A Bio-Inspired Capacitated Vehicle-Routing Problem Scheme Using Artificial Bee Colony with Crossover Optimizations

Chatchai Punriboon[1], Chakchai So-In[1]*, Phet Aimtongkham[1], and Kanokmon Rujirakul[2]
[1]Applied Network Technology (ANT) Laboratory, Department of Computer Science, Faculty of Science, Khon Kaen University, Khon Kaen, 40002, Thailand
[2]Business Computer, Faculty of Management Science, Nakhon Ratchasima Rajabhat University, 30000, Thailand

Abstract

The capacitated vehicle-routing problem (CVRP) is considered one of the descendants of the traditional vehicle-routing problem (VRP) based on the consideration of capital gains in logistics and supply chains. Similar to VRP, CVRP is an NP-Hard problem; finding the optimal solution is difficult, especially with large amounts of data. In other words, the problem cannot be solved using a traditional approach because of the high cost of such an approach, i.e., high computational time. Thus, this research considers the possibility of integrating a variety of crossover methodologies into an artificial bee colony (ABC) as a heuristic approach to identifying a candidate for a CVRP solver. The research also optimizes ABC to obtain a better solution, rapidly, given time constraint, considering the effectiveness of randomness and precision enhancements related to both the crossover route and path diversity. The practicality of the proposal was evaluated by pitting fourteen well-known datasets against a traditional method, including other state-of-the-art CVRP heuristic solutions, and the performance improvement was confirmed in terms of both accuracy (i.e., finding the best solution) and the computational time as tradeoff.

Keywords: Artificial Bee Colony, Bio-inspired, Capacitated Vehicle Routing Problem, Crossover.

1 Introduction

Vehicle-routing problems (VRP)[5] are critical in the area of logistics and supply chains, especially regarding how to manage the schedule of vehicles - vans and trucks - traveling to different depots or customers that might be located in different regions. The optimal management scheme results in an investment-cost (fixed-cost) reduction for each transport. A better scheme will also decrease the travel time or even determine a shorter path for the optimization of fuel consumption. Through these reductions, the variable costs of each transport will also be reduced. However, for some varieties of these problems, this optimization leads to high computational time, and, traditionally, such problems can be considered NP-Hard problems[24], [34].

One of this problem's unique characteristics is that it is difficult to find the solution via exact methods, especially when the problem size is large and the conditions and constraints are complex. Thus, there have been many attempts by researchers to search for different methodologies to mitigate this phenomenon. However, some approaches may be suitable only for a particular problem, and those approaches' accuracy may decrease or their overall time complexity may increase when they are applied to different contexts[24]. Presently, there are many descendants of the traditional VRP[10], including VRP with Capacity Restriction (CVRP), VRP with Time Windows (VRPTW), VRP with Backhaul (VRPB), and VRP with Pickup and Delivery (VRPPD), each of which has its own features and characteristics. In

*Corresponding author: Applied Network Technology (ANT) Laboratory, Department of Computer Science, Faculty of Science, Khon Kaen University, Khon Kaen, 40002, Thailand, Email: chakso@kku.ac.th, Tel: +66-430-09700 this research, however, our focus is on CVRP, the Capacitated Vehicle-Routing Problem[33], in which there are specific conditions based on the vehicle capacity and distance constraint; thus, the research objective with respect to CVRP is to provide a heuristic-based solution for practical deployment in the logistics and supply chain industry and to fulfill the requirements imposed by decision makers[15].

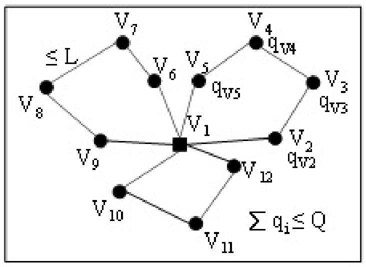

Figure 1: CVRP: Example

In general, CVRP features an undirected graph. In other words, there exists a graph, route, or path $G = (V,E)$. Suppose nodes $(V) = \{V_1,V_2,V_3,\ldots,V_n\}$ such that v_1 is the depot and the other nodes $(V_2,...,V_n)$ are lists of customers; edges $(E) = \{(V_i,V_j)|V_i,V_j \in V, i \neq j\}$ are sets of intermediate paths. M is the number of vehicles used for transportation. Note that each vehicle has a capacity Q. There are also other constraints as follows (as examples shown in Fig. 1):

- Each possible route (R) consists of nodes (V) and edges (E), and the start and end points of travel will be the same: either V_1 or the depot.
- The vehicle will visit each customer, V_i, only once and travel only one route.
- The total capacity of each vehicle, Q, will be limited, and the capacity of each customer i is given as q_i. Equation (1) shows this constraint numerically.
- In each route R_i, there is a constraint on the total distance L.

$$\sum q_i \le Q \tag{1}$$

For decades, the CVRP problem has been intensively researched. Tabu Search, proposed by É. Taillard et al. in 1993[32], is a classic approach to the problem. A parallel iterative search method was applied using Tabu, resulting in higher accuracy; however, because of its high level of time complexity, this method is not feasible in practice. Thus, several generations of Tabu Search have been investigated. For example, in 1998, C. Rego[27] applied a subpath ejection method to mitigate this problem.

Many techniques have also applied metaheuristics to alleviate the complexity, yielding best possible solutions that are close to the best solution given computational timing constraint. For example, in 2007, U. Derigs and R. Kaiser[9] proposed a hybrid model of an attribute-based hill climber and Tabu Search to determine a better answer given a time constraint. Similarly, in 2009, S. W. Lin et al.[18] used simulated annealing instead of the hill climber. Note that although there are many optimizations based on Tabu Search, its main limitation is infeasibility caused by high computational time.

Recently, there have also been novel techniques based on soft computing (SC)[34] (a subset of machine learning), especially to solve problems - in particular, NP-Complete problems - that have no certainty as an approximation solution of computation. There are several approaches based on SC, such as Neural Network (NN), Support Vector Machine (SVM), Fuzzy Logic (FL), Evolutionary Computation (EC), e.g., Genetic Algorithm (GA), Bayesian Network, and Chaos Theory[19].

There are many real-world applications, such as engineering, robotics and sensor networks[30], which apply these techniques, primarily due to their key advantage of computational time alleviation. The area of logistics and supply chains is another application that utilizes types of SC[15] and especially EC[2][4][20][22][25][26], especially bio-inspired algorithms including Ant Colony Optimization (ACO)[6][28][35] and Artificial Bee Colony (ABC)[3][29][31].

Recently, ABC has been a promising approach to a CVRP solution[3]: it has the key advantages of simplicity, high accuracy but with computational time as tradeoff. However, ABC has some limitations, in particular, the time that is required to determine the best solution when the data began to converge[31].

Therefore, in this research, our contributions are three-fold: 1) the enhancement of ABC with randomness to better find a solution rapidly based on timing constraint, 2) the integration of various crossover operations based on GAs into ABC to increase the opportunity to find the best solution (population diversity), and 3) the consideration of other criteria for the crossover route and path diversity. The combination of these considerations is Random Artificial Bee Colony with Optimized Crossovers (R-ABC-OC) for CVRP.

This article is organized as follows: Section 2 briefly explores some recent related work, especially on CVRP and SC. Then, the concept of our optimization, i.e., R-ABC-OC, will be discussed in Section 3. Next, Section 4 presents the comparative performance of our proposals versus that of a traditional CVRP solution with ABC, including intensive evaluation against other well-known CVRP solutions based on fourteen standard datasets. Finally, Section 5 contains both our conclusions and possible future research.

