HYBRID ANT COLONY ALGORITHM FOR THE MULTI-DEPOT PERIODIC OPEN CAPACITATED ARC ROUTING PROBLEM

Bilal Kanso

Department of Computer Science, Lebanese University, Beirut

ABSTRACT

In this paper we present a hybrid technique that applies an ant colony optimization algorithm followed by simulated annealing local search approach to solving the Multi-Depot Periodic Open Capacitated Arc Routing Problem (MDPOCARP). This problem is a new variant of OCARP that has never been studied in the literature and consists of determining optimal routes in each period where each route starts from a given depot, visits a list of required edges and finishes by the last one. The final edge of the route is not required to be a depot. We developed a constructive heuristic, called Nearest Insertion Heuristic (NIH) to build an initial solution. The proposed algorithm is evaluated on three different benchmarks sets and numerical results show that the proposed approach achieves highly efficient results.

KEYWORDS

Arc routing problem, Periodic, Multi-Depot, Meta heuristic, Ant Colony Optimization, Simulated Annealing algorithm.

1. INTRODUCTION

The Capacitated Arc Routing Problem (CARP) is a hard-combinatorial optimization problem introduced which consists of a network (or graph) where edges have a positive length and some of the edges (so-called required edges) are served [1]. The CARP aims to find an optimal set of vehicles routes of total minimum length. Each route should start and end at a predefined depot and each required edge should be serviced in exactly one route. A wide range of real word applications can be modeled as arc routing problems such as transportation industry, collection of household waste, mail delivery, meter reader, etc.

The periodic CARP (PCARP) is a generalization of CARP in which routes are built over M periods that represent predefined schedules for visiting customers. A solution of the PCARP is a set of optimal routes for each planning period so that the both period constraint and capacitated vehicles along routes are satisfied. The Multi-Depot Open CARP (MDOCARP) is an extension of CARP in which vehicles can depart at more than one depot for servicing customers. The vehicles are not required to return back to a depot as in the CARP, instead they finish their route at the last served task. The Multi-Depot Periodic Open CARP (MDPOCARP) is a variant of OCARP in which the customer demands are served throughout planning periods. In MDPOCARP, there is a frequency f for each customer c indicating how many times within the planning period associated to c, the customer c must be visited.

In this paper, we intend to contribute to solve MDPOCARP. Then, we focus on three features of the CARP problem at the same time: the periodic, open and multi-depot CARP. The motivation for addressing this problem comes from the wide range of real-world applications that can be seen as MDPOCARP. Indeed, the study of both periodic and multi-depot CARP is receiving increasing attention, mostly in the last years, thanks to many real-world applications related to recycling, periodic deliveries of products to customers, periodic visits for providing specific services, Household waste collection, transportation of goods to supermarkets by manufacturing companies using trucks, maintenance of equipments, quality inspectors or home health care. In all these applications, vehicles start from a particular depot and finish at the last serviced customer. There is no need to return back at the end of the working period to the depots.

This work proposes an Ant Colony Optimization (ACO) combined with the guided local search meta-heuristic Simulated Annealing (SA) for the MDPOCARP. Computational experiments were performed to evaluate the proposed method. The results are compared to a common initial heuristic method. The obtained solutions are high quality and results reveal a good performance. Furthermore, the use of multiple depots is addressed where we will show by experiments that the number of depots and the total travelled distance are proportional. This means as the number of involved depots increases, the generated solutions are significantly better, and the total cost reached is decreased.

The remainder of this paper is organized as follows. Section 2 presents the related works associated to the studied problem. Section 3 describes the MDPOCARP problem and introduces an illustrative example. Section 4 presents a constructive initial solution and describes our proposed meta heuristic for solving the MDPOCARP problem. Section 5 shows the numerical experiments that we conducted on small, medium and large new instances. Finally, our conclusions follow in Section 6.

