Annals of Operations Research (2022) 308:549–570
https://doi.org/10.1007/s10479-020-03625-5

S.I. : ARTIFICIAL INTELLIGENCE IN OPERATIONS MANAGEMENT

Swarm intelligence-based hyper-heuristic for the vehicle routing problem with prioritized customers

Abbas Tarhini[1] · **Kassem Danach**[2] · **Antoine Harfouche**[3]

Published online: 7 May 2020

Abstract

The vehicle routing problem (VRP) is a combinatorial optimization management problem that seeks the optimal set of routes traversed by a vehicle to deliver products to customers. A recognized problem in this domain is to serve 'prioritized' customers in the shortest possible time where customers with known demands are supplied by one or several depots. This problem is known as the Vehicle Routing with Prioritized Customers (VRPC). The purpose of this work is to present and compare two artificial intelligence-based novel methods that minimize the traveling distance of vehicles when moving cargo to prioritized customers. Various studies have been conducted regarding this topic; nevertheless, up to now, few studies used the Cuckoo Search-based hyper-heuristic. This paper modifies a classical mathematical model that represents the VRPC, implements and tests an evolutionary Cuckoo Search-based hyper-heuristic, and then compares the results with those of our proposed modified version of the Clarke Wright (CW) algorithm. In this modified version, the CW algorithm serves all customers per their preassigned priorities while covering the needed working hours. The results indicate that the solution selected by the Cuckoo Search-based hyper-heuristic outperformed the modified Clarke Wright algorithm while taking into consideration the customers' priority and demands and the vehicle capacity.

Keywords Warm intelligence · Hyper-heuristic · Combinatorial problem · Vehicle routing problem · Clarke Wright algorithm · Cuckoo search algorithm

✉ Abbas Tarhini
abbas.tarhini@lau.edu.lb

Kassem Danach
kassem.danach@iul.edu.lb

Antoine Harfouche
Antoine.HARFOUCHE@edhec.edu

1 Information Technology and Operations Management Department, Lebanese American University, Beirut, Lebanon

2 Department of Management Information Systems, Islamic University of Lebanon, Khalde, Lebanon

3 Department of Management Information Systems, Université Paris Nanterre, Paris, France

1 Introduction

Transportation has a significant impact on today's societies; it has large impacts on economic growth and employment (Fink et al. 2019). Transportation employs millions of people globally and it is considered as a major component of organizations' costs (Baradaran et al. 2019; El Khoury et al. 2014; Comtois et al. 2013). Further, transportation depends heavily on oil resources and is considered a focal source of CO_2, CO, N_2O, and NH_3 emissions. As stated by a US EPA report, approximately 28% of the national greenhouse-gas emissions in 2017 were generated by transportation. It is therefore becoming a priority for transportation companies to optimize their transportation processes since small improvements can lead to huge impacts on the environment and on organizations' cost reductions. Furthermore, today, companies are both concerned with the costs and highly interested in providing the best customer service to optimize fulfillment, logistics, and production, which in turn lead to tight customer loyalty, and thus better organizational performance.

In transportation, the Vehicle Routing Problem (VRP) deals with the transportation of goods between depots and customers, where a set of routes must be defined for a number of vehicles to travel from their depot(s) to customers (Côté et al. 2020). The traveling cost between the depot and each customer and between each pair of customers is given. The VRP solution must find a route for each vehicle, starting and ending at the depot, such that a set of customers is served by exactly one vehicle, the overall cost of the routes is minimized and customer satisfaction (fulfilling their demands) is maximized while taking into account a set of given constraints. Typically, the solution to a VRP has to take into consideration several other restrictions, such as the capacity of the vehicles, the working hours of the salespersons, and the priority of the desired customers. Further, there are several variants to the VRP that take into account different factors such as the nature of the transported goods, the quality of the service required, and the characteristics of the customers and the vehicles. In Fig. 1 below, we show a typical input for a VRP problem and one of its possible outputs:

The literature presents different algorithms that have been used to solve the VRP such as the Tabu Search (Du and He 2012; Jin et al. 2012), the Artificial Bee Colony algorithm (Szeto et al. 2011; Gomez and Salhi 2014), the Bee Mating Optimization algorithm (Marinaki et al. 2010), Ant Colony Optimization (Akpinar 2016), the Genetic Algorithm (GA) (Nazif and Lee 2012), Particle Swarm Optimization (PSO) (Kim and Son 2012; Chen et al. 2006), the Water Flow Alike algorithm (Zainudin et al. 2015), the membrane algorithm (Niu et al. 2015), the Cooperative Parallel metaheuristic (Jin et al. 2014) and the Clarke

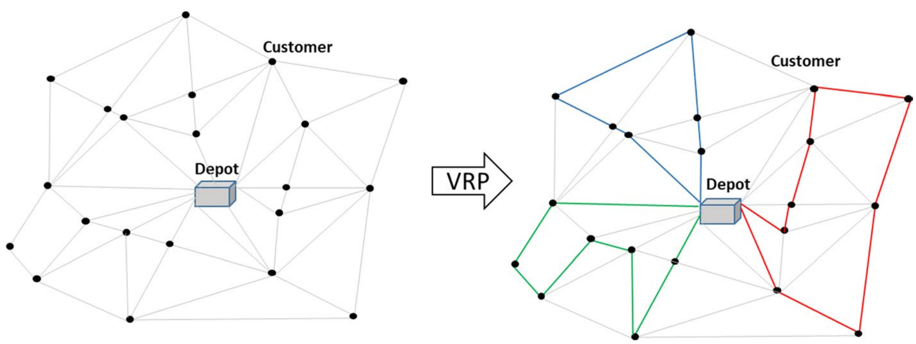

Fig. 1 An instance of a VRP (left) and its solution (right)

Wright algorithm (Clarke and Wright 1964; Shour et al. 2015). The Clarke Wright (CW) algorithm was developed in 1964 to solve the VRP. The CW is classified as a constructive method used to addresses a variant number of vehicles and works evenly for both directed and undirected problems. Further, a recent metaheuristic known as the cuckoo search (CS) was introduced by Yang and Deb in 2009 and has received much attention from researchers in various optimization areas. The CS has been applied to continuous optimization problems where it has shown better performance when compared to popular meta-heuristic algorithms such as the GA, Particle Swarm Optimization (PSO) and others (Ouaarab et al. 2014; Yang and Deb 2010; Yildiz 2013).

Recently, it became very popular among researchers to use search methods for selecting heuristics to solve computational search problems (Burke et al. 2010). This new optimization paradigm is called Hyper-heuristics and is described as using “heuristics to choose heuristics”. The main difference between hyper-heuristics and meta-heuristics is that hyper-heuristics directly search a space of heuristics rather than a space of problem solutions. Thus, when applied to a specific problem, a hyper-heuristic aims to find a proper combination of easy-to-implement low-level heuristics that could produce an acceptable domain solution (Burke et al. 2013).

Motivated by the above literature, this paper proposes a modified version of the Clarke Wright algorithm and an enhanced cuckoo search-based hyper-heuristic that selects, in each step, the most suitable low-level heuristic that directly searches for a VRPC solution in the problem’s search space. In fact, the reason for using a hyper-heuristic based on the Cuckoo Search metaheuristic was motivated by the advantages of this metaheuristic. Compared to other heuristics, it has fewer adjustable parameters that need to be configured, and it also has the potential to better balance exploitation and exploration. Regarding our proposed Clarke Wright algorithm, it extends the classical CW to tackle prioritized customers. Both methods are tested with a set of eighteen randomly generated test cases that simulate actual data in the VRP with a predefined capacity of each vehicle, route time, and customer priority. The goal is to minimize the traveling distance of vehicles and reduce the time when moving freight from the depot to prioritized customers. In addition, both methods are also tested on real data from a distribution company operating in Lebanon. The results of the cuckoo search-based hyper-heuristic outperformed the modified Clarke Wright algorithm.

The rest of this paper is organized as follows. Section 2 presents the literature review. Section 3 describes the VRP problem and its formulation. Section 4 presents a description of the classical and modified Clarke Wright algorithms. Section 5 presents the classical Cuckoo Search algorithm. The cuckoo search-based hyper-heuristic is presented in Sect. 6. Section 7 presents the empirical results. Finally, the conclusion is presented in Sect. 8.