2 Related Works

Several research studies have proposed techniques to achieve possible solutions to VRP[24]. In this section, however, our focus is only on CVRP. Even though various techniques have been investigated[33], this research only considers a specific approach in which SC-based optimization [19], particularly a class of Evolutionary Computation (EC), has been integrated as a heuristic method for the purpose of practicality.

As a pioneer, É. Taillard[32] proposed a parallel technique for iterative search that would result in high accuracy regardless of computational time constraint. Taillard and his colleague, Y. Rochat and É. Taillard[29], also proposed a probabilistic diversification and intensification of local search. In terms of precision, that technique's performance is very high; however, there is no report on the timing tradeoff. Thus, this technique has been used worldwide as a benchmark for other (meta)heuristic approaches.

With respect to ECs, B. M. Baker and M. A. Ayechew[2] proposed the GA to achieve higher accuracy (a better solution) via crossover operators used as a heuristic approach for CVRP solvers; however, its main limitation remains its high computational time. Similarly, J. Berger and M. Barkoui[4] investigated how to apply GAs in parallel to improve their computational times required to find the best solution (close to the actual solution). Here, however, these GA approaches can only utilize diversity based on some of the crossover operators.

C. Prins[25] introduced a hybrid technique of Tabu search and GA (also with crossover optimizations) for the purpose of accuracy improvement over the computational time tradeoff. Note that V. F. Yu et al.[36] proposed the application of Symbiotic organisms search (SOS) for CVRP solver by imitating the ecology such that each route represents an organism which will develop by nature using parasitism as over-cross (i.e., swap) to create new generation. SOS gains higher precision than GA including DE (Differential Evolution); however, its precision is lower in case large-scale networks.

S. Kır et al.[16] and D. Zhang et al.[37] also applied Tabu search for CVRP. The first approach used Tabu as an initial route generation instead of random; then, they applied adaptive large neighborhood (ALN) search to polish the route given various operators, e.g., swapping and insert. The latter approach hybrids Tabu search and ABC; similar to [16], Tabu as the initial route generation but then ABC is used instead of ALN. Although these Tabu enhancements can improve the precision but with complexity trade-off.

In addition, D. Mester and O. Bräysy[20] proposed a novel methodology of active-guided evolution as a metaheuristic approach to solve CVRP in a large-scale dataset. C. Prins[26] (the same author as in [25]) again proposed a hybrid model. This time, however, the proposal involved hybridized greedy randomized adaptive search and evolutionary local search.

In addition, Y. Nagata and O. Bräysy[22] introduced a technique that involved applying a memetic algorithm for CVRP with a crossover optimization. Although this method can achieve very high precision, it consumes a great deal of computational time like 14 minutes. However, this approach has been widely used as a standard benchmark for other derivatives.

There are also several proposals focusing on metaheuristics and swarm intelligence (one class of EC). For example, several approaches focused on Ant Colony Optimization (ACO)[6][28][35]. B. Bullnheimer et al.[6] and B. Yu et al.[35] applied ACO for CVRP; but, that approach's many operations led to a situation of high computational time. Similarly, M. Reimann et al.[28] also proposed an idea of ACO with divide-and-conquer concepts to further reduce time for better solution towards the actual solution. Recently, R. Goel et al.[12] also adopt ACO such that the concentration of the pheromone is used instead of random route selection and Firefly algorithm is then used to develop a proper route over. There is no report on the computational complexity due to the additional search.

Recently, several researchers have conducted experiments based on Artificial Bee Colony (ABC) because of the key advantages of ABC's simplicity and accuracy gain. For example, P. Ji and Y. Wu[13] revised the traditional ABC with the assumptions that any new place found by the onlookers around a certain food source would become a new food source and that after a food source is abandoned, a concerned employed bee will find a new food source in the vicinity. These assumptions can be used to enhance performance by three percent.

In addition, W. Y. Szeto et al.[31] greatly improved the ABC used for CVRP by considering various neighborhood operations to achieve higher accuracy; this approach can be considered a baseline ABC for CVRP, although its precision can be further improved.

3 Random Artificial Bee Colony Using Optimized Crossovers

In this section, a detailed explanation of our proposal is discussed. There are four main stages to achieve our goals: (1) based on traditional ABC, a representation of ABC in CRVP, called ABC-CVRP; (2) an additional random condition with various neighborhood operations, called R-ABC-CVRP; (3) an optimization of crossover operators derived from GAs for CVRP, called OC-CVRP; and (4) a consideration of crossover routes and path diversity, called R-ABC-OC-CVRP.

3.1 ABC for CVRP

Artificial Bee Colony (ABC) is one of the heuristic algorithms[34], [21] used to determine the best solution that is close to the actual optimal solution (which may be impractical, especially at a high cost, i.e., computational infeasibility). Traditionally, ABC behavior attempts to imitate how bees collect food. There are three types of bees working in different functions as follows:

- **Employed Bee:** This type of bee only functions to randomly find food supplies and then inform onlooker bees about the food resource.

- **Onlooker Bee:** Once the resource is known, this type of bee will select some of the best resources and then collect the actual food. In addition, this bee will find nearby resources, if they exist, and then collect them all. It should be noted that once there is no more supply, the onlooker bee will become a scout bee.

- **Scout Bee:** The purpose of this type of bee is food exploration in a random manner; again, once the resource has been found, the employed-bee cycle will restart.

With this cycle of behavior, ABC can be applied to resolve the problem of CVRP as follows (See also Algorithm 1 below):

```
Data: FoodSize, Threshold, $\alpha$, $\beta$
Result: best–path
Initial Food Generation;
Fitness Function Evaluation;
while Threshold do
    Employed Bees;
    Onlooker Bees;
    Scout Bees;
    Fitness Function Evaluation;
end
```

Algorithm 1: ABC-CVRP

1. **Initial Food Generation:** This step is used to randomly create the paths or routes, i.e., initial population, as the initial data for further procedures. Fig. 2 shows an example of this initial process in that there are three routes here (from the first node and then going around until the first node is visited again: V_1 back to V_1).

2. **Fitness Function Evaluation:** To select a proper set of the selected routes, this state is used to measure performance as the selection criteria, as shown in equation (2). Here, C is the travelling cost or the summation of the distances in meter (D), the penalty of overweight (q) in kilogram, and the penalty of overtime (i.e., overdue in minute) (t). α (meter/kilogram) and β (meter/minute) are the coefficient parameters as adjustable weights over overweight and overtime ranging between 0.1 and 1. Note that after the overall cost is derived over population (X), the selection criteria will be applied to make ordered paths based on the lowest cost.

$$C = D + \alpha q + \beta t \tag{2}$$

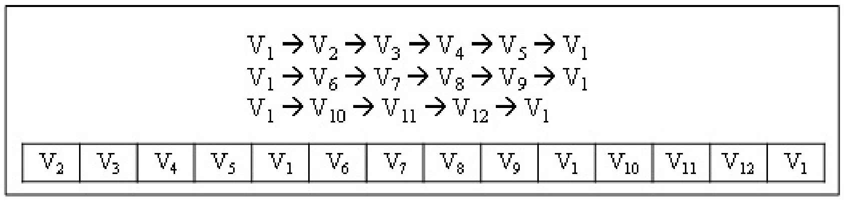
$V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_4 \rightarrow V_5 \rightarrow V_1$
$V_1 \rightarrow V_6 \rightarrow V_7 \rightarrow V_8 \rightarrow V_9 \rightarrow V_1$
$V_1 \rightarrow V_{10} \rightarrow V_{11} \rightarrow V_{12} \rightarrow V_1$
V_2 V_3 V_4 V_5 V_1 V_6 V_7 V_8 V_9 V_1 V_{10} V_{11} V_{12} V_1

Figure 2: ABC for CVRP: Initial Population

3. **Employed Bee:** This state is used to randomly select a set of data (routes) Y from the selected population X to prepare for the next stage, i.e., cost evaluation.

