paper, the vehicle routing problem is our interest. The VRPBTW is a NP-hard problem [3]. Because it can be considered as the VRP, which is well-known NP-hard problem, only backhauls and time windows are additionally considered. Research on VRPBTW has been mainly focused on heuristic and meta-heuristic approaches which can produce high quality of solutions within reasonable computational time. The review on metaheuristic for VRPBTW are provided as follows. A Genetic Algorithm (GA) with greedy route construction heuristic for VRPBTW was performed by [4]. A Tabu Search (TS) heuristic for VRPBTW was proposed by [5]. The result was compared with solution of other heuristic approach and the optimal solutions. An ant system was presented to solve the VRPBTW by [6]. This ant system approach was based on insertion algorithm proposed by [7]. A Guided Local Search Approach (GLSA) and section planning technique for VRPBTW were also presented by [8].

Recently, a Differential Evolution Algorithm (DEA) was presented by [9]. DEA was tested with benchmark problem and compared the results with the optimal and best known solutions. They could find some of the results that were better than the best known solutions. Moreover, an advanced Hybrid Metaheuristic Algorithm (HMA) was proposed by [10]. The HMA combines Simulated Annealing (SA) and Tabu Search (TS) meta-heuristic to solve VRPBTW. The HMA was applied to the benchmark problem and the results showed that HMA could also find better solution than the best known solutions. In addition, [11] proposed an Artificial Bee Colony (ABC) algorithm with \( \lambda \)-interchange local search technique for VRPBTW. The results shown that the proposed algorithm is comparable to HMA and GA.

In this paper, we present Memory integrated Artificial Bee Colony Algorithm (MABC) and additional local search techniques, which are \( \lambda \)-interchange and 2-opt* to reduce the time spent searching for solution and increase chances of finding better solution for VRPBTW. The Gelinas’s benchmark problems in [12] are used to evaluate the performance of the MABC algorithm.

2 Problem Definition

VRPBTW is formulated based on the existing mathematical formulation for VRP, with each vertex representing a customer [13]. Let \( G(V, A) \) be a graph with a vertex set \( V = \{0\} \cup L \cup B \), where the subset \( \{0\} \), \( L \) and \( B \) represent to the depot, linehaul customer nodes, and backhaul customer nodes, respectively. \( A \) denotes all possible arcs that are connected between nodes. \( G(\overline{V}, \overline{A}) \) be a graph in \( G \) which is defined as:

\[
\overline{A} = A_1 \cup A_2 \cup A_3 \quad {\text{and}} \quad \overline{V} = V
\]

\[
A_1 = \{(i, j) \in A : i \in L \cup \{0\}, j \in L\},
\]

\[
A_2 = \{(i, j) \in A : i \in L, j \in B \cup \{0\}\},
\]

\[
A_3 = \{(i, j) \in A : i \in B, j \in B \cup \{0\}\}
\]

The arc set \( \overline{A} \) can be divided into three subsets. The arc set \( A_1 \) represents all of arcs from the depot to linehaul customer nodes and from linehaul customer nodes to linehaul customer nodes. \( A_2 \) represents all of arcs from linehaul customer nodes to backhaul customer nodes and from backhaul customer nodes to the depot. \( A_3 \) represents all of arcs from backhaul customer nodes to backhaul customer nodes and from backhaul customer nodes to the depot. Additionally, for each \( i \in V \), \( \Delta_i^f = \{j : (i, j) \in \overline{A}\} \) and \( \Delta_i^b = \{j : (j, i) \in \overline{A}\} \) define the forward and backward star of \( i \), respectively. Forward star of \( i \) defines the set of vertices \( j \) that have direct path from vertex \( i \) and backward star of \( i \) defines the set of vertices \( j \) that have direct path to vertex \( i \).
Using these definitions, the VRPBTW was formulated as following and it is classified as the mixed-integer programming model [10].