2 Literature review

The Vehicle Routing Problem (VRP) is known to be an NP-hard problem; its computational complexity increases exponentially as the number of customers grows (Lenstra and Rinnooy Kan 1981). Researchers have approached the vehicle routing problem using various methods. Exact methods and heuristic algorithms are the most popular ones. Although exact methods can obtain an optimal solution, they are not efficient enough, especially for large-size instances (Abu-Khzam et al. 2014; Captivo et al. 2003). Hence, the requirement to find good solutions quickly (not necessarily the optimal solutions) has led to the development of various heuristic algorithms (Cordeau et al. 2005) and approximate (meta-heuristic) algorithms (Haraty et al. 2018; Tarhini et al. 2016). Some well-structured heuristics can quickly attain feasible solutions for targeted problems. However, the feasible solutions found by heuristic algorithms are not always near the optimal one and thus they cannot guarantee the quality of these solutions (Tarhini et al. 2014).

In fact, previous works have shown that it is easy to apply meta-heuristic algorithms to various VRPs to obtain near to optimal solutions with an acceptable computational time (Yang and Deb 2010; Yang et al. 2012; Khoury et al. 2019); thus, several meta-heuristic algorithms, including Particle Swarm Optimization (PSO) (Nazif and Lee 2012; Kim and Son 2012), the Tabu Search (TS) (Ai and Kachitvichyanukul 2009; Chen et al. 2006), Simulated Annealing (SA) (Gounaris et al. 2014), Genetic Algorithms (GAs) (Zainudin et al. 2015; Jin et al. 2014), and Squeaky Wheel Optimization (SWO) (Zhen 2016), have been proposed to solve VRPs. Nevertheless, the literature does not contain any usage of the Cuckoo Search (CS) algorithm to solve the vehicle routing problem with prioritized customers at the heuristic or hyper-heuristic levels. In fact, an interesting work proposed by Ouaarab et al. (2014) used the CS to solve the traveling salesperson problem (TSP), and the results show that the CS algorithm outperformed some other popular meta-heuristic algorithms. In addition to solving continuous optimization problems (Yang et al. 2012; Gandomi et al. 2013), the CS achieved remarkable performance in constrained optimization problems (Yang and Deb 2013; Bulatović et al. 2013; Bhargava et al. 2013), selecting the web service composition (Chifu et al. 2012), training a neural network (Vazquez 2011), bin packing (Layeb 2011) and manufacturing scheduling systems (Burnwal and Deb 2013).

On the other hand, the literature shows that only a few works have used hyper-heuristics to solve the VRP. Asta and Ozcan (2014) used the HyFlex framework-based hyper-heuristic approach to solve the VRP while Garrido and Castro (2009) used an evolutionary hyper-heuristic approach. Further, Marshall et al. (2014) described a grammatical evolutionary-based hyper-heuristic for the capacitated VRP. To the best of the authors' knowledge, the use of cuckoo search-based hyper-heuristics to solve the VRPC remains unexplored in the literature. Accordingly, this work is motivated to develop a Cuckoo Search-based hyper-heuristic to solve the nonclassical VRP problem with some realistic constraints such as customer priority and constrained route times and to compare its results with those of the Clarke Wright algorithm in order to get better and more satisfactory solutions. Our work considers two types of heuristics: constructive heuristics and improvement heuristics. A constructive heuristic positions customers along a route and creates new routes as needed until all the clients have been assigned a route. On the other hand, improvement heuristics require an initial solution to start with and then they modify the placement of the customers within the routes. Thus, the role of improvement heuristics is to reduce the distance required to visit all the customers.

3 Vehicle routing problem description

3.1 Preliminary

Due to the fierce competition with their rivals, transportation and logistics companies noticed decreases in their profit margins if their trucks were not loaded at the needed capacity and routes were not optimally traversed. Accordingly, efficient and effective measures had to be taken at the operational level by optimally routing vehicles to customers. To

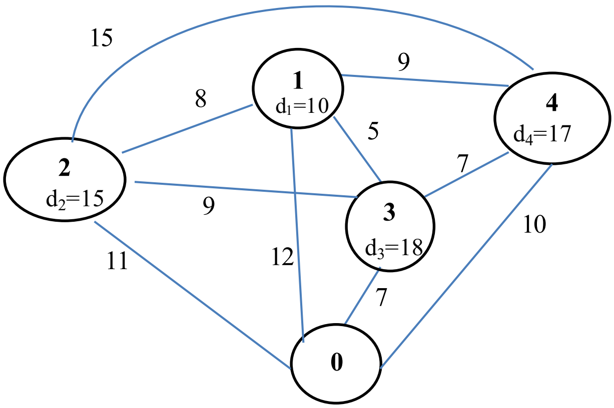

Fig. 2 VRP route establishment, the first permutation

elaborate on the effect of such routing, the following example, illustrated in Fig. 2, shows two different permutations to establish a route between the central depot 0 and 4 customers (represented as nodes), where the distance between each customer is displayed on the edge between the nodes and the demand d_i is found in the node itself. Assume that the maximum capacity per vehicle is 40. The two permutations lead to two different costs. Assume that the first permutation travels from depot 0 to customer 1 and then returns back to depot, from which it travels to customer 2 and back again to depot. Thus, the path will be 0–1–0, 0–2, 0. The demand in this trip is fulfilled with a cost of $12+12+11+11=46$. Assume that in the second permutation we traverse the path 0–1–2–0. The demand $(10+15)$ of this trip is less than the vehicle capacity and thus it is fulfilled; therefore, the cost of this trip would be $12+8+11=31$. This obviously shows that the selection of the appropriate route would reduce the cost of the trip between customers.

3.2 Mathematical formulation

Given a set of customers $\mathcal{C}=\{1,\ldots,n\}$ with priorities $\gamma_i \in \gamma=\{1\ldots n\}$ and demands $di \in \mathcal{D}=\{1,\ldots k\}$ for a product that must be served using a set of vehicles. The vehicles are situated at a central depot to which they must return after serving customers. The cost of traveling between customer i and customer j is related to the distance traversed and is denoted by cij. Each vehicle has a given maximum capacity Q. In a VRP, we need to determine a routing schedule that minimizes the total cost of deliveries such that each route starts and ends at the depot; further, every customer belongs exactly to one route, and the vehicle capacity is not exceeded in any route. The route represents a sequence of customers for each vehicle.

The VRP is known to be an NP-hard problem (Lenstra and Rinnooy Kan 1981), and we represent it using graphs. Let $G\ (V_G, E_G)$ be a graph in which the following exists: vertex $v_i \in V_G$ represents a customer to be visited, where $|V_G|=n$; the customer's demand d_i represents the number of products requested by customer v_i; the edge $e \in E_G$ joins the two vertices v_i and v_j and represents the existence of a flow between customer v_i and customer v_j; and the cost of traversing this edge e, c_{ij}, represents the distance between the two customers v_i and v_j. There exists m vehicles, where $m_i \in \mathcal{M}=\{1,\ldots y\}$, with capacities Q_j, where $j=1\ldots q$, that start and end at the central depot, which is at vertex 0. The problem to be handled is to determine the cycles $R_1,\ldots R_n$ for the vehicles that start from vertex 0 and service all vertices such that the load of vehicle j does not exceed its capacity Q_j, and the total cost of the cycles is minimized.

To sum up, in a VRP, we need to determine a routing schedule that minimizes the total cost of deliveries such that the following constraints are met:

1. each route starts and ends at the depot,
2. every customer belongs exactly to one route,
3. the total demand on each route does not exceed the vehicle capacity Q,
4. the total duration of each route does not exceed a predefined limit T,
5. a vehicle can do more than one route, and
6. a preference priority is assigned to every customer such that γ preferred customers could not be visited in the same route. The route represents a sequence of customers for each vehicle.

Given a set of customers $\mathcal{C}$, a set of vehicles $\mathcal{M}$, and the operating time at each customer t_i^o, the following represents our proposed mathematical formulation:

$$\min \Sigma_i \Sigma_j \left(t_{ij} + t_j^o \right) x_{ijm}$$

which is subject to

$$\sum_{i=0}^{n} x_{ijm} = 1 \quad \forall j \in C, m \in M \tag{1}$$

$$\sum_{j=0}^{n} x_{ijm} = 1 \quad \forall i \in C, m \in M \tag{2}$$

$$\sum_{i}^{n} \sum_{j,i \neq j}^{n} x_{ijm} q_i \leq Q \quad \forall m \in M \tag{3}$$

$$\sum_{i}^{n} \sum_{j,i \neq j}^{n} x_{ijm} P_i \leq \gamma \quad \forall m \in M \tag{4}$$

$$\sum_{k} x_{ijk} \left(t_{ij} + t_i^o \right) \leq T; \quad x \in \{0, 1\} \tag{5}$$

In the first two constraints, x_{ijm} represents the degree of a vertex that ensures that exactly one edge enters and exactly one leaves each vertex associated with a customer, respectively. Constraint 3 ensures that the vehicle capacity for each vehicle m does not exceed the defined maximum. Constraint 4 ensures that each tour does not include more than γ prioritized customers. In constraint 5, we ensure that the travel time for each vehicle (maybe for more than one tour) does not exceed a specified time limit.

