CrossMark

ORIGINAL RESEARCH

Cuckoo search via Lévy flights for the capacitated vehicle routing problem

Jon Henly Santillan[1] [iD] · Samantha Tapucar[1] · Cinmayii Manliguez[1] ·
Vicente Calag[1]

Abstract For this paper, we explored the implementation of the cuckoo search algorithm applied to the capacitated vehicle routing problem. The cuckoo search algorithm was implemented with Lévy flights with the 2-opt and double-bridge operations, and with 500 iterations for each run. The algorithm was tested on the problem instances from the Augerat benchmark dataset. The algorithm did not perform well on the problem instances, save for a select few on which the algorithm achieved the close to near-optimal result and one on which the algorithm achieved the optimal result. Increasing the number of iterations for each run of the algorithm on the two large-scale problem instances led to obtaining solutions closer to the optimal solution compared to the ones obtained with fewer number iterations. This gives an idea that the larger the problem instance becomes, the slower the algorithm converges to the optimal solution. Several other factors may also have contributed to the overall performance of the algorithm. Regardless of its performance, the algorithm was able to obtain routes that satisfied the constraints of the capacitated vehicle routing problem. The potential of the cuckoo search algorithm in solving combinatorial problems is demonstrated in this study in which the performance of the algorithm on routing problems was explored.

Keywords Capacitated vehicle routing problem ·
Combinatorial optimization · Cuckoo search · Lévy flights

✉ Jon Henly Santillan
josantillan@up.edu.ph

1 Department of Mathematics, Physics, and Computer Science, University of the Philippines Mindanao, Mintal, 8022 Davao, Philippines

Introduction

The capacitated vehicle routing problem (CVRP) is a vehicle routing problem (VRP) in which the constraints lie mainly in the vehicle capacities and the maximum distance that each vehicle can travel (Zhang et al. 2008). The defined capacity of the vehicles makes CVRP the problem that it is, which arguably holds the closest similarity to the constraints that real life applications are under. The VRP does not take into consideration the capacity of the vehicles which play a huge role in the decision making when it comes to distribution.

CVRP is considered as a fundamental problem in the field of combinatorial optimization specifically in transportation and distribution logistics (Kara et al. 2007). It embodies a necessary basis of logistics planning (Chandran and Raghavan 2008) and evidently plays a critical part in wide-ranging practical applications. A solution to this particular routing problem will greatly benefit businesses and industries that are involved in transportation and distribution.

The cuckoo search (CS) is a metaheuristic algorithm proposed by Yang and Deb (2009) based on the obligate brood parasitic behavior of some cuckoo species. This algorithm is notably enhanced when combined with the Lévy flights (LF) rather than simple isotropic random walk method (Yang 2014). The LF is a behavior described by Reynolds and Frye (2007) as a way most animals and insects explore their landscape—by using a series of straight flight paths punctuated by a sudden 90° turn. The potential of the CS algorithm is recorded in various literatures (Vázquez 2011; Yang et al. 2012; Gandomi et al. 2013; Kaveh and Bakhshpoori 2013; Yildiz 2013).

For this paper, we explored the implementation of the CS algorithm with LF for solving the CVRP. We tested the

🦋 ② Springer

algorithm on the problem instances from Augerat et al. (1995) and recorded its performance in terms of solution quality, with observation of the running times. The goal of this study was to add to the literature on the performance of the CS algorithm as a whole and on the performance of the CS algorithm as a solution for the CVRP. Additionally, it leads to other researches of the same nature in the future, as discovering an optimal solution to a routing problem such as CVRP is vital in real-life applications that involve transportation and distribution—one of the core points in satisfying demands of a consumer base.

The succeeding texts in this paper are organized as follows: the CVRP is formally presented and the approaches to solving it are reviewed in Sect. 2, the CS algorithm is formally presented in Sect. 3, the implementation of the CS algorithm to solve the CVRP is presented in Sect. 4, the experimental results are presented in Sect. 5, and the conclusion is made with recommendations in Sect. 6.

Problem

Capacitated vehicle routing problem

The CVRP is a situation in which a number of customers with individual demands are satisfied by a number of homogenous vehicles, each with a given capacity, from a central depot. The objective of the problem is to determine the set of optimal routes traveled by a number of identical vehicles with minimal travel costs (Toth and Vigo 2002). A route is feasible when it begins and ends at the central depot, with each customer serviced exactly once, and the total demand on any route does not exceed the vehicle's capacity (Christiansen and Lysgaard 2007).

Formally, in the CVRP, a nonnegative demand $q_{u(i,k)}$—where u represents the route, i the customer, and k the vehicle—of a single commodity is to be delivered to n customers from a central depot using K independent delivery vehicles of identical capacity C (Ralphs et al. 2003; Kumar et al. 2014). Further, the goal is to complete the delivery with minimal distance D and least total cost, with $d_{u(i,k),u(i+1,k)}$ denoting the distance traveled for vehicle k from customer i to customer $i + 1$, d_k the total distance traveled by a vehicle k, and N_k the number of customers visited by vehicle k. Distances between two customers are calculated using the Euclidean distance formula. Additionally, each route must begin and end at the depot, each customer is part of exactly one route, and the total demand of each route does not exceed vehicle capacity C.

Kumar et al. (2014) translate the description to a mathematical representation:

$$\text{Min } D = \sum_{k=1}^{K} d_k, \tag{1}$$

where $d_k = \sum_{i=0}^{N_k} d_{u(i,k),u(i+1,k)},$ where $d_{u(i,k),u(i+1,k)}$ is calculated using the Euclidean distance formula

$$\text{Min } K, \tag{2}$$

$$\sum_{i=1}^{N_k} q_{u(i,k)} \leq C \quad \forall k = 1, 2, \ldots, K. \tag{3}$$

According to Kumar et al. (2014), Eqs. (1), (2), and (3) are for the objective function for minimization of total distance traveled by all vehicles, the objective for minimization of total number of vehicles used, and the vehicle capacity constraint, respectively.