2. RELATED WORKS

In the best of our knowledge a solution for the studied MDPOCARP problem has never been developed in the literature, in spite of important practical problems could be easily modelled as an MDPOCARP. In this section, we present a brief overview of contributions which are technically close to our proposal, but which focus on various aspects of routing optimization problems. We discuss the difference between the problematic addressed by those contributions and those addressed by our approach. The CARP case with the characteristics studied in this work has not been extensively studied. The periodic CARP (PCARP) has been introduced by Lacomme et al. [2]. They proposed a memetic algorithm based on a linear sophisticated crossover. Recently, a two Phased Hybrid Local Search (PHLS) algorithm was proposed by Chen and Hao [3] for the PCARP. Usberti et al. [4] have addressed a variant of the classical open capacitated arc routing problem (OCARP) in which the set of edges are extended by depots and routes are not required to take the form of cycles. This variant of OCARP has been also solved by a hybrid genetic algorithm in [5]. The classical OCARP studied in this work, where a departure from a depot is mandatory by each vehicle, has been studied by Fung and Liu [6] . They proposed a mathematical formulation and generated lower bounds. They also developed a memetic algorithm which turns out superior than the classical genetic algorithm in solution quality for the classical OCARP. The Multi-depot CARP problem was developed by Kansou et Yassine [7] where two ant colony approaches and a memetic algorithm based on the splitting procedure and a special crossover have been proposed. Regarding the vehicle routing problem VRP on nodes case, several important works have been proposed. The periodic vehicle routing problem PVRP with intermediate facilities problem was introduced by Angelelli and Speranza in [8]. They solved it by a Tabu Search algorithm and presented computational results on a set of random generated instances and on a set of PVRP instances taken from the literature. Archetti et al. [9] worked on the flexible periodic VRP and a mathematical formulation has been proposed. Cornillier et al. [10] developed heuristics for the multi-depot petrol station replenishment problem with time windows while Cornillier et al. [11] developed an efficient heuristic for the same problem with unique depot and multiple periods without time windows. Popovic et al. [12] proposed a VNS algorithm to solve the multi-product multi-period inventory routing problem in fuel delivery case. Vidal et al. [13] implemented a hybrid genetic algorithm for the multi-depot periodic vehicle routing problem VRP, heterogeneous capacitated vehicles and constrained route duration. The same last problem with identical vehicles was studied by Rahimi-Vahed et al. [14] where a modular heuristic algorithm was proposed. The authors worked on three vehicle routing problems, i.e., periodic, multi-depot and periodic with multi-depot problem. They conducted their experiments using suitable instances that show the effectiveness of the proposed method in terms of both quality and computational time. Nguyen et al. [15] proposed a genetic algorithm for the periodic VRP with time windows. They introduced two new crossover to tackle simultaneously the periodic and time windows cases. They found new best solutions for a number of large instances. The periodic VRP for Retail Distribution of Fuel Oils studied by Carotenuto et al. [16], a hybrid genetic algorithm was developed for solving it on a set of real case studies and on a set of randomly generated instances. Carotenuto et al. [16] provided a mathematical formulation for the multi-depot periodic VRP. They also proposed a hybrid genetic algorithm for solving the problem due to the large size of the real instances which the company has to deal with. Ran Liu et al. [17] studied for the first time the multi-depot open VRP (MDOVRP). The authors presented a mixed integer programming mathematical formulation and an effective meta-heuristic solution approach (hybrid genetic algorithm) for the resolution. Eduardo Lalla-Ruiz et al. [18] improved the last formulation proposed by [17] for the same problem MDOVRP and they provided better results in terms of solution quality and tighter linear bounds.

