Discrete cuckoo search applied to capacitated arc routing problem

Badis Bensedira* and Abdesslam Layeb

Department of Computer Science and Its Applications,
Constantine 2 University,
Route Ain El Bey, Constantine 25017, Algeria
Email: bensedira-badis@hotmail.fr
Email: layeb.univ@gmail.com
*Corresponding author

Zineb Habbas

Department of Computer Science,
Lorraine University,
Ile du Saulcy, 57045, Metz Cedex, French
Email: zineb.habbas@univ-lorraine.fr

Abstract: Capacitated arc routing problem (CARP) is known as NP-hard combinatorial optimisation problem. Cuckoo search (CS) is a recent meta-heuristic algorithm inspired by breeding the behaviour of cuckoos. CS has proved to be very effective in solving continuous and discrete optimisation problems. In discrete CS for solving CARP (DCSARP), the solution is encoded as a sequence of integer numbers that are positioned according to the order between their components. This discretisation is the key for using CS for solving combinatorial optimisation problems. The experimental results showed that DCSARP can be better than some meta-heuristics in relation to the quality of solution and robustness.

Keywords: ARP; arc routing problem; combinatorial optimisation; CS; cuckoo search; meta-heuristics.

Reference to this paper should be made as follows: Bensedira, B., Layeb, A. and Habbas, Z. (2017) 'Discrete cuckoo search applied to arc routing problem', *Int. J. Metaheuristics*, Vol. 6, Nos. 1/2, pp.37–54.

Biographical notes: Badis Bensedira received his Master's degree from Constantine 2 University of M'sila, Algeria in 2012 and he is a PhD student from the year 2013 in Constantine 2 University of Constantine, Algeria. His current research interests include bio-inspired methods and their application to solve optimisation problems.

Abdesslem Layeb is an Associate Professor in the Department of Fundamental Computer Science and its Applications at the Constantine 2 University, Algeria. He is a member of MISC Laboratory. He received his PhD in Computer Science from the University Constantine 2, Algeria. He is interested in discrete mathematics and combinatorial optimisation, with an emphasis on algorithms for vehicle routing, packing and scheduling problems.

Zineb Habbas was awarded a degree from USTHB University of Algiers in 1979. She received her Master degree in Computer Science from the same university in 1984 and PhD in Computer Science from INPG in France in 1992. In 2003, she received her HDR which is the authorisation to supervise research. Since 1993, she has been teaching computer science at the University of Metz. Her research area includes parallel algorithmic models and constraint satisfaction problems.

1 Introduction

The aim of arc routing problem (ARP) is to find routes including a subset of edges that minimise a given cost. The ARP can be with or without constraints. One of the well-known variants of ARP is the CARP. The CARP consists of vehicles set, which must satisfy the clients' demands spread on edges. These can be a service, a supply or a collection measured with the capacity of each edge. CARP occurs frequently in many manufacturing and industrial services. Many of the applications are related to network roads, for instance mail delivery, waste and milk collection, water readings, electricity meters and others (Dror, 2000). Despite the importance of CARP in many real applications, the practical ones remain less widespread because it has many variants and each manufacturing or service industrial has its variant. Moreover, the CARP is known to be NP-hard (Golden and Wong, 1981), even in the case of single-vehicle called rural postman problem (RPP). Since exact methods are still limited to 190 edges (Bartolini, Cordeau and Laporte, 2013), heuristics and meta-heuristics are required for solving large instances. Heuristic methods include augment-merge (Golden and Wong, 1981), path scanning (Golden, DeArmon and Baker, 1983), construct-and-strike (Pearn, 1989), augment-insert (Pearn, 1991), Ulusoy's (1985) tour splitting algorithm and the route-first cluster-second heuristic (Stern and Dror, 1979).

Meta-heuristic algorithms have become the good compromise for many problems, which have big instances. Meta-heuristics are used for solving NP-hard problems because they have proved their potential and effectiveness in many problems. Furthermore, the meta-heuristics are general algorithms characterised by the flexibility and simplicity allowing their use to solve many problems by either extension or hybridisation. In the other hand, they can be adapted easily or combined with other methods to solve many real-world optimisation problems from the fields of operations research and engineering of artificial intelligence. The solution found by the meta-heuristic is not always the best, but it has a good quality and available in a reasonable time (Nesmachnow, 2014; Talbi, 2009).

Several approaches based on meta-heuristics were developed to solve CARP and its variants for finding a solution with effectiveness, quality and reasonable time. Beullens et al. (2003) developed a guided local search (GLS). Instead of evaluating neighbourhoods set of a given solution, GLS evaluates only a subset of edges based on the information marked for each edge in the previous search procedure. As a standard meta-heuristic, genetic algorithm was applied to an extended version of CARP (Lacomme, Prins and Ramdane-Cherif, 2004). In their work, a memetic algorithm (MA)

is proposed which combines genetic algorithm with local search. The chromosome is encoded as a sequence of required edges and used in Ulusoy's heuristic to get CARP solution. MA was tested on instances with up to 190 edges and 140 vertices. Based on vehicle routing problem (VRP) contributions, some studies were devoted to transform CARP to VRP (Longo, Aragão and Uchoa, 2006).

As real application of CARP, the salting route problem was solved using evolutionary algorithm (EA) (Handa et al., 2006). EA uses the same encoding of MA but any of heuristics is used to obtain the solution. Polacek et al. (2008) proposed a variable neighbourhood search (VNS). In this work, the neighbourhood operator is applied on sequences of required edges. VNS uses Ulusoy's algorithm to find all the trips. Brandão and Eglese (2008) proposed an adapted Tabu search algorithm (TSA) to solve CARP. TSA employs five heuristics to generate initial solutions. According to the used heuristics, two versions of TSA have been applied (TSA1 and TSA2). The first version uses only path scanning heuristic. However, the second version uses five heuristics. Another application of Tabu search with global repair operator is presented in Mei, Tang and Yao (2009). Contrary to TSA, the individual scheme is a sequence of vertices separated with the depot vertex. Furthermore, a solution can be infeasible. Another MA with extended neighbourhood search for CARP (MAENS) was proposed by the same authors in Tang, Mei and Yao (2009). They have employed a novel local search operator called merge-split. This later aims to modify a part of solution. At least, two routes are merged and the unordered list of required edges is used to reconstitute other routes.