Approaches to solving CVRP

A solution to the CVRP consists of a collection of K routes with minimum travel cost for K identical vehicles each with capacity C, such that each route for each vehicle must begin and end at the depot, each customer is visited exactly by one route, and the sum of the demand of the customers visited by a route does not exceed the vehicle capacity C (Ralphs et al. 2003; Kumar et al. 2014). Since CVRP is a variation of the vehicle routing problem (VRP), which is an extension of the traveling salesman problem (TSP), the foundation of many exact approaches for CVRP is derived from the extensive and successful work done for the exact solution of TSP (Toth and Vigo 2014). CVRP has been extensively studied since the early 60s, and, in the past years, many new heuristic and exact approaches have been presented (Toth and Vigo 2002). However, despite the huge progress seen with respect to the first algorithms, such as the tree search method by Christofides and Eilon (1969), CVRP is still far from being satisfactorily solved (Toth and Vigo 2014).

Exact techniques for CVRP are applied only to small-scale problems and the qualities of constructive heuristics are often not satisfactory (Huang et al. 2008). CVRP problem instances that can be consistently solved by the most effective exact algorithms proposed so far can only accommodate up to 50 customers, while larger instances, which contain hundreds of customers, can only be solved in particular cases and can only be tackled with heuristic methods (Toth and Vigo 2002). Despite the lack of a conclusive approach, enormous improvements in the research community's ability to solve these problems are improved due to better algorithms and computational abilities (Chandran and Raghavan 2008).

A study by Fukusawa et al. (2006) applied the Robust Branch and Cut Price algorithm for solving CVRP. Fukusawa et al. (2006) used four problems instances—*P-n16-k8*, *P-n23-k8*, *P-n40-k5*, and *P-n101-k4*—from the study of Augerat et al. (1995). The algorithm was able to produce a near-optimal solution for the four problem instances.

Ren (2012) applied genetic algorithm and was able to achieve desirable results with high efficiency and fast convergence rate. Shin and Han (2011) implemented a centroid-based heuristic algorithm which was able to achieve better results in many of the problem instances under sets A, B, and P of the Augerat et al. (1995) benchmark dataset as compared to the Sweep heuristic algorithm.

Lin et al. (2009) applied a hybrid algorithm of simulated annealing and tabu search and tested the algorithm on 14 classical problems and 20 large-scale benchmark instances. The hybrid algorithm achieved best solutions for 8 out of the 14 classical problems and exhibited competitive performance with other algorithms.

Mazzeo and Loiseau (2004) developed an Ant Colony algorithm (ACO) based on a metaheuristic technique introduced by Colorni et al. (1991). The developed algorithm achieved results which show very good performance in problem instances with up to 50 nodes and promising performance in bigger problem instances.

Kumar et al. (2014) reviewed other works on CVRP which showed promising performances. Kumar et al. (2014) also introduced a genetic algorithm with new fitness assignment procedure and tested the algorithm on standard benchmark problems. The algorithm achieved results which suggest its high competitiveness with other algorithms and its effectiveness for multi-objective optimization of vehicle routing problems.

Desired algorithm

Cuckoo search

Yang and Deb's (2009) take on modeling the behavior implemented in the CS algorithm is as follows:

1. Each cuckoo lays one egg at a time and dumps its egg in a randomly chosen nest;
2. the best nests with high-quality eggs will carry over to the next generation; and
3. the number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $p_\alpha \in [0, 1]$. In this case, the host bird can either get rid of the egg, or simply abandon the nest and build a completely new nest.

In other words, there is always a probability that the cuckoo egg is discovered by the host. The effect of having an egg discovered and consequently being thrown away or abandoned is approximated by a fraction p_α of the n host nests and are replaced by new nests (with new random solutions).

These three items that model the behavior of cuckoos provide a selection process for the CS algorithm, mimicking a "survival of the fittest" characteristic because it ensures that the best eggs survive from generation to generation (Yang 2010).

Furthermore, the algorithm's strength comes from the utilization of the LF pattern—especially when the global random walk is carried out (Yang 2014). Based on the aforementioned modeled behavior, a method of generating the eggs is required and will be applied with the LF.

Essentially, there are three components in this algorithm: selection of the best, exploitation by local random walk, and exploration by randomization via LF globally (Yang and Deb 2010). To control the step size (the random pattern) of the LF in generating solutions, a user-specified coefficient α is defined. Yang and Deb (2009) stated that in most cases, we can define $\alpha = 1$. When a Lévy step is generated using a random number generator, it is first multiplied by α before it is used to generate a new egg. The cuckoo laying eggs corresponds to a new solution set and this is analogous to generating new solutions for a cuckoo *i*.

Furthermore, the CS algorithm has only two parameters to be adjusted (Yang and Deb 2009):

1. The LF step size coefficient α, which controls the scale of the flight by multiplication and should be related to the size of the solution space of the objective function; and
2. The fraction p_α of eggs to be discarded. This value dictates how much exploration the algorithm will execute. If increasing, chances of getting trapped in a local minima is decreased. If decreasing, chances of getting trapped in a local minima is increased. This can also be interpreted by understanding that once *n*, which is the number of host nests, is fixed, p_α essentially controls the elitism and the balance of the randomization and local search.

Yang and Deb (2009) discovered in their own validation and testing of the CS algorithm that the convergence rate, to some extent, is not affected by the parameter p_α, which means that there is no need to fine-tune this parameter for a specific problem. In their testing of their algorithm, where $n = 15, 16, \ldots, 50$ and $\alpha = 1$, they found that $n = 15$ and $p_\alpha = 0.25$ are sufficient for most optimization problems.

These two parameters represent only a small number of user-specified parameters compared to the other optimization algorithms of this type (Yang et al. 2013). It is found

that the less parameters to be manipulated in an algorithm, the more generic it will be. Complexities, such as parameters overtly affecting results, will be avoided. The algorithm will be able to perform its task without being constrained and affected by too many parameters.