3. Problem Description

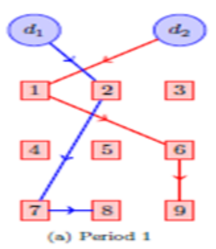
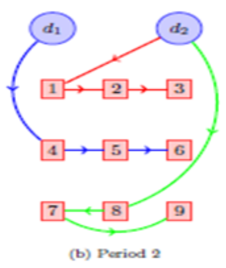
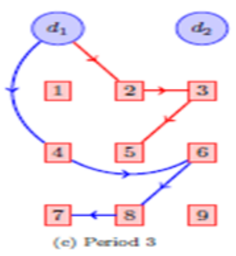
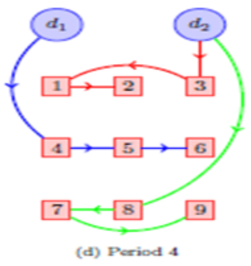
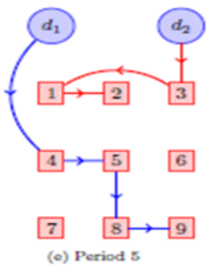
The Multi-Depot Periodic Open problem (MDPOCARP) is an extension of Open CARP. It consists of a fleet of identical vehicles. Each vehicle is assigned to one depot and cannot exceed its capacity. The vehicle starts its route from one depot and it ends at the least served task. The tasks must be served a given number of periods (which is equal to its frequency ft on a whole multi period horizon H). H is composed of np periods and each task t has a predefined set of possible service combinations of periods Ct such that $f_t \leq n_p$. Each k belonging to a combination Ct must contain ft periods where the task will be served, i.e. $|k| = f_t$, $\forall$ $k \in C_t$, $\forall$ $t \in E_r$. The number of tasks, or required edges, will be denoted by n and the cost of each task t is C(t). In the experiments conducted in this work, we will use $n_p = |H| = 5$ periods (see Section 5). In order to code the combinations of periods, we use the binary code and five sets of combinations depending on the frequencies. For example, the combination of five periods 10100 means that the task has the frequency 2 and will be served in period 1 and period 3.

Table 1: Information about an example with nine tasks

Tasks	1	2	3	4	5	6	7	8	9
Frequency	5	5	3	4	4	4	5	5	3
Combination	11111	11111	01011	01111	01111	11110	11111	11111	11010
Closest depot	d_2	d_1	d_2	d_1	d_2	d_2	d_1	d_2	d_1

Table 1 presents the information about a concrete example of the studied problem. It consists of 9 tasks that should be visited in 5 periods according to given frequencies. For example, Task 5 must be visited 4 times and Task 9 must be visited 3 times. Each task t has a given combination indicating the periods in which the Task t should be visited. For example, Task 5 should be visited in all periods except Period 1 and Task 9 should be visited in Period 1, Period 2 and Period 4. Table 2 illustrates the routes representing the solutions of the studied problem MDPOCARP. For example, the solution in period 1 consists of two routes (d_1, $task_2$, $task_7$, $task_8$) and (d_2, $task_1$, $task_6$, $task_9$) while the solution in period 4 consists of three routes (d_1, $task_4$, $task_5$, $task_6$), (d_2, $task_3$, $task_3$, $task_1$) and (d_2, $task_8$, $task_7$, $task_9$).