Notations

\( K \) : Set of vehicles
\( d_i \) : Demand/supply for customer \( i \) : \( \forall i \in V \setminus \{0\} \)
\( c_{ij} \) : Distance from node \( i \) to node \( j \) : \( (i, j) \in A \)
\( t_{ij} \) : Travel time between node \( i \) to node \( j \) : \( (i, j) \in \mathcal{A} \)
\( a_i \) : Earliest arrival time at customer \( i \) : \( \forall i \in V \setminus \{0\} \)
\( b_i \) : Latest arrival time at customer \( i \) : \( \forall i \in V \setminus \{0\} \)
\( s_i \) : Service time at customer \( i \) : \( \forall i \in V \setminus \{0\} \)
\( u_k \) : Capacity of vehicle \( k \) : \( \forall k \in K \)
\( T_{max} \) : Maximum route time allowed for vehicles
\( M \) : Large penalty value

Decision variables

\( x_{ijk} \) : 1 if vehicle \( k \) travels from customer \( i \) to \( j \), and 0 otherwise
\( w_{ik} \) : Service start time of vehicle \( k \) for customer \( i \).

Objective

\[
\text{Min } z = \sum_{(i,j) \in A} \sum_{k \in K} c_{ij} x_{ijk} \tag{1}
\]

Subject to:

\[
\sum_{j \in V} d_i x_{ijk} \leq u_k \quad \forall k \in K \tag{2}
\]

\[
\sum_{i \in V} d_j x_{ijk} \leq u_k \quad \forall k \in K \tag{3}
\]

\[
\sum_{i \in V} x_{ijk} = 1 \quad \forall k \in K \tag{4}
\]

\[
\sum_{j \in V} x_{ijk} = 1 \quad \forall k \in K \tag{5}
\]

\[
\sum_{j \in V} x_{ijk} = 1 \quad \forall j \in V \setminus \{0\} \tag{6}
\]

\[
\sum_{j \in V} x_{ijk} = 1 \quad \forall i \in V \setminus \{0\} \tag{7}
\]

\[
\sum_{i \in V} x_{ijk} = \sum_{i \in V} x_{jik} \quad \forall j \in V \setminus \{0\}, \forall k \in K \tag{8}
\]

\[
w_{ik} + s_i + t_{ij} - w_{jk} \leq M(1 - x_{ijk}) \quad \forall i \in V \setminus \{0\}, \forall k \in K \tag{9}
\]

\[
w_{ik} + s_j + t_{ij} - w_{jk} \leq M(1 - x_{ijk}) \quad \forall i \in V \setminus \{0\}, \forall j \in V \setminus \{0\}, \forall k \in K \tag{10}
\]

\[
a_i \leq w_{ik} \leq b_i \quad \forall i \in V \setminus \{0\}, \forall k \in K \tag{11}
\]

\[
0 \leq w_{ik} \leq T_{max} \quad \forall k \in K \tag{12}
\]

\[
x_{ijk} \in \{0,1\} \quad \forall i \in V, \forall j \in V, \forall k \in K \tag{13}
\]

\[
w_{ik} \geq 0 \quad \forall i \in V, \forall k \in K \tag{14}
\]

According to the model, objective (1) is to minimize the sum of the route distance. Constraints (2) and (3) are the capacity restrictions which ensure that load of vehicle \( k \) will not exceed the capacity in linehaul and backhaul customers, respectively. Constraints (4)-(8) are the classic VRP constraints. Constraints (4) and (5) say that each route must leave from the depot and must return to the depot. Constraints (6) and (7) ensure that the exactly one arc enters and leaves each customer node. Constraint (8) states that a vehicle leaves from the same customer node it has entered. Constraints (9)-(13) define time windows that each customer must be serviced in the time windows and also these constraints prevent subtour. Finally, constraints (14) and (15) guarantee that all decision variables must be the proper variable types.

3 Proposed Algorithm

This section describes the MABC algorithm, generation of initial solution and local search operations which are \( \lambda \)-interchange and 2-opt*. These local searches improve the solution by exchanging customer nodes between 2 selected routes.