Yang (2010) simplifies the general implementation of the CS algorithm by using the following simple representation: each cuckoo egg in a host nest represents a solution and each cuckoo can only lay one egg which represents one solution. Each egg then carries two pieces of information: its coordinates in the solution space and its fitness value. With this information, the new egg/solution is to be evaluated. If the new egg/solution is significantly better or has more potential, it is to replace the previous solution in the nest which is now inferior in comparison to the new one. A situation where the nests can contain more than one egg which would represent a set of solutions can be achieved by the algorithm.

Like other metaheuristic algorithms, the CS execution is dependent on a stopping criterion. In Bacanin's (2011) object oriented software implementation of a novel version of the CS algorithm, the CS algorithm has the stopping criterion maxGeneration = 500, thus having 500 cycles per run. Bacanin (2011) states that the results of his implementation based on the four utilized benchmarks for performance evaluation are of optimal value and for a reasonable threshold 10^{-15}, the results are perfect. This coincides with Yang and Deb's (2009) review of their algorithm, where they run the algorithm at least 100 times and where each run stops when the variations of the function values are less than the given tolerance $\varepsilon \leq 10^{-5}$.

Additionally, Yang and Deb's (2009) review of the CS algorithm includes comparison of the performance of the genetic algorithm (GA) and particle swarm optimization (PSO). While both have performances ranging from 77 to 100%, CS aces all ten standard optimization benchmarks, where each algorithm has been run at least 100 times so as to carry out meaningful statistical analysis. Yang and Deb (2009) added that the primary reasons for such performance of the CS algorithm are the fine balance of randomization and intensification and the fact that fewer parameters are to be fine-controlled, which makes it evidently better and efficient for multimodal objective functions. These results emphasize the potential of CS, with Yang (2010) himself stating that CS is potentially far more efficient than other algorithms of similar goal.

With the various research findings, CS has been found to be more generic and robust for many optimization problems in comparison with other metaheuristic algorithms. This is not to say that CS cannot be hybridized with other mentioned algorithms, which has been done by other researchers, namely Kundra and Sadawarti (2015), and can produce even better outcomes.

Lévy flights

The foraging path of any animal is effectively a random walk, as the next move is based on the current location and the transition probability to the next location (Melin et al. 2015). Such randomization can be carried out in three ways: uniform randomization, random walks, and heavy-tailed walks (Yang 2010). The LF is a foraging pattern under the heavy-tailed walks and is a flight strategy exhibited by many organisms like fruit flies or *Drosophila melanogaster*.

LF essentially provides a random walk whose random step length is drawn from a Lévy distribution:

$$\text{Levy} \sim u = t - \lambda, (1 < \lambda \leq 3), \tag{4}$$

which has an infinite variance with an infinite mean (Yang 2010). This randomization plays an important role in both exploration and exploitation in metaheuristic algorithms such as the CS (Kaveh and Bakhshpoori 2013).

The LF pattern can also be described by many relatively short steps (corresponding to the detection range of the searcher) that are separated by occasional longer jumps (Noah et al. 2013). Another description of the LF pattern is an intensified search around a solution, followed by big steps in the long run (Ouaarab et al. 2014). Consequently, this is what is called a Lévy-flight-style intermittent scale-free search pattern (Roy and Chaudhuri 2013). A scale-free search pattern means that regardless of the scale the searching pattern will not differ and will present the same fractal patterns regardless of the range over which they are viewed (Noah et al. 2013). In terms of searching in the solution space, small-scale searches occur locally while large-scale searches occur globally—leading to an automatic balance between exploration and refinement (Yang et al. 2013). This means that when LF is generating new solutions, the search will mostly stay around the best solution obtained so far, which speeds up the local search. However, to avoid from being trapped in a local optimum, or in other words be stuck with a solution which is only best in a small area of the solution space, some of the new solutions will also be generated by a far field randomization whose locations are decidedly far enough from the current best solution (Yang and Deb 2009). Thus, LF holds a crucial role in controlling the balance between intensification and diversification (Ouaarab et al. 2014). It is the exponential property of LF that gives it a scale invariant property (Roy and Chaudhuri 2013). Yang and Deb (2009) also state that in most optimization problems, the search for a new best solution is made more efficient by LF. An example of an LF pattern is shown in Fig. 1.

In the CS algorithm, the selection of the best by keeping the best nests or solutions is equivalent to some form of elitism commonly used in genetic algorithms (Yang and
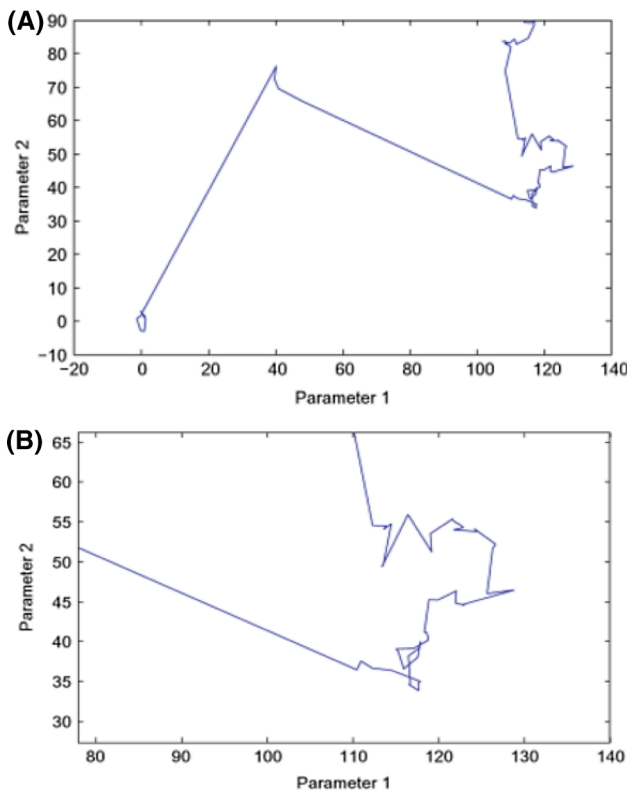
Fig. 1 a An example of a 100-step LF and **b** a zoomed-in section of the same LF (lifted from Yang et al. 2013)

Deb 2010). This elitism secures the best solution's position in the population by constantly passing it to the next generation with no risk of it being eliminated. The exploitation around the best solutions is performed by using a local random walk (Yang and Deb 2010):

$$x^{t+1} = x^t + \alpha \varepsilon_t, \tag{5}$$

x^{t+1} is a new solution generated using LF, x^t is the current best solution where the new solution is derived, α is the mentioned step size parameter, and if ε_t follows a Gaussian distribution, then this becomes a standard random walk. If ε_t is drawn from a Lévy distribution, the step of move is larger, and could potentially be more efficient (Yang and Deb 2010). There is possibility of the step being too large, and, therefore, there is risk that the move is too far away. Fortunately, the elitism mentioned keeps the exploitation moves within the neighborhood of the best solutions locally by keeping the best solutions of each iteration.