Table 2: example of a feasible MDPOCARP solution

4. Algorithm for the MDPOCARP Problem

We propose a Hybrid algorithm (that we call HACSA) based on both the Ant Colony optimization (AC) and the Simulated Annealing (SA) local search algorithm to solve the MDPOCARP. Ant colony algorithms are global search optimization techniques based on the foraging behavior of real ants, which have been found to be very effective in solving the travelling salesman problem TSP [19] [20] and the capacitated arc routing problem CARP [1]. In this work, we construct an initial solution S0 by a constructive heuristic explained in the next section. The AC algorithm uses a colony of Na artificial ants where each ant k represents a giant route (order of all different tasks) without delimiters, i.e. the capacity constraint is not verified. At each iteration of the AC, every ant starts its route from a random task. Moving of ants depends on two amounts: (1) Visibility amount μij which is a constant and represents the inverse of the distance C_{ij} between tasks i and j (C_{ij} = C(i, j) is the shortest distance calculated by Dijkstra's algorithm from the end node of i to the start node of j); (2) Pheromone amount τij which is a variable and represents initially the inverse of the cost of initial solution S_0, i.e. $\tau ij = \frac{1}{C(S0)}$, ∀ i, j ∈ set of tasks. The cost $C(S_0)$ represents the sum of distances traversed by all M vehicles on all periods. Note that the amount of pheromone will be updated at the end of each iteration v by the following rule $\tau_{ij}(v+1) = \rho\tau_{ij}(v) + (1 - \rho)\Delta\tau_{ij}$, where ρ is the evaporation parameter in [0,1], $\Delta\tau ij = \frac{1}{C(best)}$ if (i,j) is used by the ant that gives the best solution best and it will be zero otherwise.

4.1. Initial Solution for MDPOCRP

To build the initial solution of the proposed Hybrid Ant Colony algorithm, we propose a constructive heuristic, called Nearest Insertion Heuristic (NIH), which constructs the solution in sequential manner and mainly relies on the insertion method of tasks into a current solution. This heuristic consists in:

- Sorting in increasing order all tasks to be inserted in an empty solution according to the distance from each task to its associated closet depot;
- Choosing the starting task s of the route as the one which has the minimal distance from closet depot to it;

- Choosing other tasks of the route in such a way the next task after the last selected task among the candidate tasks using the nearest neighbor rule i.e. the closet task to the last selected one;
- For each task t in the constructed route, we choose the best combination ct for task t and then we insert it into each day of c_t. c_t is the best combination that minimizes the insertion cost of t in the current solution, i.e., into each day d of that combination. Note that, in a given day, the insertion cost is the cost of inserting t after the last inserted task if the capacity constraint and the maximal distance constraints are verified. For the other case, it represents the cost to insert it at the beginning of a new route in the same day d.

4.2. Local Search Procedure

In order to improve the obtained solutions using our ACO-based algorithm, we integrate the Simulated Annealing local search method (SA). The (SA) is a heuristic which is commonly used to prevent the optimization algorithm to quickly falling into local minimum [21] [22]. It is based on random acceptance strategy with certain probability. At each iteration of our proposed (SA) algorithm, we will use one of the following four moves:

- Swap: this move consists in exchanging the positions i and j of two different tasks belonging to the same feasible route;
- Relocate: this move consists in removing one task from a position i and putting it in another position j chosen in the same route;
- 2-opt: this move consists in choosing two positions i and j in a given route R and then re-organizes R by replacing the sequence of tasks (i, i+1, … ,j) by the sequence (inverse(j), inverse(j-1), … , inverse(i)) where inverse(t) represents the inverse task for the task t;
- Cross-exchange: this move involves two routes R_1 and R_2 where it chooses a task in a position i of R_1 and a task in a position j of R_2. It generates two new routes R_1' and R_2' as follows: R_1' takes the first part of R_1 from first task to task located at position i completed by another part of the route R_2 starting with the task located at position j. R_2' is built in the same way and that by linking the remaining first part of R_2 with the second remaining part of R_1.

The algorithm used in the local search procedure is a Simulated Annealing (SA) method presented in **Algorithm 1.**

Algorithm 1: Simulated Annealing algorithm

```
Input: $T_{min}$, $T_{max}$ and K
Initialization: a feasible solution S, T=$T_{max}$
while $T > T_{min}$ do
    Generate a feasible neighborhood solution S' by applying a random local search move i.e.
    swap, Relocate, 2-opt or Cross-exchange)
    Calculate ΔE = Cost(S')-Cost(S)
    Calculate the probability P(ΔE, T) = $e = e^{-\frac{\Delta E}{T}}$
    If ΔE < 0 then
        Accept S' and update the best solution S = S'
    Else
        Accept S' with the probability P(ΔE, T)
    Update the temperature T = K × T
Return the best solution found so far S.
```