Ant colony optimisation (ACO) is another popular meta-heuristic for tackling the CARP, ACO was proposed in Santos, Coutinho-Rodrigues and Current (2010). In this work, the initial population is generated with different approaches and the local search is based on 12 move operators. Usberti, Paulo and André (2013) have tackled the CARP using GRASP with path re-linking. This approach contains construction and local search phases. In the construction phase, the solutions are generated using a greedy randomised heuristic. The local search phase improves the initial solutions using four move operators. Recently, a hybrid meta-heuristic approach (HMA) developed by Chen, Hao and Glover (2016). HMA initialises the population with path scanning heuristic and improves the solution using six move operators before introducing the main phase. Compared with other MAs, HMA uses a novel crossover operator and a local search. For more details on real applications of CARP, we recommend the recent state of art in the book of Corberan and Laporte (2015).

The aim of this paper is the implementation of a new approach based on cuckoo search (CS) algorithm. The discretisation of the CS algorithm is required to solve the CARP. That is why we propose a discretisation of CS algorithm and the implementation of different steps according to this one. The use of Lévy flight function has a great role in the diversification and convergence to the optimal solution. The remainder of this paper is organised as follows: Section 2 presents the CARP formulation. An overview of the CS algorithm is presented in Section 3. In Section 4, the proposed approach is described. Experimental results are discussed in Section 5, and a conclusion is provided in Section 6.

2 Capacitated arc routing problem

2.1 Intuitive presentation

Like VRP, CARP is a widespread problem in combinatorial optimisation domain. In general, the CARP problems aim to determine the optimal set of rounds crossed by a fleet of vehicles to serve a set of customers spread on a given network (Golden and Wong, 1981). In CARP, each arc or edge has a demand and two different costs: a routing cost and a serve cost. Most formulations of CARP problems neglect the cost of service. The basic variant of CARP is the RPP. In a RPP problem, a single vehicle, assumed to have unlimited capacity and serves all the arcs or edges in the network. The RPP is a generalisation of travel salesman problem (TSP) where the customers are the edges or arcs in the network. CARP is considered as a generalisation of RPP where we have a set of vehicles with limited capacity. By default, the vehicles have the same capacity.

Informally, CARP solution is feasible, if and only if:

- Each tour starts and ends in the one deposit of the network.
- Each required edge (which has a strictly positive demand) is served on one vehicle.
- The sum of demands collected or granted by each vehicle should not exceed its own capacity.

2.2 Mathematical formulation

Mathematically, several formulations have been proposed to model the CARP problem. Let us consider a graph $G = (V, E)$, where V is a set of nodes and E is a set of edges, and K vehicles. Each edge $e = (v_i, v_j) \in E$ of G is presented with two arcs $e^1 = (v_i, v_j)$ and $e^2 = (v_j, v_i)$. e^1 and e^2 take the cost and request of e. Two sets of binary variables are used in this formulation: x^p_{ij} (respectively l^p_{ij}) equals 1 if the vehicle p passes by (respectively serves) the arc (v_i, v_j) and equals 0 otherwise. The problem consists in minimising the following quantity:

$$\text{Min} \sum_{p=1}^{K} \sum_{e \in E} \left(x_{ij}^{p} + x_{ji}^{p} \right) w_e \tag{1}$$

$$x^p \left(\delta^+ \left(v_i \right) \right) = x^p \left(\delta^- \left(v_i \right) \right) \forall\ v_i \in V, p = 1, \ldots \ldots, K \tag{2}$$

$$\sum_{p=1}^{K} \left(l_{ij}^{p} + l_{ji}^{p} \right) = 1 \forall e = \left\{ v_i, v_j \right\} \in E_r \tag{3}$$

$$x_{ij}^{p} \geq l_{ij}^{p} \qquad \forall\ e = \left\{ v_i, v_j \right\} \in E_r, p = 1, \ldots . K \tag{4}$$

$$\sum_{e=\{v_i,\, v_j\}\in E_r} d_e\left(l_{ij}^p + l_{ji}^p\right) \le Q \quad \forall\, p = 1, \ldots\ldots, K \tag{5}$$

$$x^p\left(\delta^+(S)\right) \ge x_{ij}^p + x_{ji}^p \quad \forall\, e = \{v_i,\, v_j\} \in E(S),\ p = 1, \ldots., K \forall\, S \subseteq V \setminus \{v_1\} \tag{6}$$

$$x_{ij}^p\ , x_{ji}^p\ , l_{ij}^p\ , l_{ji}^p \in \{0,1\} \quad \forall e = \{v_i, v_j\} \in E_r, p = 1, \ldots K \tag{7}$$

The objective function represents the sum of routing and service costs of all edges or arcs. The first constraint (Eq. 2) ensures that each node v_1 having an entry $\delta^+(vi)$ must have an output $\delta^-(vi)$ which allows vehicles to leave a node once they return to it. The second constraint ensures that each edge is served exactly once by one vehicle (Eq. 3). Equation (4) ensures that the vehicle can traverses the required edge more than once before he serves it. The capacity constraint is presented by inequality (Eq. 5) and the graph connectivity is ensured by constraint in (Eq. 6). Originally, this formulation was provided by Golden and Wong (1981) and then improved by Welz (1994). There are other formulations that improve this formulation in terms of variables number or the integration of other constraints concerning other variants of CARP (Dror, 2000).

The search for a solution needs many steps starting from the construction of the order of the required edges and the way of edges assignment to vehicles. The solution is not feasible where an edge cannot be assigned to a vehicle (the capacity constraint) or it is impossible for a vehicle to move from one edge to another. Most industrial problems formulated by CARP have large instances, which increases the required computation exponentially depending on the size of problem. Therefore, it is better to find other techniques to deal with these problems. In this paper, we are interested in applying CS algorithm to tackle this problem.