3.1 Memory integrated Artificial Bee Colony Algorithm (MABC)

The ABC algorithm is an evolutionary algorithm that is inspired by the natural foraging behavior of honey bees in finding food or nectar source around the hive. The solution of the problem are considered as food source and the groups of bees try to exploit the food sources in the hope of finding good quality nectar or high quality of solutions.
1. Generate a set of initial food sources (or initial solutions) \( X_i \).
2. Evaluate the fitness \( f(X_i) \) for each food source.
3. Set \( v = 0 \), \( l_i = 0 \), and \( MEM \) of \( X_i = \emptyset \), \( i = 1,2,\ldots,N_b \)
4. While \( (v < MaxIteration) \) do
   a. For each \( X_i \), generate \( X'_i \) by randomly select pair of routes \((r_n, r_j)\) that do not appear in \( MEM \) of \( X_i \) and apply two local search operations,
      \[ \text{If } f(X'_i) > f(X_i) \text{ Then } X_i = X'_i, l_i = 0 \text{ Else } l_i = l_i + 1 \text{ and add to } MEM \text{ of } X_i \]
      End If, Next
   b. Set \( f(\tilde{X}) = 0 \), \( F = \emptyset \)
   c. For each onlooker
      Select food source \( X_i \) by using roulette wheel selection method and generate \( X'_i \) by randomly select pair of routes \((r_n, r_j)\) that do not appear in \( MEM \) of \( X_i \) and apply two local search operations,
      \[ \text{If } f(X'_i) < f(\tilde{X}) \text{ Then } \tilde{X} = X'_i \text{ End If, Next} \]
   d. For each food source \( X_i \), \( \text{If } f(X_i) < f(\tilde{X}) \text{ Then } i \in F \text{ End If, Next} \)
      Set \( \tilde{X} = X_i \) where \( j = \arg\max_{i \in F \setminus \{i\}} \)
      For each food source \( X_i \), If \( X_i = \tilde{X} \) Then Replace \( X_i \) with \( \tilde{X}, l_i = 0 \), replace \( MEM \) of \( X_i \) with \( MEM \) of \( \tilde{X} \) and \( \text{Else } l_i = l_i + 1 \text{ End If, Next} \)
   c. For each food source \( X_i \), If \( l_i = \text{limit} \) Then Replace \( X_i \) with a randomly generated solution, \( l_i = 0 \), and set \( MEM \) of \( X_i = \emptyset \) End if, Next
      \( f, v = v + 1 \)
Loop

Figure 1: Memory integrated artificial bee colony algorithm for solving the VRPBTW.

In the ABC algorithm, the bees are divided into three types including employed bees, onlookers and scouts [14]. Employed bees are responsible for exploiting available food sources and gathering required information. The information is shared to onlookers then the onlookers select existing food source to be further explored. Employed bee can abandon the old food source when the onlookers can find the best food source. In this case, the employed bee associated with the old food source will be assigned to the best food source found by onlookers. However, any food sources will also be abandoned if the iteration limit is reached. After that, the employed bee becomes a scout bee to look for new food source randomly.

MABC algorithm which adopts ABC algorithm in [11] with addition of memory and local search can be performed by following step 1 to 4. Step 1 to 3 are for the initial solutions and parameters setting. Step 4 describes an improvement on solution (or food source) \( X_i \) by generating \( X'_i \) with local search processes including \( \lambda \)-interchange and 2-opt*. The best results from both local searches will be used to generate \( X'_i \). Next, the additional of memory (MEM) concept as in step 4 is proposed to avoid ABC algorithm to fall into the local optimal too fast. The memory can be considered as Tabu list or forbidden list. Additional of memory is used for memorizing the search paths in solution \( X_i \) that has been proceeded by local searches without any improvement. The MABC algorithm can be summarized as in Figure 1.

3.2 Initial solutions

An initial solution is constructed based on the weighted time oriented nearest neighbor heuristic process that was proposed by Solomon [7] and this technique was applied with ABC algorithm in [11]. The closet of node \( i \) to node \( j \), \( c_{ij} \) is introduced to generate the initial solutions appropriately.

In term of closest, \( c_{ij} \) is determined by three cost factors [15]. The first factor as in (16) is distance from customer \( i \)’s location to customer \( j \)’s location, \( d_{ij} \)

\[
  d_{ij} = \sqrt{(y_i - y_j)^2 + (x_i - x_j)^2}
\]

(16)

where \((x_i, y_i)\) is geometric location of customer \( i \).

The second factor is urgency, \( u_{ij} \) in (17). Urgency of customer \( j \) is time left to service customer \( j \) after it serves customer \( i \).
\[ u_j = b_j - (T_i + s_i + t_j) \]  \hspace{1cm} (17)

where \( T_i \) denotes the service start time of customer, \( s_i \) is defined as the service time of customer \( i \), \( t_j \) denotes travel time between customer \( i \)'s location to customer \( j \)'s location.