4. **Onlooker Bee:** Given Y, this state will select the data (routes) with the best cost from Y to create new data based on the neighbourhood operator, traditionally using a random-swap operation between nodes.

5. **Scout Bee:** This state is used to randomly create a new population, i.e., new routes.

It should be noted that the 3^{rd} through 5^{th} steps will be iteratively repeated based on a specific threshold, i.e., the iterative threshold ($Threshold$), as a limitation of ABC processes. Again, there is a computational time tradeoff that yields to the accuracy gain.

3.2 Random ABC for CVRP

One of the main limitations of ABC-CVRP, as discussed above, is that an Onlooker Bee has no variety on the neighborhood operator (only a random swap), most likely leading to precision deficiency. Similar to the diverse operators proposed by W. Y. Szeto et al.[31], e.g., Random Swap of Subsequence (RSS), Random Insertion (RI), Random Insertion of Subsequence (RIS), Reversing a Subsequence (RAS), Random Swap of Reversed Subsequence (RSRS), and Random Insertion of Reversed Subsequence (RIRS), all these operators will be used as our baseline step, one time only, to reduce the computational time to find a better solution close to the actual solution.

In this research, however, not only these six operators but also four extra operations and a traditional random swap are simultaneously applied to further improve our accuracy for the CVRP problem. In addition to these operators, we integrated the randomness before performing these operators for the purpose of computational time reduction, perhaps causing these extra operations, as stated in Algorithm 2.

```
Data: FoodSize, Threshold, $\alpha$, $\beta$, $\gamma$
Result: best–path
Initial Food Generation;
Fitness Function Evaluation;
while Threshold do
    Employed Bees;
    if random $< \gamma$ then
        Onlooker Bees;
        Scout Bees;
    end
    Fitness Function Evaluation;
end
```

Algorithm 2: Random ABC-CVRP (R-ABC-CVRP)

- **Random Condition:** As shown in Line 7, this step is used to randomly select the operation with probability (γ) to perform the functionality of the Onlooker Bees. Here, the random number lies in the range between 0 and 1, with the predefined probability ($threshold$) also between 0 and 1. Note that if the threshold is high, the opportunity to perform the next stage is also high.

- **Onlooker Bee:** This state is used to select the best cost route from Employed Bee (Y) to create new populations using the following ten neighborhood operators, as set forth below:

  1. **Random Swap (RS):** This is a traditional neighborhood operator used to swap the data of two positions randomly. As shown in Fig. 3(i), nodes V_i and V_j are swapped such that $i < j$; in this figure, i is 1 and j is 7, which is then 7 and 1 after swapping.
  2. **Random Swap of a Subsequence (RSS):** This step is used to swap a set of nodes in sequence, e.g., given two sets of nodes, i.e., $V_i, \ldots, V_j$, and $V_k, \ldots, V_l$ such that $i < j < k < l$. Based on Fig. 3(ii), suppose the first set is 1 to 4, and 7 to 8 for the second set; once swapped, the final route will be 7, 8, 5, 6, 1, 2, 3, 4, and 9.

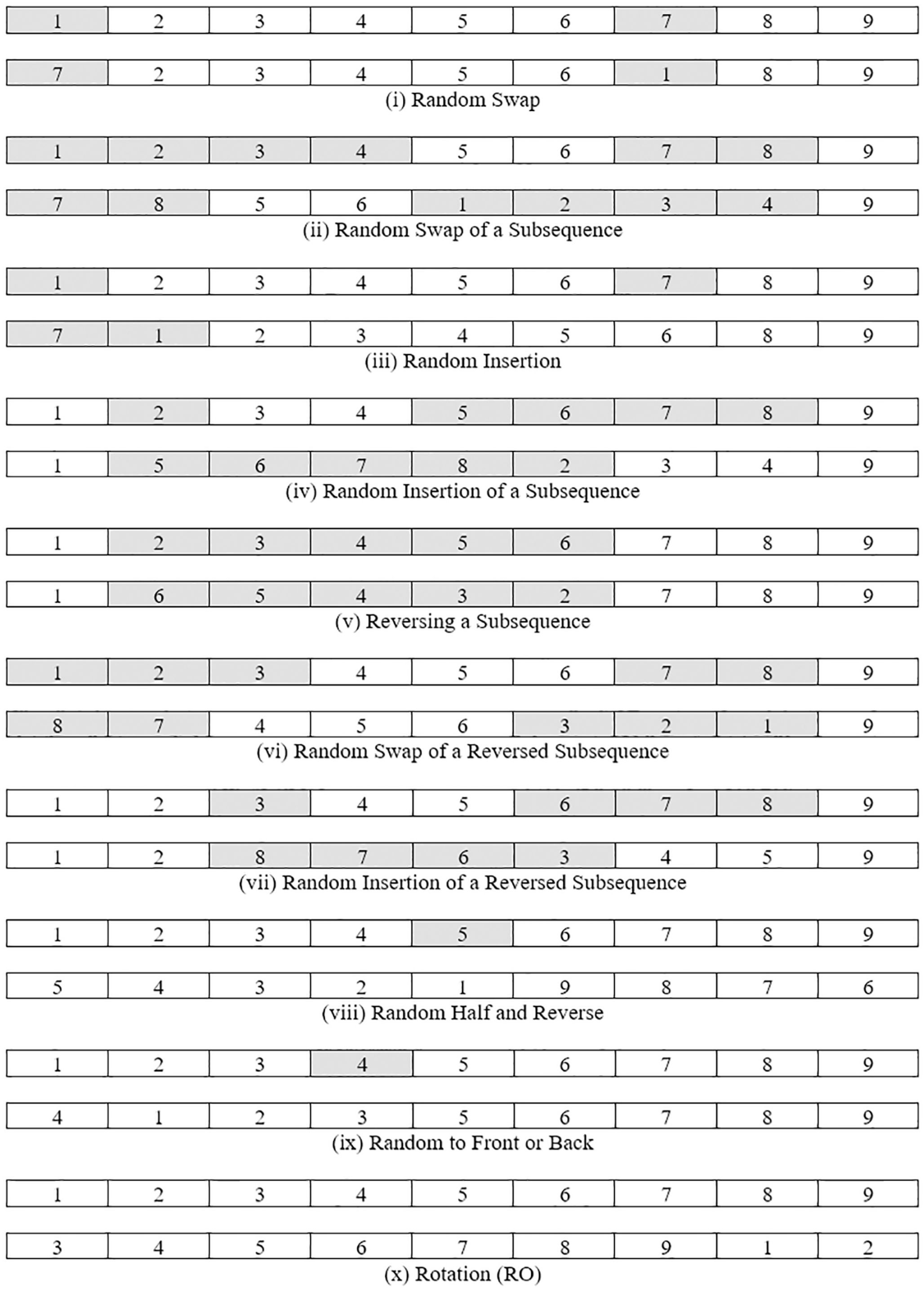

Figure 3: Neighborhood Operations

3. **Random Insertion (RI):** Similar to RS, this step randomly selects the two nodes V_j and V_j such that $i < j$. Next, it inserts V_j left before V_i. Based on Fig. 3(iii), V_i and V_j are 1 and 7,

and the final route will then move to node 7 before node 1.