4 Clarke–Wright algorithm

4.1 The classical Clarke–Wright algorithm

The Clarke–Wright savings algorithm is one of the known heuristics that can be used to solve the VRP. It was developed in 1964 and is classified as a constructive method in which tours are built up by adding nodes to partial tours or combining subtours to meet the capacities and costs (Clarke and Wright 1964). It applies to problems for which the number of vehicles is not fixed (it is a decision variable), and it works equally well for both directed and undirected problems. When two routes (0,…,i,0) and (0,j,…,0) can feasibly be merged into a single route (0,…,i,j,…,0), a distance saving $S_{ij} = c_{i0} + c_{0j} - c_{ij}$ is generated. A description of the classical Clarke Wright (CW) algorithm is given as follows.

Algorithm 1 – Classical Clarke Wright

1. Starting solution: each of the *n* vehicles serves one customer.
2. For all pairs of nodes i, j, i…j, calculate the *savings* for joining the cycles using edge [i,j]: $S_{ij} = c_{0i} + c_{0j} - c_{ij}$.
3. Sort the savings in decreasing order.
4. Take edge [i,j] from the top of the savings list. Join two separate cycles with edge [i,j] by deleting *(0,j)* and *(i,0)* and introducing *(i,j)* if
   (i) the nodes belong to separate cycles
   (ii) the maximum capacity of the vehicle is not exceeded
   (iii) i and j are the first or last customer on their cycles, one starting with *(0,j)* and one ending with *(i,0)*
5. Repeat (4) until the savings list is formed or the capacities do not allow more merging.

4.2 The modified Clarke–Wright algorithm

A modified version of the Clarke–Wright algorithm (CW) can be implemented by applying two new constraints to the algorithm. The first is the driver time parameter, which should not be less than the upper bound, U_t; and the second parameter is the customer preference priority, P_p, where the route should not contain more than *n* preferred customers. In the modified CW algorithm, customers are clustered by vehicles. First, compute the Euclidean distance matrix ($d_{i,j}$) according to the following equation:

$$d_{i,j} = \sqrt{\left(X_i - X_j\right)^2 + \left(Y_i - \mathrm{Y}\mathrm{j}\right)^2}$$

where X_i, Y_i and X_j, Y_j are the geographical locations of customers *i* and *j*, respectively. Second, the savings value between customers *i* and *j* is calculated as follows:

$$S_{i,j} = d_{0,j} - d_{i,j}$$

where $d_{0,j}$ is the traveling distance between the depot and customer *j*, and $d_{i,j}$ is the traveling distance between customers *i* and *j*. Bodin et al. (1983) updated the Clarke–Wright formulation. After the calculation, all savings values are collected in the savings list and calculated as follows:

$$S_{i,j} = d_{0,j} + d_{j,0} - d_{i,j}$$

A description of the modified Clarke Wright (CW) algorithm is given as follows.

Algorithm 2 – Modified Clarke Wright

1. Import_Instance()
2. Calculate the Saving Method as $S_{i,j} = d_{0,j} + d_{j,0} - d_{i,j}$
3. IF(cst_Route_time && cst_Priority_cdt) Then
   a. For all pairs of nodes i, j, i…j, calculate the *savings* for joining the cycles using edge [i,j]
   b. Sort the savings in decreasing order.
   c. Take edge [i,j] from the top of the savings list. Join two separate cycles with edge [i,j] by deleting *(0,j)* and *(i,0)* and introducing *(i,j)* if
      i. the nodes belong to separate cycles
      ii. the maximum capacity of the vehicle is not exceeded
      iii. i and j are the first or last customer on their cycles, one starting with *(0,j)* and one ending with *(i,0)*
   d. Repeat (c) until the savings list is formed or the capacities do not allow more merging.
   EndIf

5 A classical cuckoo search algorithm

Nature-inspired meta-heuristics have proven their adeptness and efficacy on a wide range of problems, which has contributed to the introduction of new nature-inspired meta-heuristic solutions over the years. All meta-heuristic algorithms share two important characteristics, which are intensification and diversification. The dominance of these algorithms comes from the fact that they imitate the best features of nature, especially those of biological systems that evolved from natural selection over millions of years.

The Cuckoo Search (CS) is one of the latest nature-inspired metaheuristic algorithms that belong to the swarm intelligence category. The results of several studies show that the CS has better performance than other natural algorithms such as PSO and the GA (Yang and Deb 2009, 2010) when solving continuous optimization problems. Yang and Deb (2009) first introduced the CS in 2009 based on the brood parasitism behavior of cuckoos. It is inspired by the aggressive reproduction behavior of cuckoo birds that lay their eggs in communal nests through which they might remove others' eggs to increase the hatching probability of their own eggs. If a host bird discovers the eggs are not its own, it will either throw away these strange eggs or simply desert its nest and build a new nest elsewhere. The cuckoo's egg might be found by the host bird with a certain probability $P_{ed} \in [0,1]$.

The behavior of cuckoos is modeled by the CS algorithm. After each step, the worst solutions are discarded and new solutions are generated. This models that the worst nests are being identified by host birds, which means that they have to be discarded and new nests are created by host birds. Then, in each iteration, a cuckoo solution tries to replace a nest among the solution nests to get the best solution after each repetition. Thus, solution X_i^{t+1} is generated from solution X_i^t of cuckoo i by performing a Lévy flight (a method for generating eggs) as per the equation below:

$$X_i^{t+1} = X_i^t + \alpha \oplus \text{Lé vy}\,(s,\ \lambda)$$

Table 1 The set of considered low-level heuristics used for the improvement step

llh-imp#	Description of the Low-level heuristic for the improvement step
llh-imp1	Re-Order: Reorder the sequence of customers (in a random tour) in order to find a better order of customers along the route
llh-imp2	Re-allocate: Remove a random customer from a random tour and add it to another tour if and only if the solution is improved
llh-imp3	Swap: Randomly swap 2 customers from 2 different tours if and only if the solution is improved
llh-imp4	Destroy: Destroy a part of the solution and reconstruct it using the proposed constructive heuristic if the solution is improved
llh-imp5	Merge: Merge an existing tour into other tours if the solution is improved

Table 2 The set of Lévy-flight low-level heuristics used for the perturbation step

llh-lévy#	Description of Low-level heuristic for perturbation step
llh-lévy1	Swap1: Randomly swap 2 customers from 2 different existing tours
llh-lévy2	Swap2: Randomly swap 3 customers from 3 different tours if that many exist
llh-lévy3	Insert/Delete: Remove an allocated customer from a random tout and re-allocate it to another tour
llh-lévy4	NoisyDestroy: Destroy a part of the solution and reconstruct it using the proposed constructive heuristic
llh-lévy5	NoisyMerge: Merge an existing tour into the other tours

where $\alpha < 0$ is the step size, which is related to the scale of the problem of interest. In the majority of cases, the most commonly used value of α is 1. The most important characteristic of Lévy flights is their intensive search around a solution and the occasional big steps of Lévy flights can minimize the probability of falling into the local optima. A Lévy flight is modeled as a probability density function:

$$\text{Lé vy}(s, \lambda) \sim s^{-\lambda}, (1 \ < \ \lambda \ \leq \ 3)$$

This has an infinite variance with an infinite mean. Here, s is the step size drawn from a Lévy distribution. A detailed description of the CS can be found in the work done by Yang and Deb (2010).

6 Proposed swarm intelligence cuckoo search based hyper-heuristic for the VRPC

The hyper-heuristic algorithm (Algorithm 3) that we propose in this paper uses the Cuckoo Search algorithm (Yang and Deb 2009) to combine low-level heuristics such that a domain specific solution, sol_{domain}, (i.e., the VRPC) would be guided towards the optimal or a near-optimal solution. In our approach, we have used two sets of low-level heuristics. The first set is applied sequentially to improve the solution; this set is named *llh-imp* and shown in Table 1. The second set is applied according to a selection method using the levy-flight concept; this set is named *llh_levy* and shown in Table 2.