In Kaveh and Bakhshpoori's (2013) study, a more defined version of Eq. (5) is presented where instead of the ε_t representation for a random walk, S is a parameter that represents the length of random walk with LF according to Mantegna's algorithm:

$$x^{t+1} = x^t + \alpha \cdot S. \tag{6}$$

A random walk is a process which consists of taking a series of consecutive random steps. It can be expressed as

$$S_n = \sum_{i=1}^{n} X_i = X_1 + X_2 + \cdots + X_n = \sum_{i=1}^{n-1} X_i + X_n$$
$$= S_{n-1} + X_n, \tag{7}$$

where S_n represents the random walk with n random steps and X_i represents the ith random step with predefined length. The step size or length can vary according to distribution, and in this study's case it will follow the Lévy distribution.

In terms of implementation (Kaveh and Bakhshpoori 2013), the generation of numbers with LF consists of two steps: the choice of a random direction and the generation of steps, which obey the chosen Lévy distribution. The generation of steps can be quite tricky, but there are a few ways to achieve it. One of the most efficient and straightforward ways is to apply the Mantegna algorithm. In Mantegna's algorithm (Mantegna 1994), the step length S can be calculated by

$$S = \frac{u}{|v|^{1/\beta}}, \tag{8}$$

where β is a parameter between [1, 2] interval and considered to be 1.5; variables u and v are drawn from normal distribution as

$$u \sim N(0, \sigma_u^2), \quad v \sim N(0, \sigma_u^2), \tag{9}$$

where

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin\left(\frac{\pi\beta}{2}\right)}{\Gamma[(1+\beta)/2]\beta 2^{(\beta-1)/2}} \right\}^{1/\beta}, \quad \sigma_v = 1. \tag{10}$$

Despite its optimal searching pattern, however, LF does not come without its drawbacks; because LF also has a random nature, it cannot always be guaranteed (Yang 2010). However, LF is one of the most powerful features of CS (Yang et al. 2013). Accordingly, LF is core in making the CS algorithm reach its current potential.

LF is not an exclusive search pattern for CS. Another application of LF can be seen in Pavlyukevich's (2007) study on non-local search and simulated annealing. LF can also be seen on studies on human behavior foraging patterns, and even light can be related to LF (Yang 2010).

Fitness function

A fitness function measures the potential of each solution. The fitness function specific for CVRP gathered from Kumar et al. (2014) is presented as follows:

$$F(D)_i = \frac{(D)_{\max} - (D)_i}{(D)_{\max} - (D)_{\min}} \quad \forall i = 1, 2, \ldots, S, \tag{11}$$

where $F(D)_i$ is the fitness function value of distance traveled for ith solution in a population, $(D)_{\max}$ is the maximum distance traveled in a population, $(D)_{\min}$ is the minimum distance traveled in a population, $(D)_i$ is the distance traveled for the ith solution in a population, and S is the size of the population.

$F(D)$ is calculated for all solutions in a population. Since it is a minimization problem, a solution with high fitness function value is an optimal solution. Other forms of fitness can be defined in a similar way to the fitness function in genetic algorithms (Yang and Deb 2009).

Cuckoo search applications and other modifications

The CS algorithm has seen itself being applied to a variety of problems other than combinatorial optimization problems. These include structural optimization problems which are highly nonlinear and involve large numbers of design variables with complex constraints (Gandomi et al. 2013), business optimization applications (Yang et al. 2012), optimal machining parameters in milling operations (Yildiz 2013), and even optimum design of steel frames (Kaveh and Bakhshpoori 2013). The CS algorithm is also seen in the field of machine learning, where it is used in the study on the training of spiking neural networks (Vázquez 2011).

All mentioned studies have pointed out the significantly better performance of the CS algorithm due to fewer parameters compared to other algorithms. For that reason, the mentioned studies also acknowledge the overall potential of the CS algorithm. Furthermore, despite being relatively new in terms of years in the research world in comparison to other commonly used optimization algorithms (CS was introduced in 2009, PSO in 1995, GA introduced in the 1970's), the CS algorithm is no stranger to being modified or improved upon: a modified CS, with a new gradient-free optimization algorithm, has been developed and involves the addition of information exchange between the top eggs, or the best solutions (Walton et al. 2011). There is also the multi-objective CS algorithm for design optimization by Yang and Deb (2013) and even an improved CS algorithm for feed-forward neural network training by Valian et al. (2011).

Methodology

Problem instances

All the problem sets (A, B, and P) from the benchmark dataset of Augerat et al. (1995) were utilized for this study.

Further, only those problem instances with at least four vehicles were considered due to the restriction (see Sect. 4.3.4) in the use of an operation for the algorithm.