4. **Random Insertion of a Subsequence (RIS):** Similar to RIS, instead of the insertion of only one node, this operator involves the insertion of a set of nodes. For example, given nodes V_i and $V_j, \ldots, V_k$ such that $i < j < k$, the set of nodes V_j to V_k will be moved in front of V_i. Fig. 3(iv) shows that if V_i is 2 and V_j to V_k are 5 to 8, the final sequence will be 1, 5, 6, 7, 8, 2, 3, 4, and 9.
5. **Reversing a Subsequence (RS):** This step applies a reverse operation in a set of nodes, i.e., given V_i to V_j, the final sequence of this set will be V_j to V_i. Fig. 3(v) shows that V_i to V_j is 2 to 6 and once reversed, the order will be from 6 to 2.
6. **Random Swap of a Reversed Subsequence (RSRS):** Given the reverse function, this step is used to swap a set of nodes in sequence and then apply the reverse operator. For example, given V_i to V_j and V_k to V_l such that $i < j < k < l$, after applying the operation, the final sequence will be V_l to V_k and then V_j to V_i. Fig. 3(vi) shows that, say, if V_i to V_j are 1 to 3 and V_k to V_l are 7 to 8, then after RSRS, the sequence will be 8 to 7 and 3 to 1.
7. **Random Insertion of a Reversed Subsequence (RIRS):** This step is used as a combination of insertion and reverse in that, given V_i and V_j to V_k such that $i < j < k$, after RIRS, a set of nodes V_j to V_k will be moved in front of V_i in reverse position (V_k to V_i). Fig. 3(vii) shows that V_i is 3 and V_j to V_k are 6 to 8; after RIRS, the final sequence will be 1, 2, 8, 7, 6, 3, 4, 5, and 9.
8. **Random Half and Reversing (RHR):** This step applies the reverse function to half of the nodes in the route, i.e., given n nodes, the nodes from V_1 to $V_{n/2}$ will be reversed. For example, from Fig. 3(viii), suppose the original sequence is 1 to 9; after RHR, the final sequence will be 5, 4, 3, 2, 1, 9, 8, 7, and 6, where 5 is the middle node.
9. **Random to Front or Back (RF or RB):** This step is used to move the middle node either to the front or to the back, i.e., moving $V_{n/2}$ to either V_1 or V_n. For example, Fig. 3(ix) states that node 4 will randomly move to the front of the sequence, i.e., in front of node 1.
10. **Rotation (RO):** This step is used to randomly rotate a set of nodes as follows: first, a random number is drawn (either 0 or 1) to indicate a left or right rotation, and then another random number will be generated as the number of rotations, i.e., K rotations. For example, Fig. 3(x) shows that the first random number is 1 = right and the second represents $K = 3$; thus, after RO, the final sequence will be 3, 4, 5, 6, 7, 8, 9, 0, 1, and 2.

- **Scout Bee:** Initially (again), this stage will be used to randomly create new routes (population) and then replace them with the worst routes (worst cost).

3.3 Optimized Crossover for CVRP

In general, the Genetic Algorithm (GA) is one of the evolutionary computations[34], [17] used as an approximation algorithm to imitate human genetics. The GA has three stages:

- **Population Selection:** This stage is used to properly select the population members of the current generation that will be parents of the next generation (includes a size constraint).
- **Genetic Operator:** The purpose of this stage is to generate better children, primarily using crossover and mutation techniques based on the various parents.
- **Replacement:** The children after the second stage will then replace those in the scaled population given the utility function, such as a ranking-based scheme.

Observe that the first and third stages are similar to the stages defined in ABC. In the second and third stages, Employed and Onlooker Bees can perform random food selections provided diverse neighbourhood operations. However, a Scout Bee only performs a random operation that may not be necessary when the data are converged. Thus, in this research, we proposed an optimized crossover to create a new, diverse population instead of Scout Bees excluding the other two stages of GA.

Note that a mutation operation can be considered as the random swap in ABC, so there is no requirement for this operation. For the crossover operations, the example is as follows: given two routes, i.e., V_{11} to V_{1i} and V_{1j} to V_{1k} and V_{1l} to V_{1n} (n nodes) and V_{21} to V_{2i} and V_{2j} to V_{2k} and V_{2l} to V_{2n} (n nodes), after applying crossover, two more sequences will be generated as follows:

V_{11} to V_{1i} and V_{2j} to V_{2l} and V_{1m} to V_{1n};
V_{21} to V_{2i} and V_{1j} to V_{1l} and V_{2m} to V_{2n}

For example, Fig. 4 shows that given two selected routes, 1 to 7 and 8 to 14, once the crossover operation is applied, two new sequences will be generated, i.e., 1, 2, 10, 11, 12, 6, 7, and 8, 9, 3, 4, 5, 13, 14.

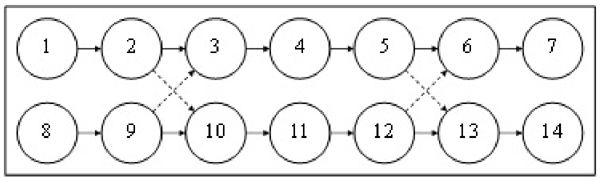

Figure 4: Crossover for CVRP: Example

In addition to Crossover for CVRP, based on our previous work[3], in which the crossover operation is performed on the shortest-path problem, we now propose the Crossover operator for CVRP, as stated in equation (3). We denote P_1 and P_2 as the parents (from the existing population), C_1 to C_4 as their children, and P_3 as the other parent selected from the best cost path.

$$(C_1, C_2, C_3, C_4) = (P_1 \cap P_2) \cup (P_3 \cap (P_1 \cup P_2)) \tag{3}$$

In addition, note that after conducting a crossover, it may be possible to create an uncompleted route (redundant nodes or incomplete visited nodes). Here, the best route will be selected to make a repair and perform two steps below (as shown in the examples in Fig. 5).

- A sequential search will be performed to delete the redundant node.
- The local optimum solution (the best existing route solution or P_3) will be selected as the route for repairing any missing nodes, and those nodes will be placed in position according to the best path.

For example, given 6, 3, 1, 8, 7, 4, 2, 5, 0, and 9 as the best path; C is the path required for reparation (6, 1, 4, 8, 0, and 2). Here, nodes 3, 5, 7, and 9 are missing, and so, to traverse over P_3 to repair C, the final repaired path will be as follows: 6, $\underline{\mathbf{3}}$, 1, $\underline{\mathbf{7}}$, 4, 8, $\underline{\mathbf{5}}$, 0, 2, and $\underline{\mathbf{9}}$.

3.4 Random ABC with Optimized Crossover for CVRP

Observe that on the one hand, Random ABC's main advantage is its computational time optimization, as one of the heuristic approaches with the nature of randomness. On the other hand, the crossover operation is effective in terms of its better accuracy achievement, i.e., its effectiveness in finding the solution, especially when the data tend to converge.

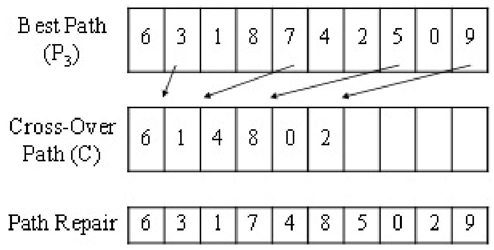

Figure 5: Path Repair: Example

Thus, this research combines these two as Random ABC with Optimized Crossover for CVRP (R-ABC-OC). Note that R-ABC-OC also considers two more cases with respect to making the crossover routes leading to low accuracy be the crossover path with either the same or different vehicle.