6.1 Egg representation

In this work, we assume that a cuckoo lays a single egg in one nest; thus, an egg in a nest is a solution represented by one individual in the population, while the nest is the container of that new cuckoo egg and its abandonment involves its egg being replaced in the population by a new one. The cuckoo egg is defined as follows:

$$\text{cuckoo}_{\text{egg}} = \left(llh\_levy, sol_{domain}\right)$$

where *llh_levy* represents a sequence of n low-level heuristics that will be applied on an improved domain solution (VRPC), sol_{domain}, in the order that they appear in the sequence, and sol_{domain} represents the domain solution (i.e., the VRPC).

6.2 Host nest initialization

In the initialization stage, an initial solution, sol_{domain}, is developed using a constructive heuristic. The constructive heuristic constructs the first current solution from scratch given a set of predefined rules. In this work, the Constructive Algorithm is executed as follows.

a. *Step 1* Choose the customer who has the highest priority.
b. *Step 2* Construct the tour by trying to add the nearest customer to the tour.
c. *Step 3* When the tour is unable to add any more customers from the rest, a new tour will be created and the same scenario will be repeated in the second step.

6.3 Local search (intensification and diversification) and levy flight

To reduce the probability of their eggs being discovered, some cuckoo species have evolved in such a way that they can engage in a kind of surveillance on nests likely to be a host (Payne and Sorensen 2005). This work uses improvement and perturbation heuristics to help the cuckoo bird's eggs imitate the pattern and shape of the host nest's eggs, and therefore they have a good chance to survive. These heuristics are iteratively performed in two separate steps until a stopping condition is satisfied. The new solution is accepted based on the simulated annealing acceptance criterion (Metropolis criterion). In this way, the optimization process might be prevented from getting stuck in a local optimum.

In the improvement step, a permutation of all low-level heuristics in the set *llh-imp*, shown in Table 1, is applied sequentially to the current solution. Each of the low-level heuristics $llh\text{-}imp_i$ is applied repeatedly before applying $llh\text{-}imp_{i+1}$ as long as $llh\text{-}imp_i$ is able to improve the solution.

In the perturbation step, we simulate the moves conducted by cuckoos in the search space via Lévy flights. The cuckoo chooses a direction and step size from its current nest to search for the best nest in a restrictive range of its current nest according to the value of the Lévy flights, which depend on two factors: the remaining time for a cuckoo to lay its egg (i.e., the algorithm execution time) and the performance enhancement of the solution. This step guarantees a certain level of diversification by selecting one of the following three strategies to be applied according to Eq. 2.

- Strategy 1: From Table 2, choose a random permutation heuristic $llh\_lévy_i$ and apply it to the candidate solution.
- Strategy 2: From Table 2, randomly choose a permutation of the perturbation heuristics and apply it as long as the candidate solution in hand is improved. Hence, the improving low-level-heuristic is applied repeatedly as long as it is able to improve the solution.
- Strategy 3: From Table 2, choose the perturbation heuristic that is known to have the best reward.

The strategy selection depends on the variable Lévy described in Eq. 6:

$$Levy(k,t) = \begin{cases} 1, & if\, k = 2\, and\, t < \frac{T_l}{3} \\ 2, & if\, k = 2\, and\, t > \frac{T_l}{3} \\ 3, & if\, k \geq 3 \end{cases} \tag{6}$$

where T_l is the algorithm's remaining execution time. K is calculated as per the equation below:

$$K = Levy(k,t)^{-1} + m \tag{7}$$

The value of m is based on whether the performance improvement level falls below 30%, from 30 to 60%, or above 60%.

$$m = \begin{cases} 1, & if\, \alpha * L < L + 0.3 * L \\ 2, & if\, L + 0.3 * L < \alpha * L < L + 0.6 * L \\ 3, & if\, \alpha * L > L + 0.6 * L \end{cases} \tag{8}$$

L indicates the performance level and is calculated as follows:

$$L = \left| \frac{Objective_{Function(Current\ Solution)} - Objective_{Function(Best\ Solution)}}{Objective_{Function(Current\ Solution)}} \right|.$$

6.4 Termination criterion

The termination criterion or stopping condition is the condition that ends the search. In this work, the hyper-heuristic will stop searching when the number of nonimproved solutions reaches a defined threshold (related to the number of customers) or the execution time reaches a certain limit ω. The termination criterion is defined in the following equation:

Algorithm 3- Swarm intelligence Cuckoo Search based Hyper-heuristic

Inputs: *Ç; γ; D; searchSpace*; *llh-imp; llh-lévy; diverseStrategies;* **Output**: sol_{opt}
begin
$sol_{domaint}$ = **Constructive_Heuristic_Domain_Sol(***Ç; γ; D; searchSpace, L***)**
$sol_{domaintOpt}$ = **Improvement_Local_Search_Heuristic (**sol_{domain}*; llh-imp, accept_criteria***)**
$cuckoo_{egg}Set$ = **Create_Diverse_Set_Cuckoo_Eggs***(llh-lévy, diverseStrategies,* $sol_{domainOpt}$**)**
foreach $cuckoo_{egg}$ **in** $cuckoo_{egg}Set$ **do**
  $cuckoo_{egg}$ = **Evaluate_LévyFlight***(*$cuckoo_{egg}$*, LévyFlightsEqu,*$sol_{domainOpt}$ *,L, accept_criteria)*
end foreach
nestSet = **Create_Random_Nest_Eggs***(llh-lévy, diverseStrategies,* $sol_{domainOpt}$**)**
foreach $nest_{egg}$ **in** *nestSet* **do**
  $nest_{egg}$ = **Evaluate_ LévyFlight***(*$nest_{egg}$*, LévyFlightsEqu,* $sol_{domainOpt}$ *, L, accept_criteria)*
end foreach
while *(stopping condition not satisfied)* **do**
$cuckoo_{egg}$ = **Get_Random_Cuckoo_Egg***(*$Cuckoo_{egg}Set$*)*
$nest_{egg}$ = **Get_Random_Nest***(nestSet)*
$cuckoo_{egg}$ = **Modify_** *llh-lévy(*$cuckoo_{egg}$*)*
$cuckoo_{egg}$ =**Evaluate_ LévyFlight(***$cuckoo_{egg}$,LévyFlightsEqu,*$sol_{domainOpt}$ *,L, accept_criteria)*
if *(***Fitness***(*$cuckoo_{egg}$*)* < **Fitness***(*$nest_{egg}$*))* **then**
begin
$nest_{egg}$ = $cuckoo_{egg}$
best_ llh-lévy = cuckoo. llh-lévy
end if
$sol_{domainOpt}$ = **Update_Optimal_Solution***(cuckoo, nest,* $sol_{domainOpt}$*)*
nestSet = **Replace_Worst_Nests***(nestSet, Percent_eggDisc)*
end while
return $sol_{domainOpt}$
end

$$T(\mathrm{ms}) = \mathrm{Min}\{\max\{\varpi, nbofcustomers * 1000\}, \text{nb of consecutive non-improved solutions}\}. \tag{9}$$

7 Experimental results

In this section, the results of a Cuckoo Search-based hyper-heuristic are presented and are compared with those of the modified Clarke Wright algorithm. These algorithms are tested on real and synthetic data. A set of *eighteen synthetic* test cases were generated based on the methodology adopted by Tarhini et al. (2016); each of these test cases has different numbers of customers and vehicles. Further, real data were collected from a distribution company operating in Lebanon. The company distributes products to more than 200 customers in several industries across Lebanon. This section will present the results of the synthetic data first followed by the results of the real data.