3 Cuckoo search

Until now, several complex problems stay without optimal solutions. Indeed, the computation complexity of the exact methods increases exponentially with the size of those problems. That is why meta-heuristic algorithms are considered to be the best way for solving many difficult problems (Gandomi, Yang and Alavi, 2011). The majority of these algorithms have been inspired by the natural systems such as physical, biology, etc. Yang and Deb (2009) developed a bio-inspired algorithm namely CS which is based on the behaviour of cuckoo. cuckoos use an aggressive strategy of reproduction that involves the female hacks nests of other birds to lay their eggs. Sometimes, the egg of cuckoo in the nest is discovered and the hacked birds discard or abandon the nest and start their own brood elsewhere. CS is based on the following three rules:

- Each cuckoo lays one egg at a time, and put it in a randomly chosen nest.
- The best nests with high quality of eggs (solutions) will carry over to the next generations.
- The number of available host nests is fixed, and a host can discover an alien egg with a probability $p_a \in [0, 1]$. In this case, the host bird can either throw the egg away or abandon the nest to build a completely new nest in a new location.

Concerning the first assumption, according to the problem, we can suppose that each cuckoo lays and put several eggs at a time. CS algorithm is presented in the pseudo-code as shown in Figure 1.

Figure 1 Cuckoo search

Objective function f(x), $x = (x_1, \ldots\ldots, x_d)^T$

Initial a population of N host nests $x_i (i = 1, 2 \ldots N)$;

While (*t*<*MaxGeneration*) or (stop criterion);

- Get a cuckoo (say *i*) randomly by Lévy flights;
- Evaluate its quality/fitness F_i;
- Choose a nest among n (say j) randomly;
- if (F_i>F_j),
  Replace *j* by the new solution;
  end
- Abandon a fraction (p_a) of worse nests
- Build new ones at new locations via Lévy flights;
- Keep the best solutions (or nests with quality solutions);
- Rank the solutions and find the current best;

End while

Cuckoo search algorithm is characterised by its simplicity because it uses few parameters. The original version of CS operates in continuous search space. In the standard CS algorithm, the new generation of cuckoos is based on Lévy flights (Eq. 8) named by the French mathematician Paul Lévy, representing a model of random walks characterised by step length, which obeys to the probability distribution. Lévy flight is used to model many transitions in the states of many phenomena physical, chemical, organic and natural in reality.

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \oplus \text{Levy}(s, \lambda) \tag{8}$$

$$\text{Levy}(s, \lambda) \sim s^{-\lambda}, (1 < \lambda \leq 3) \tag{9}$$

where $x_i^{(t+1)}$ and $x_i^{(t)}$ represent the solution *i* at time *t* and *t* + 1, respectively. α > 0 is the step size, which should be related to the scales of the problem of interests. Generally, α

takes the value of 1. The product $\oplus$ means entry-wise multiplications. This entry-wise product is similar to those used in PSO, but here the random walk via Lévy flights is more efficient in exploring the search space as its step length is much longer in the long run.

Compared to results of several meta-heuristic algorithms such as PSO and GA (Yang and Deb, 2009), CS algorithm was applied for many optimisation problems and has shown an encouraging efficiency. CS algorithm has contributed for solving a lot of continuous problems and gives effective results. Unfortunately, the original CS algorithm allows only the resolution of many problems where the solution representation is based on continuous coordinates. Recently, some studies are proposed to solve some discrete problems using CS algorithm, for example, the travelling salesman problem (TSP) (Ouaarab, Ahiod and Yang, 2014) and job shop scheduling problem (Ouaarab, Ahiod and Yang, 2015). The adaptation of CS algorithm for solving CARP problems is detailed in the next section.

4 Discrete cuckoo search for CARP

4.1 CARP solution representation

Cuckoo search algorithm was designed for continuous optimisation problems; therefore, it cannot be adopted directly to CARP problem. Basic CS solution is used to get the permutation of required edges. For that, we have used the smallest position value method (Tasgetiren et al., 2006) which sorts the indexes according to the ascending order of CS solution (Figure 2). The sorted CS solution becomes CARP solution with implicit paths between two consecutive required edges. Finally, the shortest path is computed with Dijkstra algorithm.

Figure 2 An example of SPV method

	1	2	3	4	5
Existing cuckoo solution	2.1248	-0.1235	2.3654	0.1245	-1.1547
Sequence of required edges	5	2	4	1	3
	-1.1547	-0.1235	0.1245	2.1248	2.3654

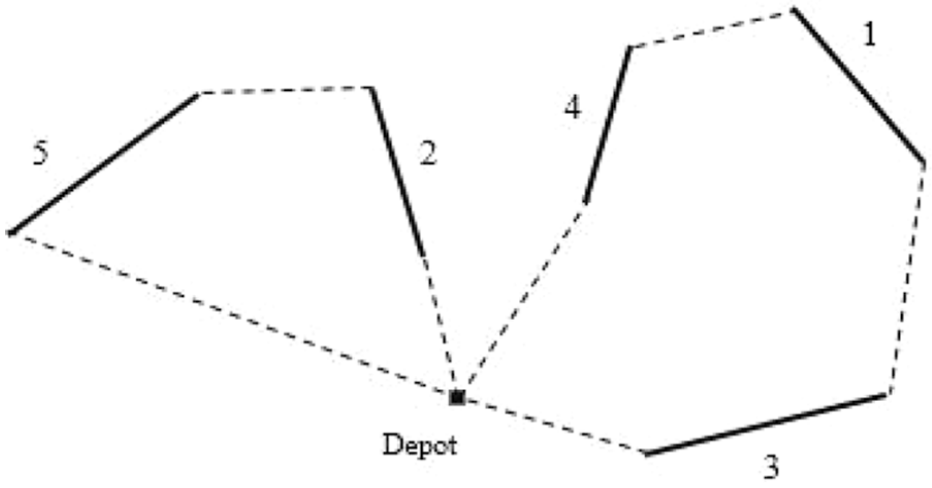
Two steps are used to construct the CARP solution. The first step assigns the required edges to vehicles using FIFO method (First In First Out). In the second step, the required edges and the depot vertex are connected with shorted paths. For the sake of clarity, Table 1 contains an instance of CARP.