Fig. 6 shows an example of those two cases: Case 1 (1, 2, 3, 4, 5, 6, and 1) and Case 2 (1, 2, 3, 4, 5, 6, and 1; and 1, 7, 8, 9, 10, 11, and 1). There are two steps used to alleviate this problem: the intersection search of two lines and switch lines - the algorithm continues to search for the possible crossover path and then determines the possible solution.

- **Intersection Search of Two Lines:** This step is first used to find the crossover path, as previously stated (Fig. 6), and the methodology of our proposal is as follows:

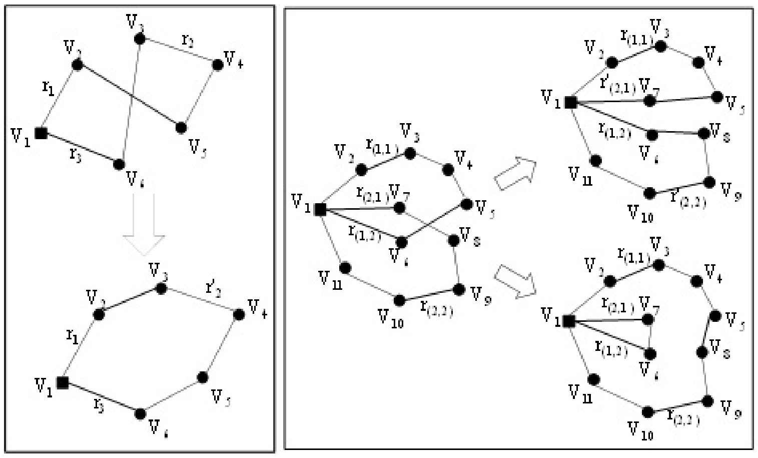

Figure 6: Crossover Route: Example

1. **Two-Path Selection:** This sub-step is used to select two possible intermediate-paths (4 nodes) to generate their gradients to form the straight line representation, as stated in equations (4) and (5).

$$y = mx + c \tag{4}$$

$$m = (y_2 - y_1) \div (x_2 - x_1) \tag{5}$$

where, m is the slope of each straight line, i.e., of coordinates (x_1, y_1) and (x_2, y_2) and c is the constant value.

2. **Crossover Point Derivation:** This step is used to derive the crossover point (x_c and y_c) based on the two paths, as stated in equation (6), using a simplified matrix manipulation.

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} -m_1 & 1 \\ -m_2 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \tag{6}$$

3. **Crossover Point Confirmation:** This step is used to confirm whether the crossover point of the actual position lies on the path in a specific range, i.e., (x_1, y_1) to (x_2, y_2) and (x_3, y_3) to (x_4, y_4). To confirm the answer, the checking condition of x_c is within a specific range between the minimum and maximum of (x_1 to x_4) and (y_1 to y_4) for y_c.

- **Switch Lines:** Once there exists a crossover path, this step is used to unfold the route as previously stated; in particular, there are two more subcases, which are set forth below.

1. **Crossover Path with The Same Vehicle:** This is the case when there is a vehicle that travels over the same point within a single route (R). In other words, if there are routes R_i and R_j, here, $i = j$. To generate a new solution, we are given $R = \{V_1, V_2, V_3, \ldots, V_n, V_1\}$ and the (x_c, y_c) coordinate that can derive the node both before and after the crossing point, i.e., V_p and V_q. This route can be unfolded from R (the sum of subroute r) to R', as stated in equations (7) through (10). It should be noted that r^R is the reverse of subroute r.

$$r_1 = \{V_1, V_2 \ldots V_p\} \tag{7}$$

$$r_2 = \left\{V_{p+1} \ldots V_q\right\} \tag{8}$$

$$r_3 = \left\{V_{q+1} \ldots V_n, V_1\right\} \tag{9}$$

$$R' = \left\{r_1, r_2, r_3^R\right\} \tag{10}$$

Note that in Fig. 6, the route (R) (1, 2, 3, 4, 5, 6, and 1) again can be rearranged to 1, 2, 3, 4, 5, 6, and 1. Here, r_1 is (1, 2), r_2 is (3, 4, 5) and r_3 is (1, 6).

2. **Crossover Path with Different Vehicles:** Each individual vehicle will not travel back to the same point; however, it is possible for another traveling vehicle to pass through that particular point. In other words, if there are routes R_i and R_j, here $i \neq j$. There are two possible solutions, which can be noted as stated in equations (11) through (14) below. Suppose there are two routes (R_1 and R_2) with different vehicles: $R_1 = \{r_{(1,1)}$ to $r_{(1,2)}\} = \{V_{(1,1)}, V_{(1,2)}, V_{(1,3)}, \ldots, V_{(1,n)}, V_{(1,1)}\}$ and $R_2 = \{r_{(2,1)}$ to $r_{(2,2)}\} = \{V_{(2,1)}, V_{(2,2)}, V_{(2,3)}, \ldots, V_{(2,n)}, V_{(2,1)}\}$. It should be noted that to simplify the notation, $V_{(1,1)}$ and $V_{(2,1)}$ are actually V_1.

$$r_{(1,1)} = \left\{V_{(1,1)}, V_{(1,2)} \ldots V_{(1,p)}\right\} \tag{11}$$

$$r_{(1,2)} = \left\{V_{(1,p+1)} \ldots V_{(1,n)}, V_{(1,1)}\right\} \tag{12}$$

$$r_{(2,1)} = \left\{V_{(2,1)}, V_{(2,2)} \ldots V_{(2,q)}\right\} \tag{13}$$

$$r_{(2,2)} = \left\{V_{(2,q+1)} \ldots V_{(2,n)}, V_{(2,1)}\right\} \tag{14}$$

Here, the first possible solution (R') is derived from equations (15) and (16), and the second solution can be computed from equations (17) and (18). Based on Fig. 5, the second case can be rearranged to (R'_1 = 1, 2, 3, 4, 5, and 7; and R'_2 = 1, 6, 8, 9, 10, 11, and 1) or (R'_1 = 1, 2, 3, 4, 5, 8, 9, 10, and 11; and R'_2 = 1, 7, 6, and 1).

$$R'_1 = \left\{r_{(1,1)},\ r_{(2,2)}\right\} \tag{15}$$

$$R'_2 = \left\{r_{(2,1)}, r_{(1,2)}\right\} \tag{16}$$

$$R'_1 = \left\{r_{(1,1)}, r^R_{(2,1)}\right\} \tag{17}$$

$$R'_2 = \left\{r^R_{(2,2)}, r_{(1,2)}\right\} \tag{18}$$

Observe that based on our intensive evaluation, there is a possibility that the algorithm cannot find the best path, because there is no such thing in the current population; this was indicated by the convergence of path finding. Thus, similar to the concept of backtracking, we randomly interchange the current population (once the solution tends to converge given the round threshold) with the best selection path, and so the probability to traverse the route in other directions (selecting the previous best solution for further operations) is high. We call this *reborn*, and its details are stated as follows:

- **Best Solution Population:** This stage is used to collect the best-known solution, if it exists.
- **Reborn:** Once the solution finder is converge, i.e., the population with other possible exchanging operations cannot be used to construct a better path, we randomly select some of the best paths in the previous states to replace the existing population with high probability of diversity and thus, a better solution can perhaps be derived.

This reborn step can only be used if there is no intersection between two lines to reduce the computation time for better solution; this will also be applied given the threshold of finding convergence, as detailed in Algorithm 3 as follows:

After the initial food generation and initial fitness evaluation (Lines 3-4) based on the population size (*FoodSize*), the employed bee will be utilized in Line 6. There is a random effect given not only threshold γ but also the optimization of onlooker bee states (with ten possible neighbourhood operations) (called R-ABC) (Lines 7 to 10). An extra crossover step based on GA (called OC) is then used instead of scout bees (Line 9).