7.1 Synthetic data

Each of the eighteen synthetic test cases is represented by four main components. They are the distance matrices, the route time matrices, the demands of customers and the customer's priority. Table 3 summarizes the results of the modified Clarke Wright (CW) algorithm and the Cuckoo Search-based hyper-heuristic (CsHh) applied on the eighteen test cases of

Table 3 Comparison results of the modified Clarke Wright and the cuckoo search based hyper-heuristic

Problem	Cn	Vn	Extended Clarke Wright			Cuckoo search-based hyper-heuristic		
			O.F (cost)	Distance	Exe time (s)	O.F. (cost)	Distance	Exe time (s)
Tc-1	8	3	150	556	0.59	150	556	0.72
Tc-2	8	4	198	716	0.74	183	701	0.78
Tc-3	10	4	211	691	0.74	191	671	0.92
Tc-4	10	6	211	859	0.89	199	841	0.99
Tc-5	10	7	453	1063	1.08	422	1032	1.04
Tc-6	11	5	201	701	0.75	196	683	1.00
Tc-7	20	5	265	760	0.80	242	744	2.00
Tc-8	25	5	298	783	1.00	281	771	2.01
Tc-9	30	5	314	802	1.01	287	792	2.11
Tc-10	40	5	342	821	1.03	313	801	4.80
Tc-11	40	7	328	811	1.08	302	797	4.90
Tc-12	50	5	389	877	1.11	344	823	5.00
Tc-13	50	7	372	862	1.12	322	808	6.70
Tc-14	75	10	413	895	1.18	388	870	7.80
Tc-15	100	12	453	916	1.32	413	890	10.47
Tc-16	100	20	429	899	1.40	389	871	15.36
Tc-17	150	25	530	1028	1.47	481	998	17.00
Tc-18	200	30	577	1103	1.67	512	1089	21.51

TC1-TC18. The second column of Table 3 (Cn) represents the number of customers to be visited and the third column (Vn) represents the number of vehicles used. The remaining columns show the Objective function, traversed distance, and execution time in milliseconds for both the CW algorithm and the CS Hyper-heuristic. The CW algorithm and CsHh were both implemented using VB.Net.[1] Furthermore, the tests were carried on a PC with an Intel core i3 CPU operating at 1.2 GHz, 4 GB of Ram, and Windows 10.

In Table 3, for each instance, several parameters are shown and compared: the objective function value (best found cost), the distance of the best route, and the time taken to find this best route. Figure 3 illustrates the results shown in Table 3 in which it is found that the CSs hyper-heuristic outperformed the CW algorithm in terms of the Objective function value and the route distance for the eighteen test cases.

Moreover, it is worth noticing from Table 3 that the magnitude of the performance improvement of the CS over the CW algorithm incrementally increases as the problem size gets larger; this is illustrated in Fig. 4. One explanation for these results is that for small problem sizes, the CS hyper-heuristic was able to cover similar diverse solutions in the narrow solution space as the CW and thus it gave similar results. However, for large problem sizes, the diversification method in the CS enabled this hyper-heuristic to find a better solution than the CW by covering all possible combinations.

Further, although the CS algorithm outperformed the CW in large problem sizes with the chance of obtaining a better diversified solution space, nevertheless, the execution time

[1] The code is found at the following link: https://github.com/abbastarhini/VRP.git.

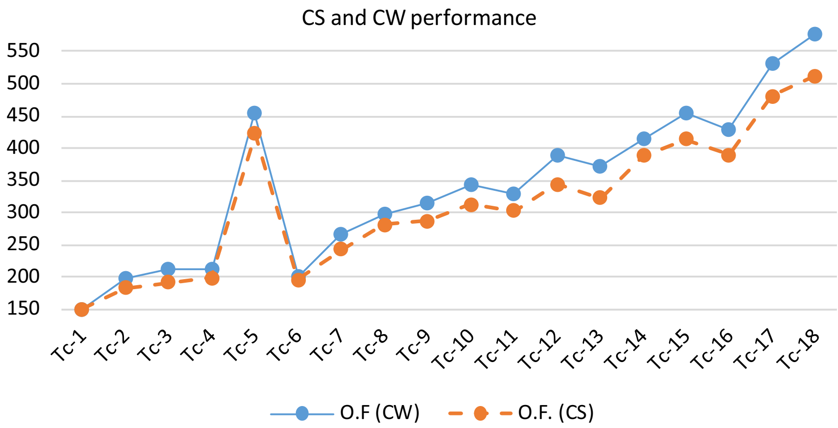

Fig. 3 A comparison of the Cuckoo Search-based Hyper-heuristic results with the Clarke Wright algorithm results

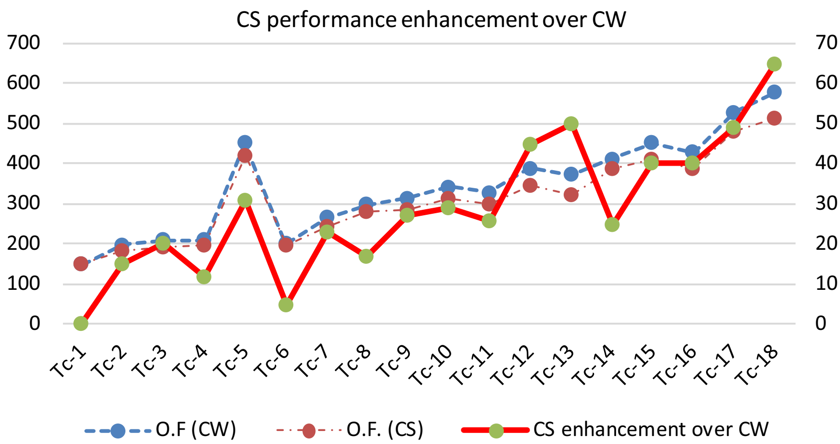

Fig. 4 The magnitude of the CS performance enhancement over the CW

for the CS (21.5 s) was much longer than that of the CW (1.67 s), as shown in Fig. 5; however, this is still acceptable since such solutions are produced off-line. In Fig. 5, the x-axis shows the test cases, the left y-axis is the execution time, and the right y-axis shows the performance enhancement of the CS over the CW. In fact, one reason for the CS hyper-heuristic having a higher computational time than the CW goes back to the fact that the operations performed in any hyper-heuristic are more complex than those done in the CW's iterations. In the CS hyper-heuristic, all operations, ranging from the initial constructive solutions to the local search improvement method and ending with the perturbation methods, take more time than what is done in the CW, which is based on the notion of saving operations; thus, the CW scores very high on simplicity and speed. In fact, this is clearly noticed as the problem size gets bigger where the diversified set of solutions needs more computational efforts to be created than in small-sized problems where the computational efforts are close to those in the CW.

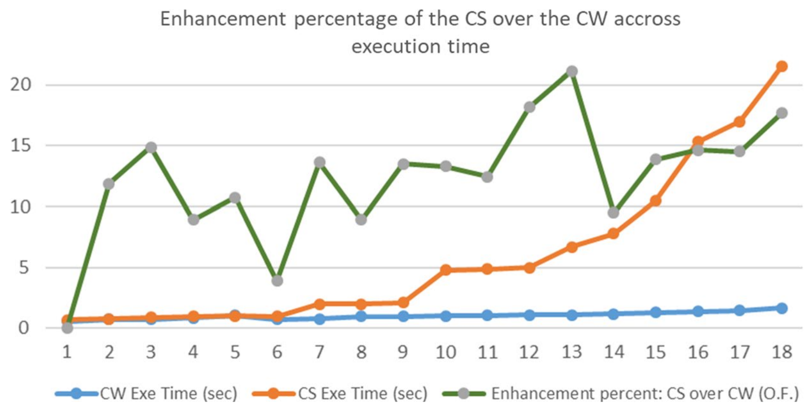

Fig. 5 Enhancement percentage of the CS algorithm over the CW across execution times

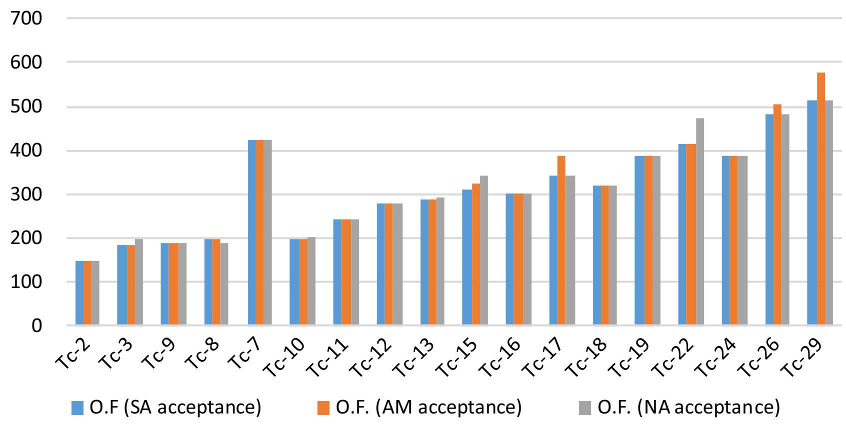

Fig. 6 Cuckoo search-based hyper-heuristic tuned on three acceptance criteria

In addition, we tuned the cuckoo-search hyper-heuristic performance by using three different acceptance criteria. The first acceptance criterion is Naive Acceptance, which allows the nonimproving solutions to be accepted with a probability of 0.5. The second adopted acceptance criterion is All Moves, which accepts the candidate solution regardless of its objective function. As shown in Fig. 6, the CS hyper-heuristic is best tuned using the SA acceptance criterion because it enables it not to be stuck in a local optimum.