Table 1 CARP instance

Required edges	*1*	*2*	*3*	*4*	*5*
Requests	2	4	3	4	3

Figure 3 shows an example of CARP solution based on the CS solution in Figure 2 and the parameters of CARP instance in Table 1. The capacity of vehicles is 10. The required edges are in boldface. Dashed lines indicate shortest deadheading paths. With R required edges, the solution can be computed in $O(2R)$. The passage on edges is established under the order in CS solution. Note that an infeasible solution can exist when FIFO method assigns only a sub-ensemble of required edges. In this case, we affect a penalty cost to this solution.

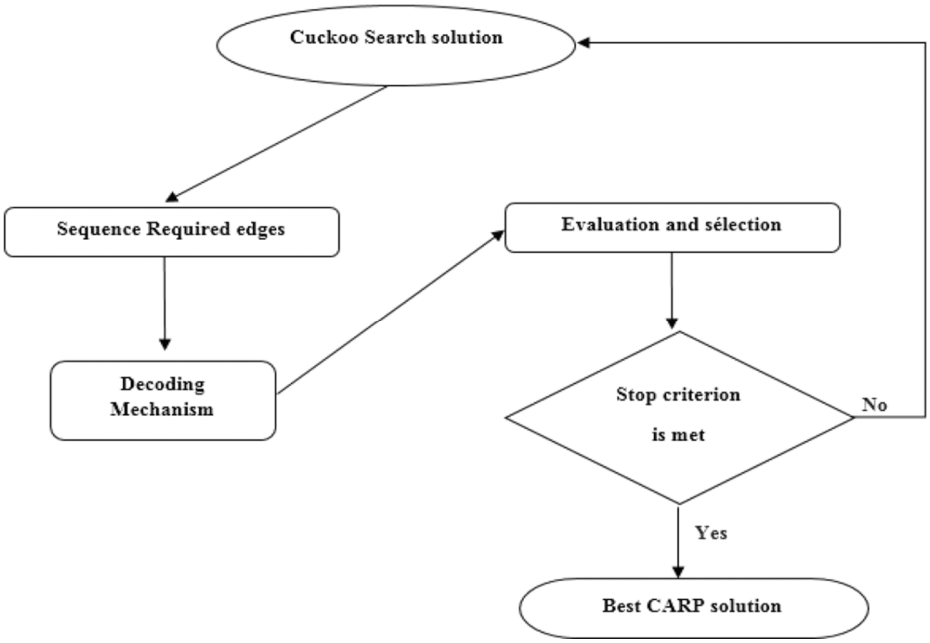
Figure 3 An example of CARP solution with 2 vehicles



4.2 Discrete cuckoo search algorithm

After each iteration of the discrete cuckoo search for CARP (DCSARP) algorithm, each cuckoo creates a solution. The solution represents a sequence of the required edges. To build a CARP solution, we add a decoding mechanism that uses Dijkstra algorithm to find the routes between each pair of edges. Figure 4 gives the flowchart of the proposed DCSARP. It should be noted that the proposed algorithm can be extended to deal with a large number of discrete combinatorial problems. The discrete cuckoo search algorithm is presented as follows:

Figure 4 Flowchart of the DCSARP algorithm



The first step in this algorithm is initialising the parameters. The main advantage of the DCSARP algorithm is that there are fewer parameters to be set compared to other population-based algorithms. There are essentially three main parameters to initialise; the population size N, the number of vehicles V and the required edges R. the size of nest (egg) is equal to R. In the second step, a swarm of N nests are created and initialised to generate some possible CARP solutions. Like any population-based algorithm, it is recommended to use some heuristics to get a diverse population containing both good and bad nests. Moreover, a distance matrix is created and initialised using costs vector D in the purpose of reducing the time convergence. The algorithm progresses through several generations according to the CS dynamics (Figure 5). During each iteration, a new cuckoo is built. The next step is the evaluation of the current cuckoo. For that, we apply a discretisation function to get the required edges sequence. This last is divided between vehicles using assignment function.

Figure 5 Discrete Cuckoo Search algorithm for CARP problem

Input: V, R: number of vehicles and required edges
C: capacity
Q: requests vector
D: Costs vector
N: number of nests

Output: The best CARP solution
Objective function f(x), $x = (x_1, \ldots\ldots, x_R)^T$
Initialize a population of N host nests $x_i(i = 1, 2 \ldots N)$;
While (t<*MaxGeneration*) or (stop criterion);