A simple swap operation can be represented in the scout bee functionality; however, it is one of the onlooker-bee operations. Line 11 states the fitness evaluation to rank the population; Lines 12 to 16 indicate the convergence threshold (CT) of time to find convergence with the increase of the convergence counter (CC). Before checking the condition with threshold iterations, Lines 17 through 23 state the conditional checking for crossover routes, reborn for the purpose of population diversity (R-ABC-OC).

```
Data: FoodSize, Threshold, $\alpha$, $\beta$, $\gamma$, CT
Result: best–path
Initial Food Generation;
Fitness Function Evaluation;
while Threshold do
    Employed Bees;
    if random $< \gamma$ then
        Onlooker Bees;
        Scout Bees;
    end
    Fitness Function Evaluation;
    if there exists better solution then
        Store Current Best Solution;
    else
        Increase CC;
    end
    if $CC > CT$ then
        if Intersection Search of two lines then
            Switch Lines;
        else
            Reborn;
        end
    end
end
```

Algorithm 3: Random ABC with Optimized Crossover for CVRP

4 Performance Evaluation

This section discusses the performance of our approach based on ABC and optimized crossovers, including the crossover route and path-diversity considerations, i.e., R-ABC-OC for CVRP versus a traditional ABC[31] for CVRP. We also show algorithmic performance over different types of techniques for practicality using a set of standard benchmarks.

4.1 Empirical Configurations

The evaluation testbed is a standard configuration using the Windows 7 operating system (64-bits): Intel(R) (TM) quadcore 2.66 GHz CPU, 2048 MB DDR-SDRAM, and a 320 GB 7200 RPM disk. For comparative purposes that consider precision, the simulation model is based on MATLAB[1] on a standard baseline dataset. To enhance computational speed, standard C was also used to simulate for testing purposes[31] (14 sets from N. Christofides and S. Eilon[7] and N. Christofides et al.[8]).

For comparison purposes, the evaluation method defined a colony size according to the recommendation provided by D. Karaboga and B. Basturk[14], i.e., 50. The coefficient parameters, i.e., α and β, are in the range between 0.1 and 1; based on intensive investigation, 0.45 is the best and will be used throughout the experiments. In addition, in each evaluation there are 20 trials, each of which will be limited to 100,000 rounds of iteration based on the recommendation provided by W. Y. Szeto et al.[31]. Then, in addition to the precision metric that will be defined later, the computational time in minutes will be measured accordingly.

There are three main scenarios in which to evaluate our performance. First, we selected one of the fourteen sets, i.e., vpcn1, and then considered the measurement metrics of the minimum, maximum, and average objective values, including computational times for our optimizations at each stage, i.e., ABC-CVRP, R-ABC-CVRP, and R-ABC-OC-CVRP, against a traditional ABC for CVRP[31]. Our first stage, ABC-CVRP, was developed as an implement justification, and the result should be close to the ABC proposed by W. Y. Szeto et al.[31]. We also measured the time to convergence of those three techniques using the best case.

Second, similar to the first scenario, to elaborate on our intensive evaluation, all fourteen sets were used for evaluation; however, only R-ABC-OC was performed against the best-known approaches based on a standard benchmark provided by N. Christofides and S. Eilon[7] and N. Christofides et al.[8]. It should be noted that in the fourteen sets, n denotes the number of customers; Q is the vehicle capacity (e.g., kilogram); s is the service time (minute); L is the limitation of total distance (meter) and m is the number of vehicles.

The precision metric (meter) used in this evaluation is a percentage deviation (PD), which is specific gravity compared to the best-known solution derived from equation (19)[23]. Here, Sol denotes the best solution available (meter); BKS is the best known solution (meter). We compared three baseline algorithms provided by É. Taillard[32]; D. Mester and O. Bräysy[20] and Y. and É. Taillard[29]. We also showed the computational time in minutes.

$$PD \;=\; ((Sol - BKS) \div BKS) \times 100 \tag{19}$$

Finally, the third scenario is to numerically compare our proposal, R-ABC-OC, with other best-known approaches (eleven well-known approaches), such as É. Taillard (1993)[32], M. Gendreau et al. (1994)[11], Y. Rochat and É. Taillard (1995)[29], B. Bullnheimer et al. (1999)[6], B. M. Baker and M. A. Ayechew (2003)[2], B. M. Berger and M. A. Barkoui (2003)[4], Y. Nagata and O. Bräysy (2009)[22], S. W. Lin et al. (2009)[18], W. Y. Szeto et al. (2011)[31], R. Goel et al. (2018)[12] and V. F. Yu et al. (2017)[36]. Here, the matrices are an average of PD of fourteen cases and computational time (CPU time) (minutes), including the type of complier and computer specification. As previously stated, we developed R-ABC-OC in the C language to achieve high performance in terms of computational time.

4.2 Empirical Results and Discussions

Considering the first scenario, Table 1 shows the objective values, including the computational times of traditional ABC applied to CVRP against our proposals, i.e., R-ABC and R-ABC-OC. Observe that ABC's performance is similar to what was reported in[31], i.e., approximately 527 and 525 for the minimum and 532 and 536 for the average, although the maximum was not reported, with 0.02 minutes in computational time difference. Considering our proposal, it is obvious that R-ABC-OC's performance is outstanding, i.e., the average of objective values is just 527.85, compared to just 531.74 for traditional ABC[31].

It should be noted that our first optimization stage, i.e., R-ABC, still outperforms the traditional ABC by almost 2 in objective value reduction. In terms of computational time, it is understandable that models with more stages will introduce extra times. However, with precision improvement, a degree of computational time is not significantly affected, such as only 0.54 and 0.93 minutes versus 0.42 (traditional ABC[31]) minutes, respectively. In addition to both traditional metrics and the computational time on average, Fig. 7 shows the iteration convergence to a steady state for the three schemes, i.e., ABC, R-ABC, and R-ABC-OC.

Similar to the performance trend shown in Table 1, it is observed that R-ABC-OC is the best scheme; the fastest convergence, i.e., the steady state, will be around iteration 4500 with R-ABC-OC. For the

Table 1: Objective Values versus Computational Time for CVRP: ABC[31], ABC, R-ABC, and R-ABC-OC

	ABC-CVRP[31]	ABC-CVRP	R-ABC-CVRP	R-ABC-OC-CVRP
Minimum	526.97	524.92	524.61	524.61
Maximum	N/A	568.87	560.19	555.25
Average	531.74	536.16	529.58	527.85
CPU time (minutes)	0.42	0.40	0.54	0.93

Table 2: Comparative Results for Each Set in the Fourteen Cases Provided By [7], [8]

Instance	n	Q	s	L	m	BKS	Average	Sol	PD	Time
vrpnc1	50	160	0	∞	5	524.61a	527.85	524.61	0	0.93
vrpnc2	75	140	0	∞	10	835.26 a	840.52	835.26	0	1.47
vrpnc3	100	200	0	∞	8	826.14 a	833.49	826.14	0	4.46
vrpnc4	150	200	0	∞	12	1028.42 a	1052.52	1030.46	0.18	6.25
vrpnc5	199	200	0	∞	17	1291.29 b	1331.92	1299.64	0.65	13.33
vrpnc6	50	160	10	200	6	555.43 a	557.67	555.43	0	0.71
vrpnc7	75	140	10	160	11	909.68 a	915.53	909.68	0	1.24
vrpnc8	100	200	10	230	9	865.94 a	878.79	865.94	0	4.34
vrpnc9	150	200	10	200	14	1162.55 a	1193.88	1163.38	0.07	8.81
vrpnc10	199	200	10	200	18	1395.85 c	1428.73	1407.21	0.81	16.37
vrpnc11	120	200	0	∞	7	1042.11 a	1058.26	1043.11	0.1	3.51
vrpnc12	100	200	0	∞	10	819.56 a	825.86	819.56	0	2.07
vrpnc13	120	200	10	720	11	1541.14 a	1553.57	1542.25	0.07	2.50
vrpnc14	100	200	10	1040	11	866.37 a	869.31	866.37	0	1.92
Average									0.13	4.85

others, the iterations are 9000 and 14500 for R-ABC and ABC, respectively. It is worthwhile to note that based on the evaluation of traditional ABC comparatively given by W. Y. Szeto et al. (2011)[31], their time to convergence is beyond 50,000 rounds.