Problem representation

Toth and Vigo (2002) defined CVRP by the following graph theoretic problem. Let $G = (V, A)$ be a complete undirected graph where $V = \{0, \ldots, n\}$ is the vertex set with a corresponding Q demand set, and A is the arc set of undirected edges. Vertices $j = \{1, \ldots, n\}$ correspond to the customers where each vertex has a known nonnegative demand q_n to be delivered, whereas vertex $\{0\}$ corresponds to the depot with a demand $q_0 = 0$. Given a customer set $S \subseteq V$, let $d(S) = \sum q_{jj} \in S$ denote the total demand of a customer set. To illustrate V, a $2xm$ matrix is shown in Eq. (12) where each column represents the coordinates of each vertex or customer. Eq. (13) denotes Q, the demand set, where each element corresponds to one vertex or customer:

$$V = \begin{pmatrix} x_0 & x_0 & \ldots & x_m \\ y_0 & y_1 & \ldots & y_m \end{pmatrix}, \tag{12}$$

$$Q = (q_0, q_1, \ldots, q_n). \tag{13}$$

Furthermore, a nonnegative distance, d_{ij}, is associated with each arc $(i, j) \in A$ and represents the travel cost spent to go from vertex i to vertex j. Since $d_{ij} = d_{ji}$ for all $i, j \in V$, this makes for a symmetric CVRP (SCVRP). Therefore, there are no loop arcs (i, i). The arc set A is shown in Eq. (14) and is composed of the set of edges of the graph expressed as a_{ij}. The associated cost d_{ij} for all the arcs $(i, j) \in A$ is defined as the Euclidean distance between the two points corresponding to vertices i and j, where the distance is calculated using the Euclidean distance formula.

$$A = \begin{pmatrix} a_{10} & a_{21} & \ldots & a_{1j} \\ a_{20} & a_{31} & \ldots & a_{2j} \\ \vdots & \vdots & & \vdots \\ a_{i0} & a_{i1} & \ldots & a_{ij} \end{pmatrix} \tag{14}$$

The graph G includes the arcs connecting all vertex pairs, with the exception of loops. The problem is interpreted under the assumption that all the nodes presented, including the depot, are fully interconnected in a complete graph.

CS algorithm on CVRP

The flowchart (Fig. 2) and the parameters (Table 1) are presented below. For the parameters, the values assigned to n and p_α were based from Yang and Deb (2009) while the value for the maxGeneration was based from Bacanin (2011).

Fig. 2 CS algorithm flow based from the pseudocode presented by Yang and Deb (2009)
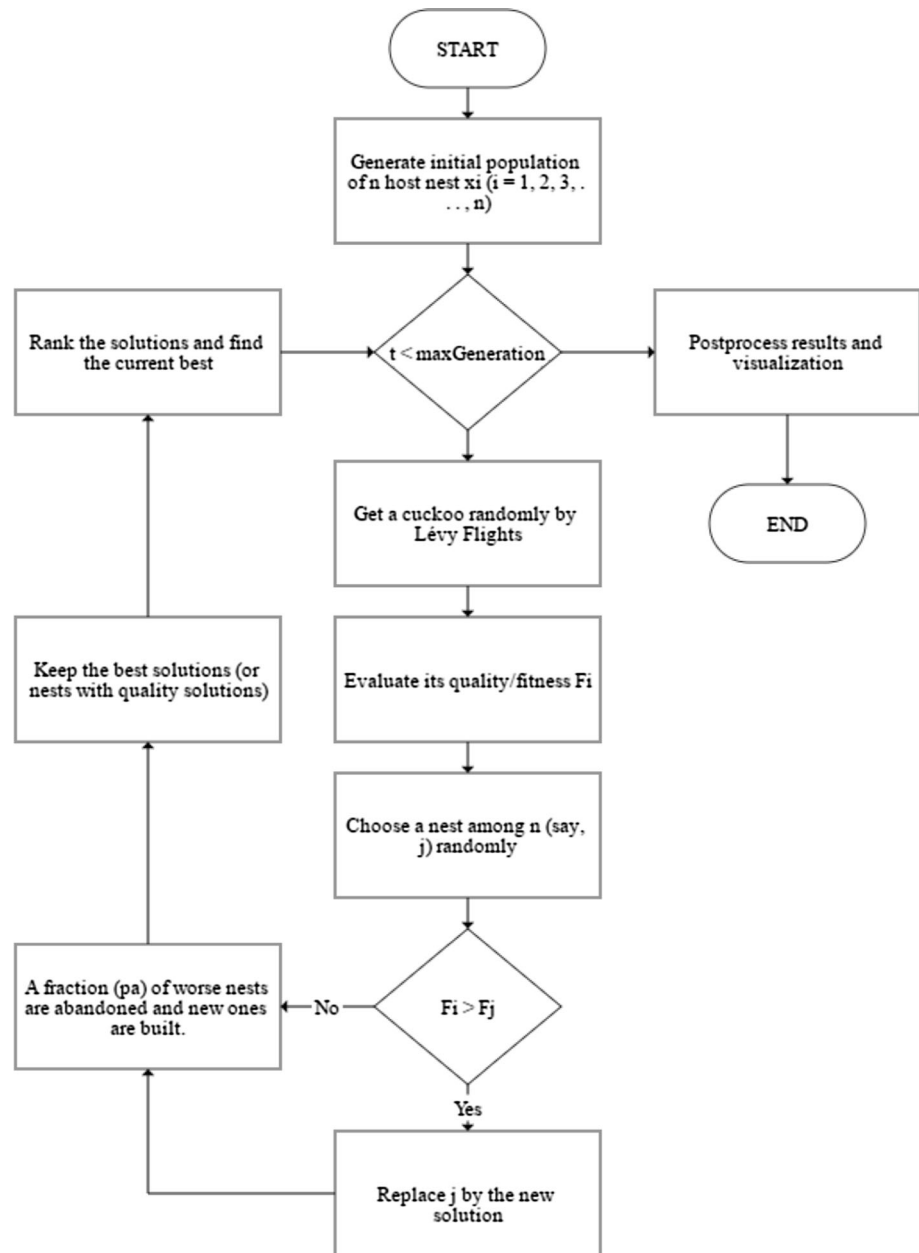
Table 1 CS algorithm parameters

Parameter name	Variable	Value
Initial population of host nests	n	15
Fraction of eggs to be discarded	p_α	0.25
Number of cycles/iterations	maxGeneration	500

Flow of the algorithm

The initial 15 solutions are generated using the information from the problem instance. A solution is then randomly chosen from the set of initial solutions for a comparison.

The second solution for comparison is also chosen from the set of initial solutions, but undergoes improvement determined by the LF value and the 2-opt and double-bridge operations. The LF value is determined using Eqs. (8), (9), and (10).