The third factor is the waiting time, \( w_j \) as in (18) which is the time remaining until the vehicle’s last possible service start.

\[ w_j = \max \{0, a_j - (T_i + s_i + t_j)\} \] \hspace{1cm} (18)

The closest term can be formulated as (19)

\[ c_j = \theta_d d_{ij} + \theta_u u_j + \theta_w w_j \] \hspace{1cm} (19)

where \( \theta_d, \theta_u, \theta_w \) are the weight of distance, urgency, and waiting time, respectively.

The lower value of \( c_j \) is more preferable. To generate different initial food sources, the initial weight of distance, urgency, and waiting time are randomly searched from range 0 to 1.

The procedure of this heuristic starts every routes by finding the unrouted customer closest (in term of \( c_j \)) to the depot. Next, calculate \( c_j \) for unrouted customer \( j \) to the last customer of each route \( i \) and also calculate \( c_j \) for unrouted customer \( j \) to the depot. Select the lowest \( c_j \) that satisfies both capacity and time windows constraint then add customer \( j \) to corresponding route. If closest to the depot, the new vehicle route is introduced. The approach is repeated until all customers are assigned.

### 3.3 \( \lambda \)-interchange local search

\( \lambda \)-interchange local search was introduced by [16]. It improves the solution by exchange of nodes between pair routes that are selected randomly. The number of node using exchange ranged from 0 to \( \lambda \). The exchange will be performed to all possible cases that can be exchanged. Then, the best result is selected after doing the \( \lambda \)-interchange local search. For example, if \( \lambda = 2 \) all possible exchange will have 8 cases: exchange with \((0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1) \), and \((2,2)\). Let route \( i \) and route \( j \) are selected routes to be performed the local search by exchange with operator \((1,2)\) which means that route \( i \) will use 1 node to exchange and route \( j \) will use 2 nodes to exchange. Moreover, assuming that route \( i \) has 6 nodes and route \( j \) has 4 nodes. In this case, there are 181 possible results then the best result will be the solution of this local search. Example of \( \lambda \)-interchange local search with operators (1,1) and (1,2) are shown in Figure 2.

### 3.4 2-opt* exchange local search

The 2-opt* procedure was proposed by [17]. It is able to improve the solution by exchange nodes in pair of routes that are randomly selected. After that select arc from selected route, for example, suppose that route \( i \) and route \( j \) are selected. Result of 2-opt* will occur by taking a node located next to selected arc on route \( i \) and taking a node located next to selected arc of route \( j \) and then exchange nodes between pair of routes. An example of 2-opt* procedure is depicted in Figure 3. Assuming that route \( i \) has 6 nodes and route \( j \) has 4 nodes, these lead to 35 possible results then the best result will be the solution of this local search.
3.5 Removing procedure for the pair of routes

Since the memory is integrated to ABC algorithm, the forbidden list of each food source is introduced. If route \((r_i, r_j)\) is added to forbidden list, it should be removed after a certain period of time. The removing procedure will release all pairs of routes that are consisted of \(r_i\) or \(r_j\) from the forbidden list (or MEM).

4 Computational Results

The proposed algorithm, MABC with \(\lambda\) -interchange and 2-opt * local search, was implemented via Visual Basic programming language using a laptop PC Intel Core i5, 2.3 GHz processor and 4 GB memory. To evaluate the performance, the instance of VRPBTW generated by Gelinas et al. [12] have been tested. These benchmark problems were selected from the first 5 problems of the r1 series developed by Solomon [7] and then randomly chooses 10, 30 and 50% of the 100 customer nodes to be backhaul customer, without any changes to the other attributes. Gelinas et al. generated additional test problems by considering only the first 25 and first 50 customer nodes. Therefore 45 test problems for the VRPBTW were generated. However, in this paper, we select 34 problems of which the optimal solutions are known to the computational experiments.

The parameters for MABC consist of: the Number of bees (\(Nb\)), limit, \(\lambda\)(Lambda) and maximum iteration (MaxIteration). Design of experiments was conducted by [11] in order to determine the appropriate parameters which are employed to these experiments. Therefore, the parameters for MABC are as following: \(Nb = N\_node/10\), limit = \(N\_node*5\), \(\lambda = 2\) and MaxIteration = 40,000. The number of customer is denoted by \(N\_node\).