4.3. Ants Production

Suppose that Visitedk is the set of already visited tasks by ant k. An ant k moves from the last visited task i to another task j by choosing the closest task j (j $\notin$ Visited$_k$) to i if the random variable q is strictly less than q_0. Otherwise, it is chosen by using the following probability decision rule:

$$P_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha \times \mu_{ij}^\beta}{\sum_{k \notin Visited_k} \tau_{ik}^\alpha \times \mu_{ik}^\beta} & \text{if } j \notin Visited_k \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where α and β are respectively the pheromone and visibility parameters, q0 is a constant parameter belongs to [0, 1] and q is a uniform random parameter in [0, 1]. This move of every ant is called the transition rule to add a task at the end of every sequence of tasks already constructed for every ant k. Once each ant is ready, then we will construct an associated feasible MDPOCARP solution by apply the following insertion procedure.

4.4. Insertion Procedure

Consider an ant k = (t_1^k, t_2^k, …, t_n^k) where each t_i^k is the task of order i in k. We associate firstly for k an empty solution S^k = (S_1^k, S_2^k, …, S_{nd}^k) where each S_d^k, d = 1, …, nd, is the associated subsolution to the period d that contains firstly M empty routes. Secondly, we will insert each task t_i^k indicated in the order of k into f_{tik} sub-solutions (ft is the associated predefined frequency of task t). For the task t_1^k we will find the best combination of days (from the associated set of predefined day combinations Ct1k that minimizes the cost to insert it in the associated sub-solutions. Note that the insertion cost of a task $t1^k$ in a combination c ∈ C_{t1k} is Insertion = $\Sigma_{d \in c}$ C(CD_{t1k}, t_1^k) where CD_t represents the closest depot to task t. And the general insertion cost to insert another task t in a combination c ∈ C_t is Insertion:

$$\sum_{d \in C} \min_{l \in S_d^k} [C(CD_t, t) + C(t, first) - C(CD_{first}, first), C(last, t), C(l, t) + C(t, l+1) - C(l, l+1)]$$

With *first* and *last* denote respectively the first served task in the current route S_{dk} and its last served task, l+1 denotes the task next to t and it can be nothing if l is the last served task (in this case C(t, l+1)=C(l, l+1)=0). We will continue to insert the other tasks, one after the other but always checking the capacity constraint of each route when trying to find the best possible combination. It means that when a route in a such period (a sub-solution) can no longer serve the tested task (selected to insert), we open another route in this period and again by checking that the number of routes will not exceed M. In case if all M routes in a given period p of a day combination c ∈ C_t are closed to a given task t due to the capacity Q and maximal number of routes M constraints, we will study another day combination c' ∈ C_t that will not involve the period p. The insertion procedure finishes when all tasks t_i^k (by the order i =1. …, n) will be inserted in the solution S^k that will be a MDPOCARP feasible solution.

Suppose that R_{dm^k} = (t_1^{dm}, ..., $t_{|Rdmk|}^{dm}$) is the m_{th} route in the period d of the solution S_k, then the total cost of a solution S_k is calculated by the following rule:

$$Cost = \sum_{l=1}^{n} f_l C(l) + \sum_{d=1}^{n_p} \sum_{m=1}^{n} C\left(CD_{t_1^{dm}}, t_1^{dm}\right) + \sum_{m=1}^{j=|R_{dm}^{k}|} C\left(t_{j-1}^{dm}, t_j^{dm}\right)$$

4.5. The Proposed HACSA Algorithm

The proposed Hybrid Ant Colony with the Simulated Annealing algorithm HACSA is shown in **Algorithm 2.**