The LF value dictates the number of times that either 2-opt or double-bridge operation is applied. The algorithm then proceeds to either apply the operation for the remainder of the iteration or proceed to the comparison of the first randomly chosen solution and the second LF improved solution. During the comparison, the better solution is kept. If the better solution happens to be the LF solution, it will take the place of the randomly chosen

$$S = [1,5][7,15,16][2,3][10,14][4,11][13][6,12][8,9]$$

Fig. 3 Sample solution representation of a problem instance with 16 customers and 8 vehicles

solution. The fitness of all the solutions is recalculated, the solutions are ranked, and the three (based from the algorithm parameters) worst nests or solutions are removed and are replaced by newly generated solutions. After maintaining the number of solutions by generating new solutions, the fitness values of the solutions are again calculated and ranked.

The whole process constitutes a single iteration. After completing 500 iterations, the best solution is recorded to mark the end of a single run.

Generation of the initial solutions

A randomization pattern is applied to generate an initial solution for the population. A vehicle is randomly chosen, with the constraint that it should be able to cater to the currently unassigned customer with the least demand. If not, the program loops until a vehicle with adequate capacity is selected.

Next, an unassigned customer is randomly chosen and is added on the vehicle's route. If the customer–vehicle pairing does not meet the problem constraints, the program moves on to the next possible pairing. Otherwise, the function removes the customer from the pool of unassigned customers and adds it to the vehicle route, updates the vehicle capacity, and moves on to the rest of the unassigned customers and vehicles.

A sample solution S representation for the CVRP by the algorithm implementation is shown in Fig. 3. This entire procedure is repeated n (number of host nests) times.

When the initial solutions are obtained, the solutions' fitness values are calculated and are consequently ranked.

2-opt operation

2-opt is a popular simple local search operation (Chang 2015) which was first introduced by Croes (1958) for a single objective TSP. Its characteristics include the following: it is applicable to both symmetric and asymmetric problems with random elements; it does not use subjective decisions, so it can be completely mechanized; it is appreciably faster than any other method proposed; and it can be terminated at any point where the solution obtained so far is deemed sufficiently accurate.

In the TSP, 2-opt improves a random initial tour by exchanging two of the edges in the tour with two other possible edges. For example, the operation will select two edges (u_1, u_2) and (v_1, v_2) from the tour—where u_1, u_2, v_1, and v_2 are distinct and appear in this order in the tour—and will replace these edges with the edges (u_1, v_1) and (u_2, v_2), provided that this change will decrease the length of the tour (Englert et al. 2014). The operation is repeated until no more improvements can be made.

In the capacitated vehicle routing problem, an attempt to improve the current solution is done by swapping two customer positions—two individual routes are chosen where two customers are singled out and swapped. This consequently affects the solution quality.

Double-bridge operation

The double-bridge is a mutation operator for the Genetic Algorithm. It involves the exchange of four edges in a specific pattern (Handl et al. 2016). It allows large-scale changes in a tour to take place and it is a move that cannot be built from the composition of a local sequence of 2- and 3-changes (Martin et al. 1991).

An example of the specific pattern of the exchange of four edges in the traveling salesman problem is as follows: the edges (a,b), (c,d), (e,f), and (g,h) in the tour are replaced by the edges (a,f), (c,h), (e,b), and (g,d) (Ouaarab et al. 2014).

The double-bridge operation only differs to the 2-opt operation in that instead of swapping two customers, it swaps four customers. Due to the nature of this operation, each problem instance used must have at least four vehicles.

Experimental results

This section presents the results of the computational experiments carried out to determine the performance of the CS algorithm applied to the CVRP. The algorithm was coded using the Java programming language and was run mainly on a 1.9-GHz system with 4 Gb of RAM.

Comparison between the known best solutions from Augerat et al. (1995) and the best solutions obtained by the CS algorithm applied in this study for each of the selected problem instances from the Augerat et al. (1995) benchmark dataset is shown in Table 2 (set A problem instances), Table 3 (set B problem instances), and Table 4 (set P problem instances).

Table 2 The known best solution and the obtained best solution for problem instances under set A of the Augerat et al. ([1995](#)) benchmark dataset

Problem instance	Known best solution[a]	Algorithm best solution[b]	Algorithm running time (ms)
A-n32-k5	784	1065.403823	892
A-n33-k5	661	914.807849	922
A-n33-k6	742	1005.327921	1250
A-n34-k5	778	1083.907859	1031
A-n36-k5	799	1092.474023	953
A-n37-k5	669	998.7175832	953
A-n37-k6	949	1279.847282	2706
A-n38-k5	730	1239.188658	1797
A-n39-k5	822	1308.271003	1484
A-n39-k6	831	1292.886224	1563
A-n44-k6	937	1417.152478	1547
A-n45-k6	944	1863.87832	39,850
A-n45-k7	1146	1576.905214	1141
A-n46-k7	914	1339.615828	1125
A-n48-k7	1073	1686.614912	1176
A-n53-k7	1010	1940.128845	2321
A-n54-k7	1167	1925.787397	2440
A-n55-k9	1073	1772.741712	2456
A-n60-k9	1354	2246.618778	1846
A-n61-k9	1034	2200.839104	154,085
A-n62-k8	1288	2372.534683	1302
A-n63-k9	1616	2897.303846	13,290
A-n63-k10	1314	2325.596352	2281
A-n64-k9	1401	2468.14981	2546
A-n65-k9	1174	2608.494599	16,661
A-n69-k9	1159	2433.930649	1993
A-n80-k10	1763	3336.213656	2235

[a] Solution computed using integer values as based from investigation

[b] Solution computed using floating-point values

The bold values in Tables [3](#) and [4](#) indicate the solutions obtained by the applied algorithm that match or are close to the known best solution from the literature. Out of these closest solutions, one (*P-n16-k8*) achieved the same set of routes as the one in the literature and would have achieved the same length as the one in the literature if not for the difference in how the values are computed.