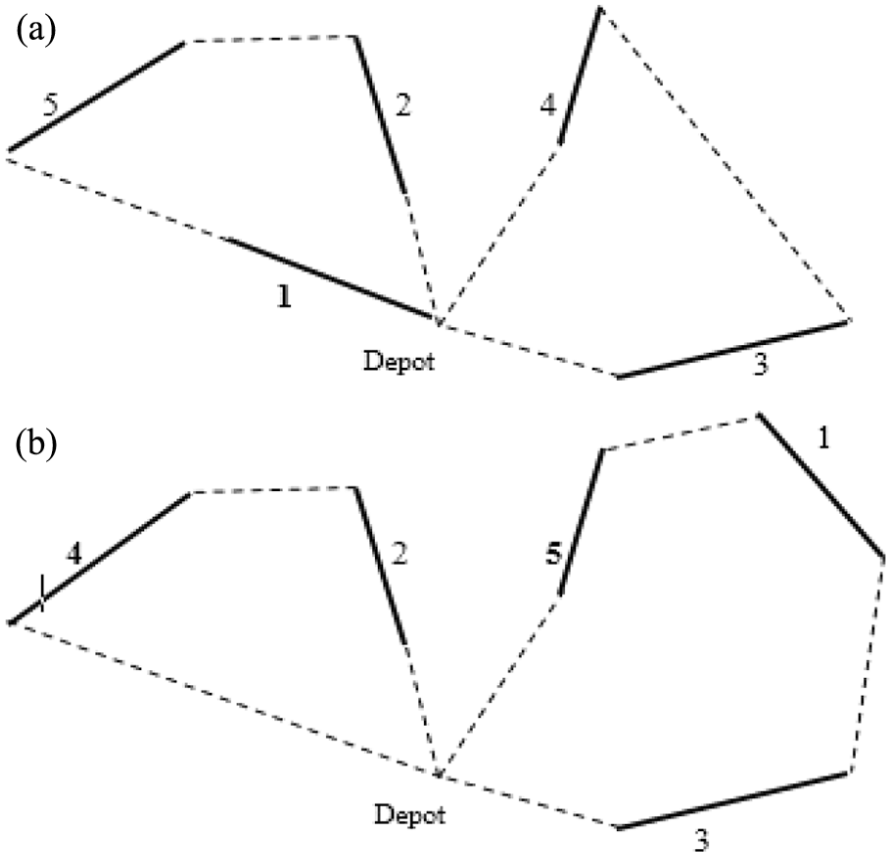
- Get a cuckoo (say *i*) randomly by Lévy flights;
- **Get its discrete representation**;
- **Expand the discrete representation to get CARP solution**;
- Evaluate its quality/fitness F_i (**Cost**);
- Choose a nest among n (say j) randomly;
- if (F_i>F_j),
  Replace the cuckoo j by the corresponding solution
  Replace *j* by the new CARP solution;
  End
- Abandon a fraction (p_a) of worse nests
- Build new ones at new locations via Lévy flights;
- **Keep the best CARP solutions (or nests with quality solutions);**
- Rank the solutions and find the current best;

End while

4.3 Moving in the search space

To maintain the exploration of search space, CS uses lévy flight function which models random walks. The use of levy flight in discrete search space increases the intensification and therefore finding the best solution is delicate. That is why DCSARP associates with lévy flight other moves. The first move is insertion and the second is swap. According to the value of lévy flight, the corresponded move is carried. In insertion move, a candidate edge (only required edges) is removed from its current position and inserted in a randomly position (except the current position). The swap move is similar, two required edges are selected and a permutation is made. Figure 6a and b illustrates the lévy flight moving applied on example of Figure 3 using insertion and swap move, respectively.

Figure 6 Lévy flight moving



5 Experimental results and discussion

The proposed DCSARP algorithm is implemented in Java using NetBeans IDE 8.0.2 under the 64 bits Seven Operating System. Experiments are concluded on laptop with configuration of Intel(R) CoreTM I3 2. 20 GHz, and 4 GB of RAM. We tested the performance of DCSARP on 63 instances of three standard test data sets for CARP. All the edges are required in these networks and the capacities of vehicles are equal:

- ***kshs***: This data set was introduced in Kiuchi et al. (1995). It includes six undirected networks ranging from 6 to 10 nodes and 15 edges.
- ***gdb***: It was proposed by Golden, DeArmon and Baker (1983) to evaluate a capacitated version of Chinese Postman Problem. This set contains 23 undirected networks ranging from 7 to 27 nodes and from 11 to 55 edges.
- ***val***: Benavent et al. (1992) proposed data set which includes 34 undirected networks ranging from 24 to 50 nodes and from 34 to 97 edges.

The data sets are downloaded from *http://logistik.bwl.uni−mainz.de/benchmarks.php*. Table 2 summarises the parameter values of DCSARP algorithm which give the best results about both of *kshs*, *gdb* and *val* instances. We set different parameters based on parametric studies of p_a, individuals and iterations number. In combinatorial problems, the lower and upper bounds have no influence on the results improvement. In the first (respectively second) study, all the parameters have been fixed except individual's number (respectively iterations) which is variable in each run. The third study is based on p_a. The p_a has been setted at each run. In addition, the parameter p_a is randomly setted at each iteration.

Table 2 Parameters setting

Parameter	*Value*
Population size	30
Solution size	Number of required edges
Number of iterations	200000
Bad solution probability (P_a)	0.2
Lower bound (LB)	0
Upper bound (UB)	1

To ensure the DCSARP performance and to prove its abilities in solving ARPs, this section compares the proposed algorithm to several CARP solver algorithms. The results reported in Tables 3–5. All the tables list the name of instance and the corresponding characteristics as number of vertices |V|, required edges |E|, lower bound (Best known), and the best known given by our contribution and other works. In addition, the results are highlighted in bold for the instance on which DCSARP achieved the best solution among the compared methods. Following the recent practice in literature, we compare our DCSARP with four population-based algorithms (MAENS; Tang, Mei and Yao, 2009, ACO; Santos, Coutinho-Rodrigues and Current, 2010, GRASP; Usberti, Paulo and André, 2013 and HMA; Chen, Hao and Glover, 2016) and one local search algorithm (TSA; Brandão and Eglese, 2008):

- ***GA***: reports the best results on *gdb*, *val*, *egl*.
- ***MAENS***: reports the best results on *gdb, val, egl, C-F* obtained from 30 runs.
- ***ACO***: reports the best results on *gdb, val, egl, C-F* obtained from 10 runs.
- ***GRASP***: reports the best results on *gdb, val, egl* obtained from 15 runs.
- ***HMA***: reports the best results on *gdb, val, egl, C-F* obtained from 30 runs. Two versions proposed are HMA^1 and HMA^2. We chose the second version for comparison which gave the best results.
- ***TSA***: reports the best results on *gdb, val, egl, C-G* obtained from one run. Two versions proposed are TSA^1 and TSA^2. We chose the second version for comparison which gave the best results.