Algorithm 2: The HACSA algorithm

```
Initialization:
Find a feasible initial solution $S_0$ with cost $C(S_0)$ by the heuristic NIH (see SubSection 4.1)
$\tau_{ij} = \frac{1}{C(S0)}$ and $\mu_{ij} = \frac{1}{Cij}$ $\forall$ i, j ∈ set of tasks
Make empty $N_a$ ants, define $\alpha$, $\beta$, $\rho$, $p_r$ and iterH
iter = 1
while iter ≤ iterH do
    choose a randomly task to start for each ant k = 1, ..., Na
    construct the remaining sequence of tasks of every ant k using the transition rule
    every ant k is used to construct a MDPOCARP solution by using the insertion procedure
    apply the local search procedure (AS) defined in Algorithm 1 on each obtained feasible
    solution
    update both the best solution found so far and pheromone quantities
    iter = iter + 1
Return the best solution found so far.
```

5. Numerical Experiments

5.1. Benchmark instances

In [5] the authors tested the hybrid genetic on the OCARP instances derived from the benchmark of CARP instances, which includes 23 gdb ([23]) 34 val ([24]), and 24 egl ([25]) instances. The authors in [2] developed a method to build the periodic instances for the original CARP problem using a horizon of 5 periods on the gdb and egl instances. They assigned a random frequency between 1 and 5 to each task and then built a set of combinations of periods for each frequency. For the studied problem MDPOCARP, we use the same OCARP instances used in [5]. Similarly, to [2] we generate our benchmark instances for three new sets: 23 mdpogdb, 34 mdpoval and 24 mdpoegl, totaling 81 instances. We constructed five sets of combinations depending on the frequencies. The minimal value of the number of vehicles M used in our experiments is the minimal number of vehicles necessary for finding a feasible solution and is obtained by carrying out 2000 random solutions to have at least 50% of feasible solutions between them. We added three depots to Golden's instances, four depots to Benavant's instances and six depots to Eglese's instance. For all instances, the first node represents a depot and the other depots are randomly selected.

5.2. Results

The experiments were implemented in Java and run on a 2.6GHz Dual Core computer with a memory of 16GB under Windows Ten. Tables 3, 4 and 5 show the results for all instances after running 1000 iterations of our algorithm. For each table, the columns correspond respectively to: (1) problem name instance; (2) the number of vehicles M; (3) cost obtained by NIH method; (4) cost obtained with our HACSA method; (5) running time T in seconds for HACSA ; (6) gap in percentage of HACSA cost compared to NIH. The gaps G_G(%), G_B(%) and G_E(%) are respectively the percent improvement of AS solution cost with respect to initial solution NIH. They are computed as follows:

$$G_X(\%) = \frac{\text{NIH} - \text{HACSA}}{\text{NIH}} \text{ X } 100 \text{ with } X \in \{G, B, E\}$$

The average gaps G_G, G_B and G_E are respectively 31.8, 45 and 42.4. The overall gap averages 39.7 between the obtained solution cost with HACSA method and that obtained with the initial method NIH for the 81 test instances shows that the HACSA method improves significantly the quality of the solutions and provides better solutions. The average running time 33.7 seconds shows the performance of our approach.

Table 3: Results of HACSA algorithm on Golden's set instances

Instance	M	NIH	HACSA	T	G_G	Instance	M	NIH	HACSA	T	G_G
mdpogdb1	5	1152	774	24	32.8	mdpogdb12	8	1559	1044	23	33
mdpogdb2	5	1372	898	36	34.5	mdpogdb13	7	1923	1552	19	19.3
mdpogdb3	5	955	668	23	30.1	mdpogdb14	5	424	275	14	35.1
mdpogdb4	4	1135	686	16	39.6	mdpogdb15	4	237	171	15	27.8
mdpogdb5	5	1388	953	45	31.3	mdpogdb16	6	495	357	41	27.9
mdpogdb6	5	1478	872	33	41	mdpogdb17	5	347	259	29	25.4
mdpogdb7	5	1154	804	32	30.3	mdpogdb18	5	691	489	68	29.2
mdpogdb8	10	1417	852	109	39.9	mdpogdb19	3	198	128	1	35.4
mdpogdb9	8	1410	860	82	39	mdpogdb20	5	471	326	17	30.8
mdpogdb10	4	1037	675	25	34.9	mdpogdb21	6	722	499	24	30.9
mdpogdb11	5	1865	1170	84	37.3	mdpogdb22	8	741	594	111	19.8
						mdpogdb23	10	934	682	105	27