Most of the solutions are far off as compared to the ones from the literature. Hence, it is reported that this study's implementation of the CS algorithm is not effective. The results obtained for these problem instances are significantly larger than the best known solutions. The large difference could be attributed to several factors, such as the parameter settings (e.g. number of iterations), the interpretation and application of the Lévy Flights, the operations used (2-opt and double-bridge), or merely the nature of random walks and metaheuristics (Yang and Deb [2010](#)).

For consideration of the thought, we tested increasing the number of iterations for every run to check if it has any effect on the solution quality generated. Increasing the number of iterations to 10,000–30,000 makes it possible for the CS algorithm to obtain better solution lengths as compared to only 500 iterations for both the large problem instances *P-n40-k5* and *P-n101-k4*. At 30,000 iterations, the best solution length obtained for *P-n40-k5* was 581.15897, with a relative error of 0.2689 when compared to the best known solution. As for *P-n101-k5* at the same

Table 3 The known best solution and the obtained best solution for problem instances under set B of the Augerat et al. (1995) benchmark dataset

Problem Instance	Known best solution[a]	Algorithm best solution[b]	Algorithm running time (ms)
B-n31-k5	**672**	**746.0532697**	953
B-n34-k5	788	1007.244397	1922
B-n35-k5	955	1250.800843	1127
B-n38-k6	805	1057.749618	1016
B-n39-k5	549	927.6861256	1047
B-n41-k6	829	1308.713827	1375
B-n43-k6	742	1056.191069	1172
B-n44-k7	909	1317.475555	1724
B-n45-k5	751	1392.000384	2724
B-n45-k6	678	1219.118974	27,334
B-n50-k7	741	1376.797584	1577
B-n50-k8	1312	1694.608511	2419
B-n51-k7	1032	2090.954927	9277
B-n52-k7	747	1286.1832	1352
B-n56-k7	707	1357.439289	1678
B-n57-k7	1153	2373.36311	1,143,075
B-n57-k9	1598	2163.481756	1944
B-n63-k10	1496	2485.938697	2626
B-n64-k9	861	1977.610765	40,562
B-n66-k9	1316	2439.593684	4413
B-n67-k10	1032	1910.631285	2180
B-n68-k9	1272	2340.216782	2642
B-n78-k10	1221	2621.041132	2786

[a] Solution computed using integer values as based from investigation

[b] Solution computed using floating-point values

30,000 iterations, the best solution length obtained was 1044.22480, with a relative error of 0.5088 when compared to the best known solution. This gives us an idea that the convergence to the optimal solution of CS algorithm may be significantly slower for the larger problem instances. The other factors are left for future studies.

Conclusion

The CS algorithm's application in this study for the CVRP was not effective in achieving desirable results for the problem instances, most notably for the large ones, from the Augerat et al. (1995) benchmark dataset, except for a select few which are composed of small problem instances and whose values either match or are close to the ones from the literature. Such results may be attributed to a number of factors. With this thought, we explored on changing the

execution setup—the number of iterations in each run in particular—of the program. The results show that high numbers of iterations produce significantly better results than the set number of iterations. This gives an idea that the convergence of CS algorithm to the optimal solution may be significantly slower (or proportional to the size of the problem instance).

Future works include an in-depth look into possible modifications that can be done to the algorithm. Additionally, instead of applying 2-opt and double-bridge for a fixed number of applications, another option is to let the operators attempt to improve the solution using a tolerance number as control. Considering other operations for solution improvement is also an option. Minimal parameters, a strong search pattern, a form of elitism by removing worst nests, all of these contribute to the excellent performance of the CS algorithm in various studies. Even so, aside from Ouaarab et al.'s (2014) application of an improved and

Table 4 The known best solution and the obtained best solution for problem instances under set P of the Augerat et al. (1995) benchmark dataset

Problem instance	Known best solution[a]	Algorithm best solution[b]	Algorithm running time (ms)
P-n16-k8	**450**	**451.9470921**	3005
P-n22-k8	**603**	**632.716074**	41,343
P-n23-k8	**529**	**542.4735471**	458,471
P-n40-k5	458	691.0703193	2171
P-n45-k5	510	855.5071137	2110
P-n50-k7	554	909.229535	2906
P-n50-k8	631	1102.864308	2,422,874
P-n50-k10	696	1098.094843	8347
P-n51-k10	741	1304.993184	68,081
P-n55-k7	568	999.6864238	2610
P-n55-k8	588	1014.466054	1616
P-n55-k10	694	1088.883214	2515
P-n60-k10	744	1323.776593	3315
P-n60-k15	968	1473.970611	38,993
P-n65-k10	792	1419.109111	2416
P-n70-k10	827	1673.300549	20,364
P-n76-k4	593	1537.691449	2466
P-n76-k5	627	1601.158634	2551
P-n101-k4	681	1933.550639	2310

[a] Solution computed using integer values as based from investigation

[b] Solution computed using floating-point values

discrete version of the CS algorithm on the Traveling Salesman Problem, there is a lack of literature of the performance of the CS algorithm on routing problems; hence the existence of this study.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Augerat P, Belenguer J, Benavent E, Corberán A, Naddef D, Rinaldi G (1995) Computational results with a branch and cut node for the capacitated vehicle routing problem. Istituto di Analisi dei Sistemi ed Informatica. Consiglio Nazionale Delle Ricerche

Bacanin N (2011) An object-oriented software implementation of a novel cuckoo search algorithm. In: ECC'11 Proceedings of the 5th European Conference on European Computing Conference, Paris, France, pp 245–250

Chandran B, Raghavan S (2008) Modeling and solving the capacitated vehicle routing problem on trees. Veh Rout Probl Latest Adv New Chall, 1st edn. Springer, US, pp 239–261

Chang C (2015) A 2-opt with mutation operator to the traveling salesman problem. Int J Adv Eng Technol Comput Sci (IROSSS) 2(1):16–21

Christiansen C, Lysgaard J (2007) A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. Oper Res Lett 35(6):773–781

Christofides N, Eilon S (1969) An algorithm for the vehicle-dispatching problem. Oper Res Soc 20(3):309–318