Table 3 Results for *kshs* instances

File	*|V|*	*|E|*	*Best*	*GLS*	*BCPA*	*DCSARP*
Kshs1	4	15	14661	14661	14661	14661
Kshs2	4	15	9863	9863	9863	9863
Kshs3	4	15	9320	9320	9320	9320
Kshs4	4	15	11498	11498	11498	11498
Kshs5	3	15	10957	10957	10957	10957
Kshs6	3	15	10197	10197	10197	10197

Table 4 Results for *gdb* instances

File	*|V|*	*|E|*	*Best*	*GA*	*TSA_2*	*GRASP*	*ACO*	*MAENS*	*HMA_2*	*DCSARP*
gdb1	12	22	316	316	316	316	316	316	316	**316**
gdb2	12	26	339	339	339	339	339	339	339	**339**
gdb3	12	22	275	275	275	275	275	275	275	**275**
gdb4	11	19	287	287	287	287	287	287	287	**287**
gdb5	13	26	377	377	377	377	377	377	377	**377**
gdb6	12	22	298	298	298	298	298	298	298	**298**
gdb7	12	22	325	325	325	325	325	325	325	**325**
gdb8	27	46	348	348	352	348	348	348	348	350
gdb9	27	51	303	303	317	303	303	303	303	308
gdb10	12	25	275	275	275	275	275	275	275	**275**
gdb11	22	45	395	395	395	395	395	395	395	**395**
gdb12	13	23	456	458	458	458	458	458	458	**458**
gdb13	10	28	536	538	540	536	536	536	536	544
gdb14	7	21	100	100	100	100	100	100	100	**100**
gdb15	7	21	58	58	58	58	58	58	58	**58**
gdb16	8	28	127	127	127	127	127	127	127	**127**
gdb17	8	28	91	91	91	91	91	91	91	**91**
gdb18	9	36	164	164	164	164	164	164	164	**164**
gdb19	8	11	55	55	55	55	55	55	55	**55**
gdb20	11	22	121	121	121	121	121	121	121	**121**
gdb21	11	23	156	156	156	156	156	156	156	**156**
gdb22	11	44	200	200	200	200	200	200	200	**200**
gdb23	11	55	233	235	235	233	235	233	235	**233**

Table 5 Results for *val* instances

File	*|V|*	*|E|*	*Best*	*GA*	*TSA_2*	*GRASP*	*ACO*	*MAENS*	*HMA_2*	*DCSARP*
1A	24	39	173	173	173	173	173	173	173	**173**
1B	24	39	173	173	173	173	173	173	173	**173**
1C	24	39	245	245	245	245	245	245	245	249
2A	24	34	227	227	227	227	227	227	227	**227**
2B	24	34	259	259	260	259	259	259	259	**259**
2C	24	34	457	457	457	457	457	457	457	**457**
3A	24	35	81	81	81	81	81	81	81	**81**
3B	24	35	87	87	87	87	87	87	87	**87**
3C	24	35	138	138	138	138	138	138	138	**138**
4A	41	69	400	400	400	400	400	400	400	**400**
4B	41	69	412	412	412	412	412	412	412	414
4C	41	69	428	430	430	430	430	430	428	448
4D	41	69	530	530	546	530	530	530	530	555
5A	34	65	423	423	423	423	423	423	423	428
5B	34	65	446	446	446	446	446	446	446	452
5C	34	65	474	474	474	474	474	474	474	476
5D	34	65	575	581	583	581	577	577	575	636
6A	31	50	223	223	223	223	223	223	223	**223**
6B	31	65	233	233	241	233	233	233	233	**233**
6C	31	50	317	317	317	317	317	317	317	332
7A	40	66	279	279	279	279	279	279	279	288
7B	40	66	283	283	283	283	283	283	283	307
7C	40	66	334	334	334	334	334	334	334	356
8A	30	63	386	386	386	386	386	386	386	405
8B	30	63	395	395	395	395	395	395	395	426
8C	30	63	521	527	529	522	521	521	521	595
9A	50	92	323	323	323	323	323	323	323	340
9B	50	92	326	326	326	326	326	326	326	342
9C	50	92	332	332	332	332	332	332	332	355
9D	50	92	389	391	399	391	391	391	389	454
10A	50	97	428	428	428	428	428	428	428	438
10B	50	97	436	436	436	436	436	436	436	465
10C	50	97	446	446	451	446	446	446	446	477
10D	50	97	528	528	530	527	526	531	525	588

Note that the previous works did not report the *kshs* results so we chose a local search algorithm (GLS; Beullens et al., 2003) and Branch-and-cut-and-price algorithm (BCPA; Longo, Aragão and Uchoa, 2006).

- ***GLS***: reports the best results on *kshs, gdb, val* obtained from one run.
- ***BCPA***: reports the best results on *kshs, gdb, val, egl*.

These reference algorithms were tested on different computers, compilers and operating systems. Hence, a fair comparison cannot be made. The comparison is based on a solution quality. In terms of time, the DCSARP is much more time-consuming than all the compared algorithms.

As it can be seen from Table 3 and Figure 7, the results obtained by DCSARP and other algorithms (GLS and BCPA) in six instances of *kshs* are identical to the best solutions. Indeed, the Friedman test shows that our algorithm is competitive.