Table 2 shows the results from the second scenario with the fourteen datasets. Based on the three best well-known approaches, our proposal's performance is comparative; for example, in (a), our PDs are in the range of 0 to 0.18 and essentially 0 for 9 sets out of 14 sets. The PD of our proposal falls between 0.65 and 0.81 for the others, i.e., (b) and (c). On average, the PD for all fourteen cases is 0.13 with a 0.25 standard deviation. Moreover, in terms of computational time, the average is approximately 4.85 minutes with a 4.83 standard deviation. R-ABC-OC took less than approximately 6.25 minutes to achieve the best solution, and the computation time is higher for the cases that require more iterations, i.e., in vrpnc5, 9, and 10 (maximum of 16.37 minutes for vrpnc10).

Finally, to intensively illustrate the practicality of our scheme against the other best well-known approaches on average, Table 3 shows the results in terms of APD and CPU time for 12 approaches. It is obvious that the best existing approaches in terms of the precision or APD are those of Y. Rochat and É. Taillard[29] and Y. Nagata and O. Bräysy[22], i.e., 0 for APD; however, the computational time is neither provided nor extensively high (13.8 minutes for the second approach), probably leading to unfeasible solutions. That trend also applied to É. Taillard[32] (APD = 0.05).

Our proposal, R-ABC-OC, yields very high precision, although it may not be better than those approaches; however, our computation time is acceptable at only 4.85 minutes. Note that although it may not be possible to make a fair comparison based on computational time caused by the use of var-

Table 3: Comparative Results Based on the Best-Known Results

References	APD	CPU Time (minutes)	Techniques	Compiler	Computer Specification
Y. Rochat and É. Taillard (1995)[29]	0	N/A	Diversify; Intensify; Parallelize	C	Silicon Graphics 100 MHz
Y. Nagata and O. Bräysy (2009)[22]	0	13.8	EAX Memetic	C++	Opteron 2.4 GHz
É. Taillard (1993)[32]	0.05	N/A	Parallel Taboo Search	C++	Silicon Graphics 100 MHz
R. Goel et al. (2018)[12]	0.093	N/A	Ant colony + firefly Algorithm	Matlab	Core i3 2.3 Ghz
R-ABC-OC	0.13	4.85	ABC + GA Optimizations	C	Core2 2.66 GHz
V. F. Yu et al. (2017)[36]	0.24	1.29	SOS	C++	Core i7 3.4 GHz
S. W. Lin et al. (2009)[18]	0.35	8.21	Variable-length Neighbor List Record-to-Record Travel	C	Pentium IV 2.8 GHz
W. Y. Szeto et al. (2011)[31]	0.46	4.4	Artificial Bee Colony	C++	Pentium 1.73 GHz
J. Berger and M. Barkoui (2003)[4]	0.49	21.25	Hybrid Genetic Algorithm	C++	Pentium 400 MHz
B. M. Baker and M. A. Ayechew (2003)[2]	0.56	29.11	Genetic Algorithm	C	Pentium 266 MHz
M. Gendreau et al. (1994)[11]	0.86	46.8	Tabu Search	N/A	Silicon Graphics 36 MHz
B. Bullnheimer et al. (1999)[6]	1.51	18.44	Ant System	N/A	Pentium 100 MHz

ious platforms, our proposal outperforms the traditional ABC, including its derivatives, as stated in the first scenario. Moreover, considering the other seven approaches with computational complexity report, R-ABC-OC is outstanding in both precision (only 0.13 versus 0.35 to 1.51) and computational time (approximately 4 minutes versus 4 to 30 minutes). To specifically compare with the first top-four approaches, again, although our APD is a bit lower than those, according to the nature of heuristics, our practicality based on computational time is low.

5 Conclusion and Future Work

In this research, we conducted an investigation of the application of SCs - applying bio-inspired computation, i.e., crossover operations and ABC, to solve the problem of a derivative of the vehicular routing problem given the capacity constraint, i.e., CVRP. Because CVRP is an NP-Complete problem, this combination can be considered as a metaheuristic approach, i.e., the best solution given the practicality versus the computational time tradeoff. Our hybrid model of GA and ABC, called Random ABC with optimized crossover (R-ABC-OC), concerns the crossover route and path diversity and can be considered as a CVRP solver candidate.

Based on our intensive evaluation, the performance of R-ABC-OC is outstanding, and in particular,

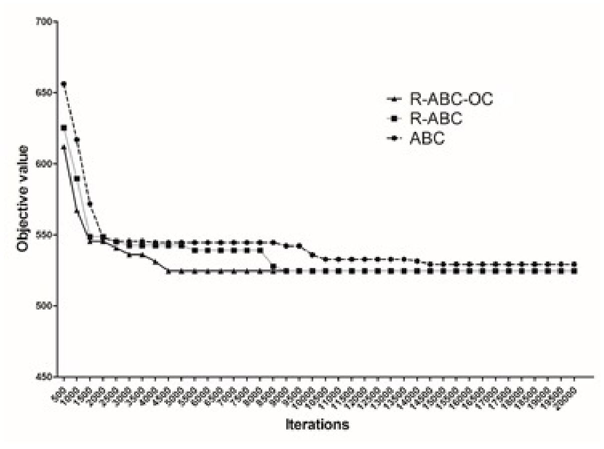

Figure 7: Iteration Convergence

it outperforms the traditional ABC with an insignificant computational time tradeoff. Considering the evaluation of the fourteen standard sets used as a benchmark, R-ABC-OC can also be considered a competitive solver in terms of both precision and computational time improvement. In addition, when compared with other schemes in various well-known models and configurations, R-ABC-OC continues to maintain high accuracy with computational time tradeoff.

Although this hybrid scheme of ABC with optimized crossovers achieves high performance, more investigations, assumptions, and constraints could be further explored - i.e., to achieve higher accuracy - and a variety of conditions, stages, and optimizations, including other combinations of hybrid models, can be well-investigated. To achieve a high degree of computational time improvement, programming experts with great coding skills may be required for code optimization. A larger dataset can also be considered for high time complexity problems, including the heterogeneity of the dataset. It is also worth noting that different aspects of VRP solvers, such as VRP with Time Windows and Pickup/Delivery, are being explored, and all of this is ongoing research.

Acknowledgments

The research were partly granted by TRFMAG 2014 and Khon Kaen University, Faculty of Science, Khon Kaen University.

References

[1] Matlab toolbox. `https://www.mathworks.com/`, [Online; Accessed on August 2, 2019].

[2] B. M. Baker and M. A. Ayechew. A genetic algorithm for the vehicle routing problem. *Computers and Operations Research*, 30(5):787–800, April 2003.

[3] A. Baykasolu, L. Oumlzbakr, and P. Tapk. Artificial bee colony algorithm and its application to generalized assignment problem. In F. Chan and M. Tiwari, editors, *Swarm Intelligence, Focus on Ant and Particle Swarm Optimization*, December 2007.