Colorni A, Dorigo M, Maniezzo V (1991) Distributed optimization by ant colonies. In: First European Conference on Artificial Life, pp 134–142

Croes G (1958) A method for solving traveling-salesman problems. Oper Res 6(6):791–812

Englert M, Röglin H, Vöcking B (2014) Worst case and probabilistic analysis of the 2-opt algorithm for the TSP. Algorithmica 68(1):190–264

Fukusawa R, Longo H, Lysgaard J, de Aragão MP, Reis M, Uchoa E, Werneck R (2006) Robust branch-and-cut-price for the capacitated vehicle routing problem. Math Progr 106(3):491–511

Gandomi AH, Yang XS, Alavi AH (2013) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 29(1):17–35

Handl J, Hart E, Lewis P, López-Ibáñez M, Ochoa G, Paechter B (2016) Parallel problem solving from nature, 14th edn. Springer, US

Huang DS, Wunsch DC, Levine DS, Jo KH (2008) Advanced intelligent computing theories and applications with aspects of theoretical and methodological issues, 1st edn. Springer, US

Kara I, Kara B, Yetis MK (2007) Energy minimizing vehicle routing problem. Combinatorial optimization and applications, 1st edn. Springer, US, pp 62–71

Kaveh A, Bakhshpoori T (2013) Optimum design of steel frames using cuckoo search algorithm with Lévy flights. Struct Des Tall Spec Build 22(13):1023–1036

Kumar VS, Thansekhar MR, Saravanan R (2014) A new multi objective genetic algorithm: fitness aggregated genetic algorithm (FAGA) for vehicle routing problem. Adv Mater Res 984–985:1261–1268

Kundra H, Sadawarti H (2015) Hybrid algorithm of cuckoo search and particle swarm optimization for natural terrain feature extraction. Res J Inf Technol 7:58–69

Lin S, Lee Z, Ying K, Lee C (2009) Applying hybrid meta-heuristics for capacitated vehicle routing problem. Expert Syst Appl 36(2):1505–1512

Mantegna R (1994) Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes. Phys Rev E 49:4677–4683

Martin O, Otto S, Felten E (1991) Large-step markov chains for the traveling salesman problem. Complex Syst 5(3):299

Mazzeo S, Loiseau I (2004) An ant colony algorithm for the capacitated vehicle routing problem. Electron Notes Discret Math 18:181–186

Melin P, Castillo O, Kacprzyk J (2015) Design of intelligent systems based on fuzzy logic, neural networks and nature-inspired optimization, 1st edn. Springer, US

Noah SA, Abdullah SNHS, Arshad H, Bakar AA, Othman ZA, Omar K, Othman Z (2013) Soft computing applications and intelligent systems, 1st edn. Springer, Berlin

Ouaarab A, Ahiod B, Yang XS (2014) Discrete cuckoo search algorithm for the travelling salesman problem. Neural Comput Appl 24(7):1659

Pavlyukevich I (2007) Lévy flights, non-local search and simulated annealing. J Comput Phys 226:1830–1844

Ralphs T, Kopman L, Pulleyblank WR, Trotter LE (2003) On the capacitated vehicle routing problem. Math Progr 94(2):343–359

Ren C (2012) Applying genetic algorithm for capacitated vehicle routing problem. In: 2nd international conference on electronic & mechanical engineering and information technology (EMEIT), Paris, France, pp 519–522

Reynolds AM, Frye MA (2007) Free-flight odor tracking in Drosophila is consistent with an optimal intermittent scale-free search. PLoS One 2(4):e354

Roy S, Chaudhuri SS (2013) Cuckoo search algorithm using Lévy flights: a review. Int J Modern Educ Comput Sci 12:10–15

Shin K, Han S (2011) A centroid-based heuristic algorithm for the capacitated vehicle routing problem. Comput Inform 30(4):721–732

Toth P, Vigo D (2002) Models, relaxations, and exact approaches for the capacitated vehicle routing problem. Discret Appl Math 123(1–3):487–512

Toth P, Vigo D (2014) Vehicle routing: problems, methods, and applications, 2nd edn. Soc Ind Appl Math, US

Valian E, Mohanna S, Tavakoli S (2011) Improved cuckoo search algorithm for feedforward neural network training. Int J Artif Intell Appl 2(3):36–43

Vázquez RA (2011) Training spiking neural models using cuckoo search algorithm. In: Proceedings of the IEEE congress on evolutionary computation, New Orleans, LA, USA, pp 679–686

Walton S, Hassan O, Morgan K, Brown MR (2011) Modified cuckoo search: a new gradient free optimisation algorithm. Chaos Solitons Fract 44(9):710–718

Yang XS (2010) Nature-inspired metaheuristic algorithms, 1st edn. Luniver, Bristol

Yang XS (2014) Nature-inspired optimization algorithms, 1st edn. Elsevier, Netherlands

Yang XS, Deb S (2009) Cuckoo search via Lévy flights. In: Proceedings of world congress on nature & biologically Inspire Computing, India, pp 210–214

Yang XS, Deb S (2010) Engineering optimisation by cuckoo search. Int J Math Model Numer Optim 1(4):330–343

Yang XS, Deb S (2013) Multiobjective cuckoo search for design optimization. Comput Oper Res 40(6):1616–1624

Yang XS, Deb S, Karamanoglu M, Xingshi H (2012) Cuckoo search for business optimization applications. In: National Conference on Computing and Communication Systems, West Bengal, India, pp 1–5

Yang XS, Cui Z, Xiao R, Gandomi AH, Karamanoglu M (2013) Swarm intelligence and bio-inspired computation: theory and applications, 1st edn. Elsevier, Netherlands

Yildiz AR (2013) Cuckoo search algorithm for the selection of optimal machining parameters in milling operations. Int J Adv Manuf Technol 64(1):55–61

Zhang J, Zhao Y, Peng D, Wang W (2008) A hybrid quantum-inspired evolutionary algorithm for capacitated vehicle routing problem. Advanced intelligent computing theories and applications: with aspects of theoretical and methodological issues, 1st edn. Springer, US, pp 31–38