Table 1 to Table 3 display the optimal solution, the best result of MABC, and also the compared algorithms which are ABC II, HMA, GA, DEA referring to the proposed algorithms in [11], [10], [3] and [9] respectively. The \(\%Gap\_opt\) for Gelinas’s benchmark problem sets with known optimal solution are shown as well. The \(\%Gap\_opt\) can be described by the following equation:

\[
\%Gap\_opt = \frac{\text{metaheuristic solution} - \text{optimal solution}}{\text{optimal solution}} \times 100\%
\]

In the Table 1, the results for 25 customer nodes, show that the proposed MABC is able to find the optimum solutions for 10 problems and yields 1 new best solution in problem R104a. The total distance of 8,432.1 and \(\%Gap\_opt\) of 0.32% for proposed MABC reveal that MABC is the best for small-size problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Optimal Solutions</th>
<th>MABC</th>
<th>ABCII</th>
<th>HMA</th>
<th>GA</th>
<th>DEA</th>
<th>%Gap&lt;sub&gt;opt&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>R101a</td>
<td>643.4</td>
<td>643.4</td>
<td>643.4</td>
<td>643.4</td>
<td>643.4</td>
<td>0.00%</td>
<td>0.00% 0.00% 0.00% 0.00% 0.00%</td>
</tr>
<tr>
<td>R101b</td>
<td>711.1</td>
<td>721.8</td>
<td>721.8</td>
<td>721.8</td>
<td>721.8</td>
<td>1.50%</td>
<td>1.50% 1.50% 1.50% 1.50% 1.50%</td>
</tr>
<tr>
<td>R101c</td>
<td>674.5</td>
<td>676.8</td>
<td>676.8</td>
<td>682.3</td>
<td>676.8</td>
<td>0.34%</td>
<td>0.34% 0.34% 1.16% 0.34% 0.34%</td>
</tr>
<tr>
<td>R102a</td>
<td>563.5</td>
<td>563.5</td>
<td>563.5</td>
<td>563.5</td>
<td>565.3</td>
<td>0.00%</td>
<td>0.00% 0.00% 0.00% 0.00% 0.32%</td>
</tr>
<tr>
<td>R102b</td>
<td>622.3</td>
<td>628.1</td>
<td>628.1</td>
<td>622.3</td>
<td>629</td>
<td>0.93%</td>
<td>0.93% 0.93% 0.00% 1.08% 0.00%</td>
</tr>
<tr>
<td>R102c</td>
<td>584.4</td>
<td>584.4</td>
<td>584.4</td>
<td>584.4</td>
<td>585.3</td>
<td>0.00%</td>
<td>0.00% 0.00% 0.00% 0.00% 0.15%</td>
</tr>
<tr>
<td>R103a</td>
<td>476.6</td>
<td>476.6</td>
<td>476.6</td>
<td>476.6</td>
<td>489</td>
<td>0.00%</td>
<td>0.00% 0.46% 0.00% 2.60% 0.00%</td>
</tr>
<tr>
<td>R103b</td>
<td>507</td>
<td>507</td>
<td>507</td>
<td>507</td>
<td>510.9</td>
<td>0.00%</td>
<td>0.00% 0.00% 0.00% 0.77% 0.00%</td>
</tr>
<tr>
<td>R103c</td>
<td>475.6</td>
<td>483</td>
<td>483</td>
<td>483</td>
<td>495</td>
<td>1.56%</td>
<td>1.56% 1.56% 1.56% 4.08% 0.00%</td>
</tr>
<tr>
<td>R104a</td>
<td>452.5</td>
<td>452.5</td>
<td>452.5</td>
<td>452.5</td>
<td>459.1</td>
<td>0.00%</td>
<td>0.29% 0.29% 0.07% 1.46% 0.00%</td>
</tr>
<tr>
<td>R104b</td>
<td>467.6</td>
<td>468.5</td>
<td>468.5</td>
<td>468.5</td>
<td>469.6</td>
<td>0.19%</td>
<td>0.19% 0.19% 0.19% 0.19% 0.43%</td>
</tr>
<tr>
<td>R104c</td>
<td>446.8</td>
<td>446.8</td>
<td>446.8</td>
<td>446.8</td>
<td>458.7</td>
<td>0.00%</td>
<td>0.00% 0.00% 0.00% 2.66% 0.00%</td>
</tr>
<tr>
<td>R105a</td>
<td>565.1</td>
<td>565.1</td>
<td>565.1</td>
<td>565.1</td>
<td>565.1</td>
<td>0.00%</td>
<td>0.00% 0.00% 0.00% 0.00% 0.00%</td>
</tr>
<tr>
<td>R105b</td>
<td>623.5</td>
<td>623.5</td>
<td>623.5</td>
<td>630.2</td>
<td>630.2</td>
<td>0.00%</td>
<td>0.72% 0.00% 1.07% 1.07% 0.00%</td>
</tr>
<tr>
<td>R105c</td>
<td>591.1</td>
<td>591.1</td>
<td>592.1</td>
<td>592.1</td>
<td>598.5</td>
<td>0.00%</td>
<td>0.17% 0.17% 1.25% 0.00% 0.00%</td>
</tr>
<tr>
<td>Total</td>
<td>8405</td>
<td>8432.1</td>
<td>8439.8</td>
<td>8497.7</td>
<td>0.32%</td>
<td>0.39% 0.38% 0.41% 1.10% 0.00%</td>
<td></td>
</tr>
</tbody>
</table>
The MABC is proposed to solve the extension of the vehicle routing problem with addition of backhauls and time windows, known as the VRPBTW. The proposed algorithm is based on ABC algorithm with additional memory for improving the effectiveness in finding the solution. The initial solutions are generated by random weighted time oriented nearest neighbor heuristic. The $\lambda$-interchange and 2-opt* local search are used as the neighborhood search mechanism. To evaluate the efficiency of the proposed algorithm, MABC is tested by obtaining 8 new best solutions which are better than other reference algorithms. It can be concluded that the proposed MABC has superior performance in solving VRPBTW by obtaining 8 new best solutions which are better than other reference algorithms. It can be concluded that the proposed MABC is out perform for VRPBTW. For the future research, the proposed algorithm may be applied to other variants of vehicle routing problem such as