Figure 7 Freidman test for *kshs* instances (see online version for colours)

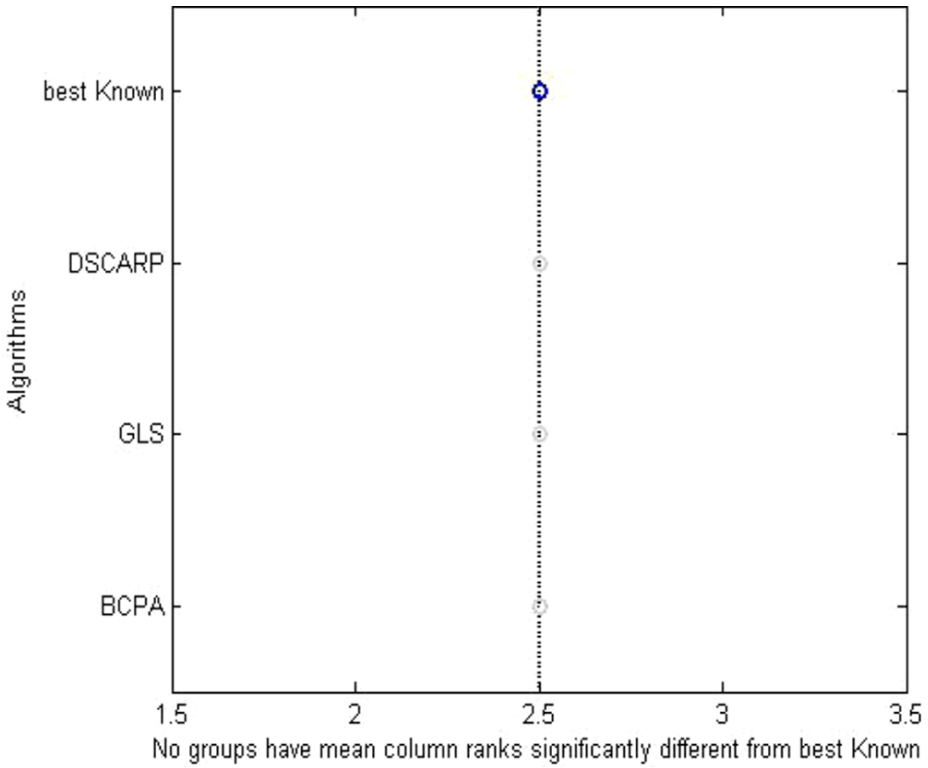


Table 4 and Figure 8 show the comparison for 23 *gdb* instances. As it can be seen, DCSARP produces better results in 20 instances. We have only three instances with results under the best known (gdb8, gdb9, gdb13). The Friedman test shows that there is no significant difference between our algorithm results and the best-known ones.

Figure 8 Freidman test for *gdb* instances

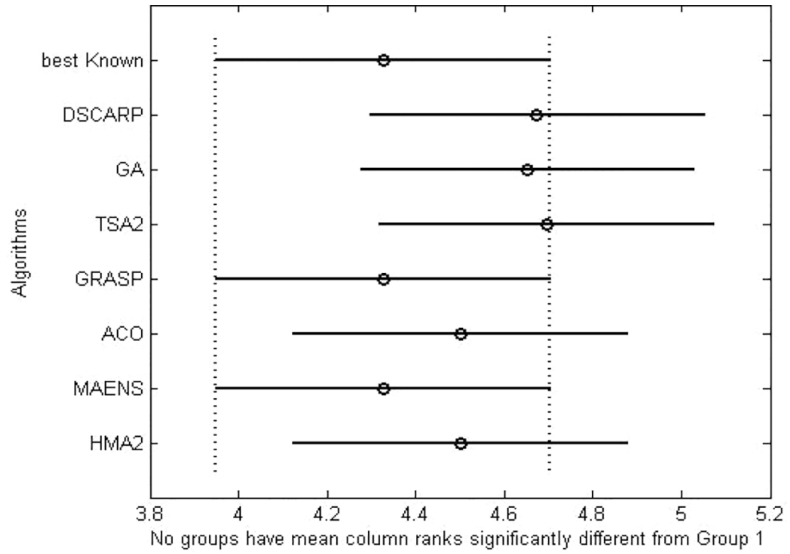
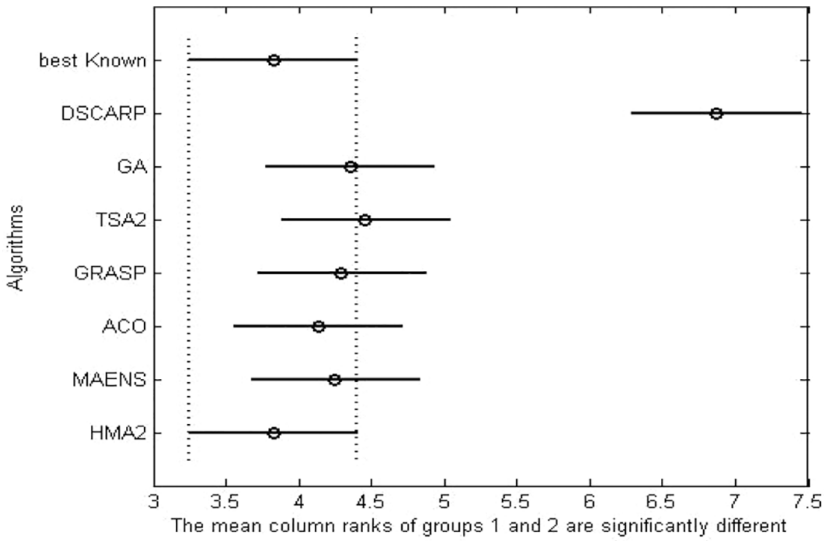


Table 5 and Figure 9 show the comparison for *val* instances. As we can see, our approach finds only 11 best known out of 34 instances.

Figure 9 Freidman test for *val* instances



6 Conclusion

The work contribution resides on the use of CS algorithm to solve the capacitated arc routing problem (CARP), and the mechanism of discretisation of continuous solution. The proposed algorithm called DCSARP provides an effective way to generate CARP solutions. DCSARP moves from one solution to another new one in discrete space using a sequence of integers based on lévy flight, and a decoding mechanism to transform the sequence of integers to CARP solution. The lévy flight function was adapted to maintain the moving in search space. Hence, two move operators (insertion and swap) are associated to lévy flight function. DCSARP is capable of achieving better solutions in *gdb* and *kshs* instances. Unfortunately, it did not succeed to get better solutions in large instances as *val* instances. Although DCSARP has shown excellent diversification in our experimental studies, the stagnation and high computational cost present its disadvantages. They remain to address in future works. As perspective, we can include a heuristic to maintain the convergence of DCSARP.