In addition, further tuning is applied to the termination criteria to measure the effect on the execution time and thus on the solution quality. There are two termination criteria: the execution time reaches a certain limit ω, as mentioned in Eq. 4, or the number of nonimproved solutions reaches a predefined threshold (related to the number of customers). The second termination criterion is when the number of nonimproved solutions reaches a predefined threshold (related to the number of customers):

$$T(\mathrm{ms}) = \max\{\varpi, nbofcustomers * 1000\}. \tag{5}$$

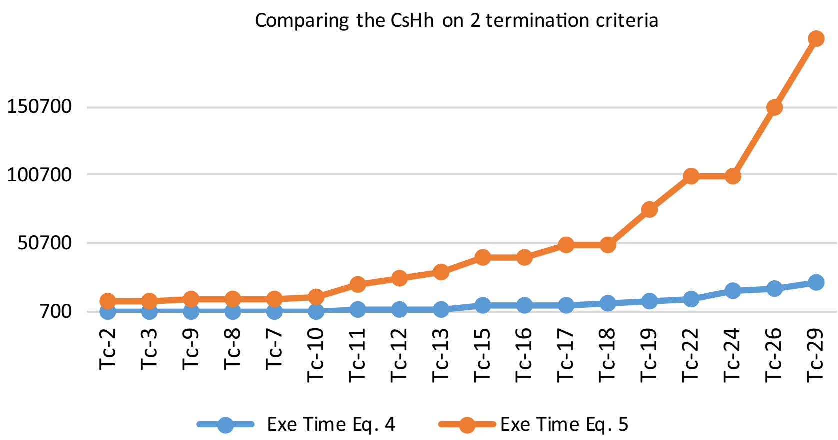

Fig. 7 Cuckoo search-based hyper-heuristic execution time based on two different termination criteria

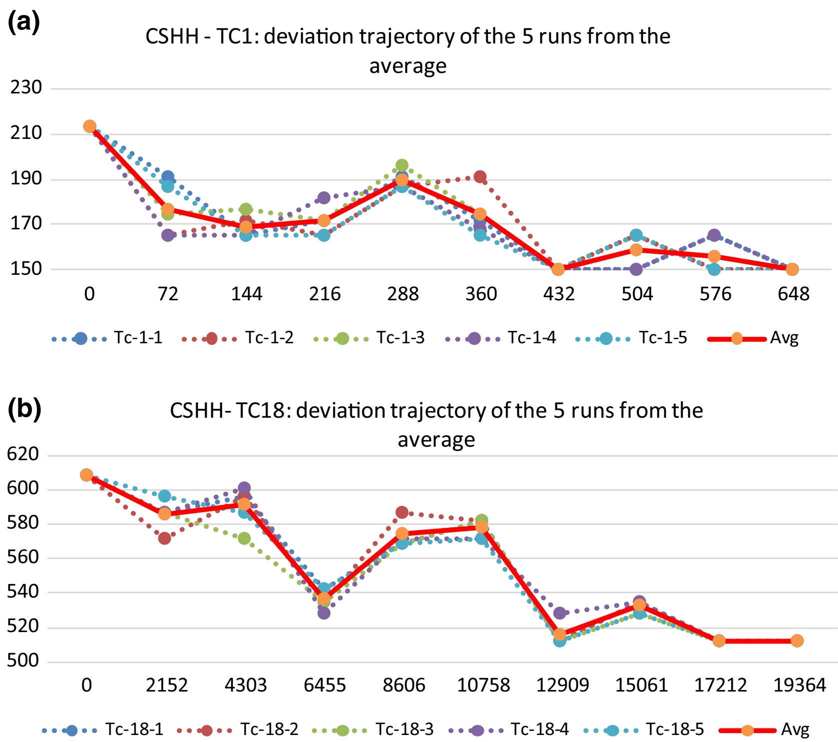

Fig. 8 **a** TC1: Deviation trajectory of the five runs from the average for the CsHh algorithm. **b** TC18: Deviation trajectory of the five runs from the average for the CsHh algorithm

Figure 7 shows the results of the CsHh for both termination criteria. The first criterion (Eq. 4) provided more efficient solutions. This is because Eq. 5 is based only on the

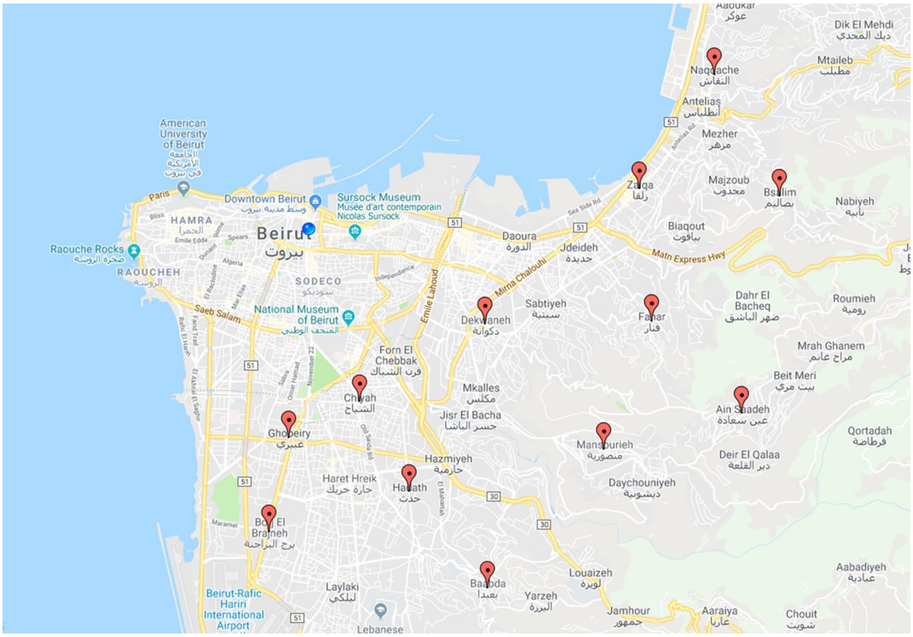

Fig. 9 Customers' locations over an area of 60 km^2

number of customers. When it grows, the termination criterion will grow exponentially and will never stop, even if the solution is found.

Finally, in order to test the solution stability of our algorithm, we conducted 5 runs of the algorithm for each test case. Figure 8a and b correspondingly show the results of executing the 5 runs on the first test case (Tc-1) with a small number of customers (8 customers) and the last test case (Tc-18) with a large number of customers (200 customers). The x-axis represents the execution time and the y-axis represents the value of the objective function. Each of the dotted colored lines represents one run of the algorithm. The solid red line represents the average of the 5 runs for the same test case. The results clearly show that the algorithm's execution is stable, where trajectory of the 5 runs are evenly distributed around the average of these runs with minimal deviation.

7.2 Case study: applying the algorithms on real data (Fueled application)

In this section, we compare the results of the CsHh and CW on real data collected from a distribution company operating in Lebanon (IBC 2019). The company distributes products to more than 200 customers in several industries across Lebanon. We applied our solution within the same time frame to a scenario with 12 customers distributed over an area of 60 km^2 shown in Fig. 9. The distance between the customers is determined via the Google Maps API. The constraints placed by our client (the Distribution Company) limit the tour to being completed within a maximum of 4 h using only three vehicles. In addition, three customers had a higher priority than others (Ghobeiry, Chiyah, and Mansourieh) and need to be served within the first hour. Further, the service times are

Table 4 Solution generated by the Clarke Wright algorithm for the distribution company's real data

Tour 1	Route	Beirut	Chiyah	Ghobeiry	Borj Brajneh	Baabda	Dikwaneh	Beirut	Total
	Distance	3.9 km	2.5 km	3.3 km	7.7 km	7.8 km	5.9 km		31.1 km
	Route time	9 min	5 min	8 min	42 min	16 min	12 min		92 min
Tour 2	Route	Beirut	Hadath	Mansourieh	Ain Saadeh	Bsalim	Beirut		Total
	Distance	6.2 km	6.8 km	6.1 km	8.5 km	12.8 km			40.4 km
	Route Time	96 min	105 min	8 min	14 min	17 min			240 min
Tour 3	Rout e	Beirut	Fanar	Naqqache	Zalka	Beirut			Total
	Distance	9.5 km	6.9 km	3.6 km	10 km				30 km
	Route time	16 min	13 min	12 min	15 min				56 min
Total tour distance				101.5 km					