The results for 50 customer nodes as in Table 2 show that MABC can determine the optimum solutions for 2 problems and obtains 5 new best solutions out of 13 instances considered this paper. The total distance of 12,907.2 and %Gapopt of 0.68% of MABC algorithm confirms that MABC is still efficient to find the solutions for medium-size problems.

Table 3 displays the results for 100 customer nodes. Although, the MABC can obtain only 2 new best solutions (R101b and R102c) for 6 instances but the total distance of MABC algorithm are close to the optimal values. %Gapopt of MABC algorithm are not significant different from GA. Moreover, the overall %CV for MABC are very lower which indicate that the MABC is capable to yield the solution with low variation.

### Table 2: Computational result for 50 customer nodes

<table>
<thead>
<tr>
<th>Problem</th>
<th>Optimal Solutions</th>
<th>MABC</th>
<th>ABCII</th>
<th>HMA</th>
<th>GA</th>
<th>DEA</th>
<th>%Gapopt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Dist</td>
<td>NV</td>
<td>%CV</td>
<td>Dist</td>
<td>Dist</td>
<td>MABC</td>
</tr>
<tr>
<td>R101a</td>
<td>1122.3</td>
<td>1133.3</td>
<td>15</td>
<td>0.03%</td>
<td>1134</td>
<td>1135.8</td>
<td>1138.1</td>
</tr>
<tr>
<td>R101b</td>
<td>1191.5</td>
<td>1191.6</td>
<td>16</td>
<td>0.32%</td>
<td>1191.6</td>
<td>1191.6</td>
<td>1192.7</td>
</tr>
<tr>
<td>R101c</td>
<td>1168.6</td>
<td>1183.9</td>
<td>16</td>
<td>0.00%</td>
<td>1183.9</td>
<td>1183.9</td>
<td>1183.9</td>
</tr>
<tr>
<td>R102a</td>
<td>974.7</td>
<td>976.5</td>
<td>12</td>
<td>0.00%</td>
<td>976.5</td>
<td>976.8</td>
<td>976.8</td>
</tr>
<tr>
<td>R102b</td>
<td>1024.8</td>
<td>1054.6</td>
<td>14</td>
<td>0.00%</td>
<td>1054.6</td>
<td>1046</td>
<td>1029.2</td>
</tr>
<tr>
<td>R102c</td>
<td>1057.2</td>
<td>1059.7</td>
<td>14</td>
<td>0.00%</td>
<td>1059.7</td>
<td>1061.6</td>
<td>1059.7</td>
</tr>
<tr>
<td>R103a</td>
<td>811.4</td>
<td>813.8</td>
<td>9</td>
<td>0.52%</td>
<td>821.6</td>
<td>815.5</td>
<td>813.3</td>
</tr>
<tr>
<td>R103b</td>
<td>882.8</td>
<td>886.3</td>
<td>11</td>
<td>0.33%</td>
<td>887.1</td>
<td>889.3</td>
<td>892.7</td>
</tr>
<tr>
<td>R103c</td>
<td>882.1</td>
<td>884.3</td>
<td>11</td>
<td>0.07%</td>
<td>885.1</td>
<td>887.7</td>
<td>885.5</td>
</tr>
<tr>
<td>R104c</td>
<td>733.6</td>
<td>733.6</td>
<td>8</td>