References

Bartolini, E., Cordeau, J.F. and Laporte, G. (2013) 'Improved lower bounds and exact algorithm for the capacitated arc routing problem', *Mathematical Programming*, Vol. 137, Nos. 1–2, pp.409–452.

Benavent, E., Campos, V., Corberan, E. and Mota, E. (1992) 'The capacitated arc routing problem: lower bounds', *Networks*, Vol. 22, No. 4, pp.669–690.

Beullens, P., Muyldermans, L., Cattrysse, D. and Van Oudheusden, D. (2003) 'A guided local search heuristic for the capacitated arc routing problem', *European Journal of Operations Research*, Vol. 147, No. 3, pp.629–643.

Brandão, J. and Eglese, R. (2008) 'A deterministic Tabu search algorithm for the capacitated arc routing problem', *Computers & Operations Research*, Vol. 35, No. 4, pp.1112–1126.

Chen, Y., Hao, J.K. and Glover, F. (2016) 'A hybrid metaheuristic approach for the capacitated arc routing problem', *European Journal of Operational Research*, Vol. 253, pp.25–39.

Corberan, A. and Laporte, G. (2015) *Arc Routing: Problems, Methods, and Applications*, MOS-SIAM series on optimization: 20, SIAM.

Dror, M. (2000) *Arc Routing: Theory, Solutions and Applications*, Kluwer Academic Publishers, Boston, USA.

Gandomi, A.H., Yang, X.S. and Alavi, A.H. (2011) 'Mixed variable structural optimization using firefly algorithm', *Computers & Structures*, Vol. 89, Nos. 23–24, pp.2325–2336.

Golden, B.L., DeArmon, J.S. and Baker, E.K. (1983) 'Computational experiments with algorithms for a class of routing problems', *Computers & Operations Research*, Vol. 10, No. 1, pp.47–59.

Golden, B.L. and Wong, R.T. (1981) 'Capacitated arc routing problems', *Networks*, Vol. 11, No. 3, pp.305–315.

Handa, H., Lin, D., Chapman, L. and Yao, X. (2006) 'Robust solution of salting route optimisation using evolutionary algorithms', *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, pp.3098–3105.

Kiuchi, M., Shinano, Y., Hirabayashi, R. and Saruwatari, Y. (1995) 'An exact algorithm for the Capacitated Arc Routing Problem using Parallel Branch and Bound method', *Abstracts of the 1995 Spring National Conference of the Oper. Res. Soc. of Japan*, 28–29.

Lacomme, P., Prins, C. and Ramdane-Cherif, W. (2004) 'Competitive memetic algorithms for arc routing problems', *Annals of Operations Research*, Vol. 131, No. 1–4, pp.159–185.

Longo, H., Aragão, M.P. and Uchoa, E. (2006) 'Solving capacitated arc routing problems using a transformation to the CVRP', *Computers & Operations Research*, Vol. 33, No. 6, pp.1823–1837.

Mei, Y., Tang, K. and Yao, X. (2009) 'A global repair operator for capacitated arc routing problem', *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 39, No. 3, pp.723–734.

Nesmachnow, S. (2014) 'An overview of metaheuristics: accurate and efficient methods for optimisation', *International Journal of Metaheuristics*, Vol. 3, No. 4, pp.320–347.

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

Ouaarab, A., Ahiod, B. and Yang, X.S. (2015) 'Cuckoo search applied to job shop scheduling problem', *Recent Advances in Swarm Intelligence and Evolutionary Computation, Studies in Computational Intelligence*, Vol. 585, pp.121–137.

Pearn, W.L. (1989) 'Approximate solutions for the capacitated arc routing problem', *Computers and Operations Research*, Vol. 16, No. 6, pp.589–600.

Pearn, W.L. (1991) 'Augment-insert algorithms for the capacitated arc routing problem', *Computers and Operations Research*, Vol. 18, No. 2, pp.189–198.

Polacek, M., Doerner, K.F., Hartl, R.F. and Maniezzo, V. (2008) 'A variable neighborhood search for the capacitated arc routing problem with intermediate facilities', *Journal of Heuristics*, Vol. 14, No. 5, pp.405–423.

Santos, L., Coutinho-Rodrigues, J. and Current, J.R. (2010) 'An improved ant colony optimization based algorithm for the capacitated arc routing problem', *Transportation Research Part B: Methodological*, Vol. 44, No. 2, pp.246–266.

Stern, H.I. and Dror, M. (1979) 'Routing electric meter readers', *Computers & Operations Research*, Vol. 6, No. 4, pp.209–223.

Talbi, E.L. (2009) *Metaheuristics: From Design to Implementation*, John Wiley & Sons, Inc. Publication, New York, USA.

Tang, K., Mei, Y. and Yao, X. (2009) 'Memetic algorithm with extended neighborhood search for capacitated arc routing problems', *IEEE Transactions on Evolutionary Computation*, Vol. 13, No. 5, pp.1151–1166.

Tasgetiren, M.F., Liang, Y.C., Sevkli, M. and Gencyilmaz, G. (2006) 'Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem', *International Journal of Production Research*, Vol. 44, No. 22, pp.4737–4754.

Ulusoy, G. (1985) 'The fleet size and mix problem for capacitated arc routing', *European Journal of Operational Research*, Vol. 22, No. 3, pp.329–337.

Usberti, F.L., Paulo, M.F. and André, L.M.F. (2013) 'GRASP with evolutionary path relinking for the capacitated arc routing problem', *Computers & Operations Research*, Vol. 40, No. 12, pp.3206–3217.

Welz, S.A. (1994) *Optimal Solutions for the Capacitated Arc Routing Problem Using Integer Programming*, PhD Thesis, Department of QT and OM, University of Cincinnati.

Yang, X.S. and Deb, S. (2009) 'Cuckoo search via lévy flights', *Proc of World Congress on Nature & Biologically Inspired Computing (NaBIC)*, India.