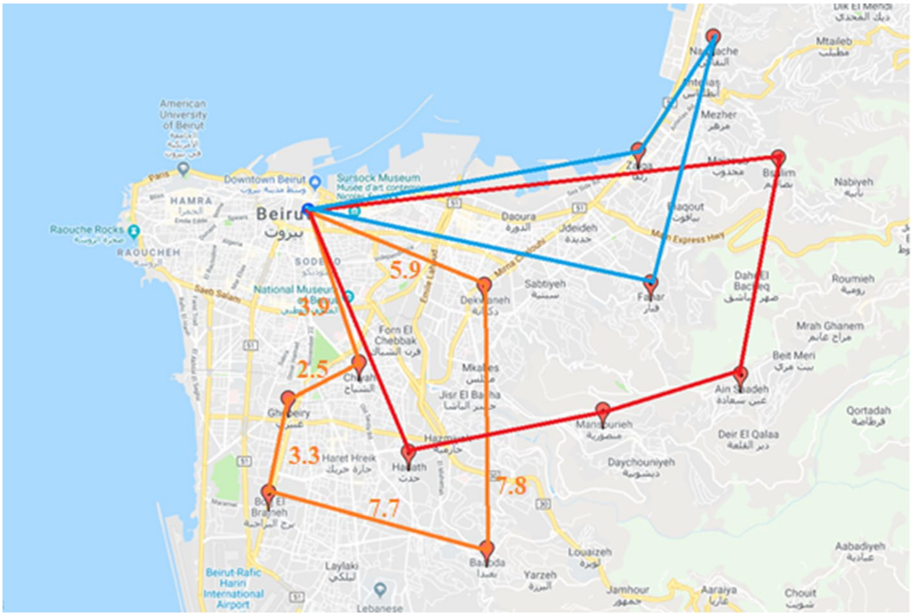

Fig. 10 The solution generated by the Clarke Wright algorithm

almost the same for all customers and will not exceed 10 min; thus, it is assumed that service time is counted within the customer travel time.

The solution generated by the Clarke Wright algorithm is detailed in Table 4 and shown in Fig. 10. The total distance needed to serve the 12 customers is 101.5 km and the maximum time needed for the three tours is within 240 min.

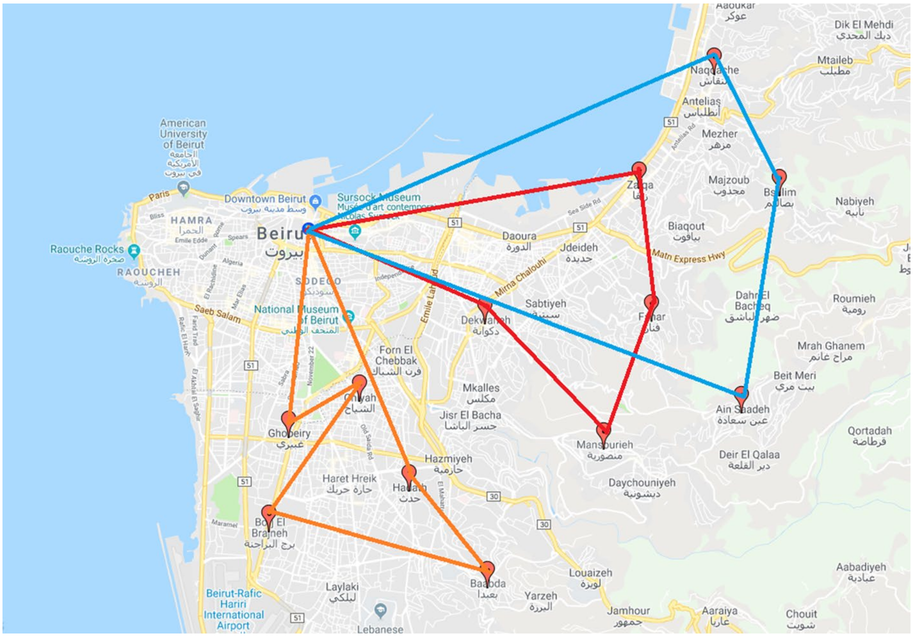

Fig. 11 The solution generated by the cuckoo search-based hyper-heuristic

Table 5 Solution generated by the cuckoo search-based hyper heuristic for the distribution company's real data

Tour 1	Route	Beirut	Ghobeiry	Chiyah	Borj Brajneh	Baabda	Hadath	Beirut	Total
	Distance	4 km	3 km	5 km	7.2 km	3 km	6.1 km		28.3 km
	time	5 min	8 min	10 min	18 min	42 min	96 min		179 min
Tour 2	Route	Beirut	Dekwaneh	Mansourieh	Fanar	Zalka	Beirut		Total
	Distance	6.2 km	5.1 km	6.1 km	3.4 km	10 km			30.8 km
	time	15 min	13 min	12 min	10 min	15 min			65 min
Tour 3	Route	Beirut	Ain Saadeh	Bsalim	Naqqache	Beirut			Total
	Distance	14.1 km	8.1 km	4.2 km	11 km				37.4 km
	time	25 min	14 min	11 min	15 min				65 min
Total tour distance				96.5 km					

The solution generated by the Cuckoo search hyper-heuristic is shown in Fig. 11 and detailed in Table 5. The total distance needed to serve the 12 customers is 96.5 km and the maximum time needed for the three tours is within 179 min.

It is clear from these results that the CsHh is able to get a better quality solution than the CW in terms of the route distance and route time. Nevertheless, the CW scored very high on simplicity and speed.

8 Conclusion

In this paper, we presented our vision to minimize the traveling distance of vehicles when moving cargo to prioritized customers. The problem under study (the VRPC) is a combinatorial optimization management problem that seeks the optimal set of routes traversed by a vehicle to deliver products to prioritized customers in the shortest possible time. We presented a modified version of the Clarke Wright algorithm and a Cuckoo Search-based Hyper-heuristic for the VRPC with the purpose of comparing the performances of these competing methods.

The Clarke–Wright (CW) savings algorithm is one of the popular heuristics known to efficiently solve the VRP. The Clarke–Wright proved to be very quick and simple to implement. However, in contexts where vehicle routes span long distances to cover a large number of customers, it is worthwhile to explore other methods that reduce the distance covered and the needed time. The Cuckoo search is one of the latest nature-inspired metaheuristic algorithms that belong to the swarm intelligence category. The proposed Cuckoo search-based hyper-heuristic combines low-level heuristics such that a domain specific solution would be guided towards the optimal or a near-optimal solution. In fact, the CW algorithm has been enhanced to solve the VRPC. One contribution of this work is modifying the classical VRPC mathematical model to include prioritized customers. Another contribution is enhancing the CW algorithm to solve the VRP with prioritized customers. A third contribution is proposing and testing a unique hyper-heuristic (CsHh) that has not been used before for solving the VRPC and comparing it with a modified version of the modified CW.

The results indicate that our proposed CsHh outperformed the modified CW algorithm. Mainly, the focus in the CsHh was on the intensification and diversification generation method and the acceptance criteria that guarantees escaping from a local optima. A unique constructive method has been used to boost the quality of the initial population, and, consequently, the quality of the solution space was improved in every generation using a local search stimulated by a Lévy flight. Clearly, this process affected the final results since it aided in yielding better solutions than the CW. An additional contribution is the practicality of the proposed method. We applied this solution to a distribution company (IBC 2019) operating in Lebanon. The company distributes products to more than 200 clients in several industries. The distribution company believes that our CsHh solution offers an attractive alternative to their commercial solver since it is more flexible at handling the company constraints regarding the customers' priority, capacity, and number of trucks, and the computation time is reasonable as long as it significantly reduced the time and cost of the distribution process.

References

Abu-Khzam, F. N., Jahed, K. A., & Mouawad, A. E. (2014). *A hybrid graph representation for exact graph algorithms*. arXiv preprint arXiv:1404.6399.

Ai, T. J., & Kachitvichyanukul, V. A. (2009). Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem. *Computers & Industrial Engineering, 56*(1), 380–387.

Akpinar, S. (2016). Hybrid large neighbourhood search algorithm for capacitated vehicle routing problem. *Expert Systems Applications, 61,* 28–38.

Asta, S. & Ozcan, E. (2014). An apprenticeship learning ¨ hyper-heuristic for vehicle routing in hyflex. In *IEEE symposium on evolving and autonomous learning systems (EALS)* (pp. 65–72).

Baradaran, V., Shafaei, A., & Hosseinian, A. H. (2019). Stochastic vehicle routing problem with heterogeneous vehicles and multiple prioritized time windows: Mathematical modeling and solution approach. *Computers & Industrial Engineering, 131,* 187–199.