<td>0.27%</td>
<td>739.3</td>
<td>738.2</td>
<td>741.4</td>
</tr>
<tr>
<td>R105a</td>
<td>970.6</td>
<td>973.7</td>
<td>11</td>
<td>0.33%</td>
<td>985.2</td>
<td>978.5</td>
<td>1002.5</td>
</tr>
<tr>
<td>R105b</td>
<td>1007.5</td>
<td>1022.5</td>
<td>14</td>
<td>0.27%</td>
<td>1024.7</td>
<td>1026.7</td>
<td>1047.8</td>
</tr>
<tr>
<td>R105c</td>
<td>993.4</td>
<td>993.4</td>
<td>11</td>
<td>0.00%</td>
<td>993.4</td>
<td>996.2</td>
<td>1018</td>
</tr>
<tr>
<td>Total</td>
<td>12820.5</td>
<td>12907.2</td>
<td>0</td>
<td>0.00%</td>
<td>12936.7</td>
<td>12927.8</td>
<td>12981.6</td>
</tr>
</tbody>
</table>

### Table 3: Computational result for 100 customer nodes

<table>
<thead>
<tr>
<th>Problem</th>
<th>Optimal Solutions</th>
<th>MABC</th>
<th>ABCII</th>
<th>HMA</th>
<th>GA</th>
<th>DEA</th>
<th>%Gapopt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Dist</td>
<td>NV</td>
<td>%CV</td>
<td>Dist</td>
<td>Dist</td>
<td>MABC</td>
</tr>
<tr>
<td>R101a</td>
<td>1767.9</td>
<td>1813.3</td>
<td>24</td>
<td>0.09%</td>
<td>1818.6</td>
<td>1811.6</td>
<td>1815</td>
</tr>
<tr>
<td>R101b</td>
<td>1877.6</td>
<td>1887.7</td>
<td>25</td>
<td>0.39%</td>
<td>1904.5</td>
<td>1891.1</td>
<td>1896.6</td>
</tr>
<tr>
<td>R101c</td>
<td>1895.1</td>
<td>1909.4</td>
<td>25</td>
<td>0.54%</td>
<td>1928.2</td>
<td>1911.2</td>
<td>1905.9</td>
</tr>
<tr>
<td>R102a</td>
<td>1600.5</td>
<td>1625.5</td>
<td>21</td>
<td>0.53%</td>
<td>1640.7</td>
<td>1623.7</td>
<td>1622.9</td>
</tr>
<tr>
<td>R102b</td>
<td>1639.2</td>
<td>1700.7</td>
<td>22</td>
<td>0.30%</td>
<td>1717.3</td>
<td>1724</td>
<td>1688.1</td>
</tr>
<tr>
<td>R102c</td>
<td>1721.3</td>
<td>1734.7</td>
<td>21</td>
<td>0.29%</td>
<td>1752.2</td>
<td>1759.8</td>
<td>1735.7</td>
</tr>
<tr>
<td>Total</td>
<td>10501.6</td>
<td>10671.3</td>
<td>0</td>
<td>0.00%</td>
<td>10761.5</td>
<td>10721.4</td>
<td>10664.2</td>
</tr>
</tbody>
</table>
vehicle routing problem with pickup and delivery and open vehicle routing problem, etc. Moreover, the hybrid algorithm between two metaheuristics can also be developed to determine better solutions for the large-scale problem of VRPBTW.

Acknowledgements

This research was funded by Faculty of Engineering, King Mongkut's University of Technology North Bangkok, Thailand. This support is gratefully acknowledged.

References