[4] J. Berger and M. Barkaoui. A new hybrid genetic algorithm for the capacitated vehicle routing problem. *Journal of the Operational Research Society*, 54(12):1254–1262, December 2003.

[5] O. Bräysy, W. Dullaert, and M. Gendreau. Evolutionary algorithms for the vehicle routing problem with time windows. *Journal of Heuristics*, 10(6):587–611, December 2004.

[6] B. Bullnheimer, R. F. Hartl, and C. Strauss. An improved ant system algorithm for thevehicle routing problem. *Annals of Operations Research*, 89(0):319–328, January 1999.

[7] N. Christofides and S. Eilon. An algorithm for the vehicle-dispatching problem. *Journal of the Operational Research Society*, 20(3):309–318, September 1969.

[8] N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. *Combinatorial Optimization*, pages 315–338, 1979.

[9] U. Derigs and R. Kaiser. Applying the attribute based hill climber heuristic to the vehicle routing problem. *European Journal of Operational Research*, 177(2):719–732, March 2007.

[10] B. Eksioglu, A. V. Vural, and A. Reisman. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4):1472–1483, November 2009.

[11] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10):1276–1290, October 1994.

[12] R. Goel and R. Maini. A hybrid of ant colony and firefly algorithms (hafa) for solving vehicle routing problems. *Journal of Computational Science*, 25:28–37, March 2018.

[13] P. Ji and Y. Wu. An improved artificial bee colony algorithm for the capacitated vehicle routing problem with time-dependent travel times. In *Proc. of the 10th international symposium on operations research and its applications (ISORA'11), Dunhuang, China*, pages 75–82, August 2011.

[14] D. Karaboga and B. Basturk. On the performance of artificial bee colony (abc) algorithm. *Applied Soft Computing Journal*, 8(1):687–697, January 2008.

[15] M. Ko, A. Tiwari, and J. Mehnen. A review of soft computing applications in supply chain management. *Applied Soft Computing Journal*, 10(3):661–674, June 2010.

[16] S. Kır, H. R. Yazgan, and E. Tüncel. A novel heuristic algorithm for capacitated vehicle routing problem. *Journal of Industrial Engineering International*, 13(3):323–330, September 2017.

[17] Z. Laboudi and S. Chikhi. Comparison of genetic algorithm and quantum genetic algorithm. *International Arab Journal of Information Technology*, 9(3), May 2012.

[18] S. W. Lin, Z. J. Lee, K. C. Ying, and C. Y. Lee. Applying hybrid meta-heuristics for capacitated vehicle routing problem. *Expert Systems with Applications*, 36(2):1505–1512, March 2009.

[19] R. Malhotra, N. Singh, and Y. Singh. Soft computing techniques for process control applications. *International Journal on Soft Computing*, 2(3):32–44, August 2011.

[20] D. Mester and O. Bräysy. Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers and Operations Research*, 34(10):2964–2975, October 2007.

[21] Z. Mustaffa and Y. Yusof. LSSVM parameters tuning with enhanced artificial bee colony. *International Arab Journal of Information Technology*, 11(3):236–242, May 2014.

[22] Y. Nagata and O. Bräysy. Edge assembly-based memetic algorithm for the capacitated vehicle routing problem. *Networks*, 54(4):205–215, August 2009.

[23] H. Nazif and L. S. Lee. Optimised crossover genetic algorithm for capacitated vehicle routing problem. *Applied Mathematical Modelling*, 36(5):2110–2117, May 2012.

[24] J. Potvin. Evolutionary algorithms for vehicle routing. `http://www.iro.umontreal.ca/~dift6751/ga_vrp_JOC.pdf`, [Online; Accessed on August 2, 2019], 2007.

[25] C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985–2002, October 2004.

[26] C. Prins. A grasp x evolutionary local search hybrid for the vehicle routing problem. In F. B. Pereira and J. Tavares, editors, *Bio-inspired Algorithms for the Vehicle Routing Problem*, volume 161 of *Studies in Computational Intelligence*, pages 35–53. Springer, Berlin, Heidelberg, 2009.

[27] C. Rego. A subpath ejection method for the vehicle routing problem. *Management Science*, 44(10):1447–1459, October 1998.

[28] M. Reimann, K. Doerner, and R. F. Hartl. D-ants: Savings based ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, 31(4):563–591, April 2004.

[29] Y. Rochat and É. D. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1(1):147–167, September 1995.

[30] K. M. Saridakis and A. J. Dentsoras. Soft computing in engineering design - a review. *Advanced Engineering Informatics*, 22(2):202–221, April 2008.

[31] W. Y. Szeto, Y. Wu, and S. C. Ho. An artificial bee colony algorithm for the capacitated vehicle routing problem. *European Journal of Operational Research*, 215(1):126–135, November 2011.

[32] É. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8):661–673, December 1993.

[33] P. Toth and D. Vigo. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, 2002.

[34] P. M. Vasant. *Handbook of research on novel soft computing intelligent algorithms: Theory and practical applications*. IGI Global, August 2013.

[35] B. Yu, Z. Z. Yang, and B. Yao. An improved ant colony optimization for vehicle routing problem. *European Journal of Operational Research*, 196(1):171–176, July 2009.

[36] V. F. Yu, A. A. N. P. Redi, C. L. Yang, E. Ruskartina, and B. Santosa. Symbiotic organisms search and two solution representations for solving the capacitated vehicle routing problem. *Applied Soft Computing Journal*, 52:657–672, March 2017.

[37] D. Zhang, S. Cai, F. Ye, Y. W. Si, and T. T. Nguyen. A hybrid algorithm for a vehicle routing problem with realistic constraints. *Information Sciences*, 394–395:167–182, July 2017.

Author Biography

Chatchai Punriboon is a Ph.D. student at the Department of Computer Science, Faculty of Science, Khon Kaen University, Thailand. He received his B.S. and M.S. degrees in Computer Science from, Khon Kaen University, in 2012 and 2016, respectively. His research interests include image processing, vehicle routing, wireless sensor networks and Internet of Things.

Chakchai So-In (IEEE/ACM SMs) is a Professor at the Department of Computer Science at Khon Kaen University and received his Ph.D. in Computer Engineering from Washington University in St. Louis, MO, USA in 2010. He was an intern at CNAP-NTU (SG), Cisco Systems, WiMAX Forums, and Bell Labs (USA). His research interests include mobile computing, wireless/sensor networks, signal processing, and computer networking and security. He has served as an editor at PLOS ONE, SpringerPlus, PeerJ, and ECTI-CIT and as a committee member for many conferences/journals such as Globecom, ICC, VTC, WCNC, ICNP, ICNC, PIMRC, IEEE Transactions, IEEE Letter/Magazines, and Computer Networks/ Communications. He has authored over 100 publications and 10 books, including some in IEEE JSAC, IEEE Magazines, and Computer Network/ Network Security Labs.

Phet Aimtongkham is a Ph.D. student at the Department of Computer Science, Faculty of Science, Khon Kaen University, Thailand. He received his B.S. and M.S. degrees in Computer Science and Information Technology from, Khon Kaen University, in 2013 and 2016, respectively. He is founder of Advanced Internetworking Co. Ltd. His research interests include computer networking, multimedia networks, wireless sensor networks and Internet of Things.

Kanokmon Rujirakul received an MSc. from NIDA in 2008 and is currently a lecturer at Business Computer, Faculty of Management Science, Nakhon Ratchasima Rajabhat University . Her research areas include image processing, mobile computing, soft computing, and distributed systems.