Bhargava, V., Fateen, S. E. K., & Bonilla-Petriciolet, A. (2013). Cuckoo search: A new natureinspired optimization method for phase equilibrium calculations. *Fluid Phase Equilibria, 337,* 191–200.

Bodin, L., Golden, B., Assad, A., & Ball, M. (1983). Routing and scheduling of vehicles and crews. The state of the art. *Computers & Operations Research, 10*(2), 63–211.

Bulatović, R. R., Dordević, S. R., & Dordević, V. S. (2013). Cuckoo search algorithm: A metaheuristic approach to solving the problem of optimum synthesis of a six-bar double dwell linkage. *Mechanism and Machine Theory, 61,* 1–13.

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., & Woodward, J. R. (2010). A Classification of hyper-heuristic approaches. In M. Grendreau & J. Y. Potvin (Eds.), *Handbook of metaheuristics, international series in operations research and management science* (Vol. 146, pp. 449–468). Cham: Springer.

Burke, E. K., et al. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society, 64,* 1695–1724.

Burnwal, S., & Deb, S. (2013). Scheduling optimization of flexible manufacturing system using cuckoo search-based approach. *The International Journal of Advanced Manufacturing Technology, 64*(5–8), 951–959.

Captivo, M., Clímaco, J., Figueira, J., Martins, E., & Santos, J. L. (2003). Solving multiple criteria 0-1 knapsack problems using a labeling algorithm. *Computers & Operations Research, 30,* 1865–1886.

Chen, A., Yang, G., & Wu, Z. (2006). Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. *Journal of Zhejiang University Science A, 7*(4), 607–614.

Chifu, V., Pop, C. B., Salomie, I., Suia, D. S., & Niculici, A. N. (2012). Optimizing the semantic web service composition process using cuckoo search. In F. M. T. Brazier, et al. (Eds.), *Intelligent distributed computing* (pp. 93–102). Berlin: Springer.

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

Comtois, C., Slack, B., & Rodrigue, J. P. (2013). *The geography of transport systems* (3rd ed.). London: Routledge. ISBN 978-0-415-82254-1.

Cordeau, J. F., Gendreau, M., Hertz, A., Laporte, G., & Sormany, J. S. (2005). New heuristics for the vehicle routing problem. In A. Langevin & D. Riopel (Eds.), *Logistics systems: Design and optimization*. Boston: Springer.

Côté, J., Potvin, J., & Gendreau, M. (2020). The vehicle routing problem with stochastic two-dimensional items. *Transportation Science, 54,* 299–564.

Du, L., & He, R. (2012). Combining nearest neighbor search with Tabu search for large-scale vehicle routing problem. *Physics Procedia, 25,* 1536–1546.

El Khoury, J., Akle, B., Katicha, S., Ghaddar, A., & Daou, M. (2014). A microscale evaluation of pavement roughness effects for asset management. *International Journal of Pavement Engineering, 15*(4), 323–333.

Fink, M., Desaulniers, G., Frey, M., Kiermaier, F., Kolisch, R., & Soumis, F. (2019). Column generation for vehicle routing problems with multiple synchronization constraints. *European Journal Operation Research, 272,* 699–711.

Gandomi, A., Yang, X. S., & Alavi, A. (2013). Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Engineering with Computers, 29*(1), 17–35.

Garrido, P. & Castro, C. (2009). Stable solving of cvrps using hyper-heuristics. In *Proceedings of the 11th annual conference on genetic and evolutionary computation, GECCO'09* (pp. 255–262), New York, NY, USA, ACM.

Gomez, A. & Salhi, S. (2014). Solving capacitated vehicle routing problem by artificial bee colony algorithm. In *2014 IEEE symposium on computational intelligence in production and logistics systems (CIPLS)*.

Gounaris, C., Repoussis, P., Tarantilis, C., Wiesemann, W., & Floudas, C. (2014). An adaptive memory programming framework for the robust capacitated vehicle routing problem. *Transportation Science, 50*(4), 141223041352002. https://doi.org/10.1287/trsc.2014.0559.

Haraty, R. A., Mansour, N., & Zeitunlian, H. (2018). Metaheuristic algorithm for state-based software testing. *Applied Artificial Intelligence, 32*(2), 197–213.

IBC. (2019). *International Business Corporation.* Retrieved May 2019 from http://ibcleb.com/.

Jin, J., Crainic, T. G., & Løkketangen, A. (2012). A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems. *European Journal of Operational Research, 222*(3), 441–451.

Jin, J., Crainic, T. G., & Løkketangen, A. (2014). A cooperative parallel metaheuristic for the capacitated vehicle routing problem. *Computers & Operations Research, 44,* 33–41.

Khoury, J., Amine, K., & Abi Saad, R. (2019). An initial investigation of the effects of a fully automated vehicle fleet on geometric design. *Journal of Advanced Transportation*. https://doi.org/10.1155/2019/6126408.

Kim, B.-I., & Son, S.-J. (2012). A probability matrix based particle swarm optimization for the capacitated vehicle routing problem. *Journal of Intelligent Manufacturing, 23*(4), 1119–1126.

Layeb, A. (2011). A novel quantum inspired cuckoo search for knapsack problems. *International Journal of Bio-Inspired Computations, 3*(5), 297–305.

Lenstra, J. K., & Rinnooy Kan, A. H. G. (1981). Complexity of vehicle routing and scheduling problems. *Networks, 11,* 221–227.

Marinaki, M., Marinakis, Y., & Zopounidis, C. (2010). Honey bees mating optimization algorithm for financial classification problems. *Applied Soft Computing, 10*(3), 806–812.

Marshall, R., Johnston, M., & Zhang, M. (2014). Hyper-heuristics, grammatical evolution and the capacitated vehicle routing problem. In *Proceedings of the companion publication of the 2014 annual conference on genetic and evolutionary computation, GECCO Comp'14* (pp. 71–72), New York, NY, USA: ACM.

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

Niu, Y., Wang, S., He, J., & Xiao, J. (2015). A novel membrane algorithm for capacitated vehicle routing problem. *Soft Computing, 19*(2), 471–482.

Ouaarab, A., Ahiod, B., & Yang, X. S. (2014). Discrete cuckoo search algorithm for the travelling salesman problem. *Neural Computational Applications, 24,* 1659–1669.

Payne, R. B., & Sorensen, M. D. (2005). *The cuckoos*. Oxford: Oxford University Press.

Shour, A., Danash, K., & Tarhini, A. (2015). Modified clarke wright algorithms for solving the realistic vehicle routing problem. In *2015 3rd international conference on technological advances in electrical, electronics and computer engineering, TAEECE 2015 7113606* (pp. 89–93).

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

Tarhini, A., Makki, J., & Chamsiddine, M. (2014). Scatter search algorithm for the cross-dock door assignment problem. In *Proceedings of the mediterranean electrotechnical conference—MELECON 6820575* (pp. 444–450).

Tarhini, A., Yunis, M., & Chamseddine, M. (2016). Natural optimization algorithms for the cross-dock door assignment problem. *IEEE Transactions on Intelligent Transportation Systems, 17*(8), 2324–2333.

Vazquez, R. A. (2011). Training spiking neural models using cuckoo search algorithm. In *Evolutionary computation (CEC), IEEE congress*.

Yang, X.-S., Deb, S. (2009). Cuckoo search via Lévy flights. In *Proceedings of the world congress on nature and biologically inspired computing (NaBIC), Coimbatore, India, 9–11 December 2009* (pp. 210–214).

Yang, X. S., & Deb, S. (2010). Engineering optimisation by cuckoo search. *International Journal of Mathematical Modeling and Numerical Optimization, 1,* 330–343.

Yang, X. S. & Deb, S., (2013). Multiobjective cuckoo search for design optimization. *Computers & Operations Research, 40*(6), 1616–1624.

Yang, X., Deb, S., Karamanoglu, M., & He, X., (2012). Cuckoo search for business optimization applications. In *National conference on computing and communication systems, Durgapur* (pp. 1–5).

Yildiz, A. R. (2013). Cuckoo search algorithm for the selection of optimal machining parameters in milling operations. *International Journal of Advanced Manufacturing and Technology, 64,* 55–61.

Zainudin, S., Kerwad, M., & Othman, Z. A. (2015). A water flow-like algorithm for capacitated vehicle routing problem. *Journal of Theoretical and Applied Information Technology, 77*(1), 125–135.

Zhen, L. (2016). Modeling of yard congestion and optimization of yard template in container ports. *Transportation Research Part B, 90*, 80–104.