Table 4: Results of HACSA algorithm on Benavent's set instances

Instance	M	NIH	HACSA	T	G_B	Instance	M	NIH	HACSA	T	G_B
mdpoval1A	2	1268	576	91	54.6	mdpoval6A	9	2549	2056	23	19.3
mdpoval1B	4	1004	535	108	46.7	mdpoval6B	4	1344	702	3	47.8
mdpoval1C	9	820	523	48	36.2	mdpoval6C	10	1160	760	2	34.5
mdpoval2A	2	1484	706	82	52.4	mdpoval7A	10	3908	3033	94	22.4
mdpoval2B	3	1263	682	58	46	mdpoval7B	5	2280	1121	17	50.8
mdpoval2C	9	917	599	55	34.7	mdpoval7C	9	1964	1034	8	47.4
mdpoval3A	4	558	243	55	56.5	mdpoval8A	3	2689	1313	16	51.2
mdpoval3B	3	467	210	65	55	mdpoval8B	4	2339	1188	10	49.2
mdpoval3C	8	408	229	38	43.9	mdpoval8C	13	3212	2102	16	24.6
mdpoval4A	5	3600	1957	6	45.6	mdpoval9A	4	2174	978	24	55
mdpoval4B	4	2612	1251	4	52.1	mdpoval9B	6	2675	1484	41	44.5
mdpoval4C	5	2816	1311	3	53.4	mdpoval9C	5	2328	1099	24	52.8
mdpoval4D	17	3111	2102	14	32.4	mdpoval9D	10	2222	1194	15	46.3
mdpoval5A	4	3316	1771	12	46.6	mdpoval10A	4	3326	1774	43	46.7
mdpoval5B	8	3978	2769	21	30.4	mdpoval10B	6	3003	1609	63	46.4
mdpoval5C	6	2949	1452	8	50.8	mdpoval10C	6	3116	1635	43	47.5
mdpoval5D	9	2822	1473	6	47.8	mdpoval10D	9	2818	1489	20	47.2

Table 5: Results of HACSA algorithm on Eglese's set instances

Instance	M	NIH	HACSA	T	G_E	Instance	M	NIH	HACSA	T	G_E
mdpoegle1A	5	12551	7481	8	40.4	mdpoegls1A	7	17829	9020	15	49.4
mdpoegle1B	7	12325	7201	6	41.6	mdpoegls1B	10	20362	10454	10	48.7
mdpoegle1C	10	10694	7464	6	30.2	mdpoegls1C	14	15091	9561	9	36.6
mdpoegle2A	7	19618	10365	6	47.2	mdpoegls2A	13	41749	20577	40	50.7
mdpoegle2B	10	16530	10293	11	37.7	mdpoegls2B	19	36258	20439	41	43.6
mdpoegle2C	14	19435	10974	12	43.3	mdpoegls2C	26	35044	21225	35	39.4
mdpoegle3A	8	21355	11315	18	47	mdpoegls3A	13	37034	18947	42	48.8
mdpoegle3B	12	22379	12081	14	46	mdpoegls3B	20	382112	22216	35	41.9
mdpoegle3C	17	20407	14080	14	31	mdpoegls3C	28	40054	25301	55	36.8
mdpoegle4A	9	25803	12960	18	49.8	mdpoegls4A	18	52135	24229	56	53.5
mdpoegle4B	14	22188	13336	16	39.9	mdpoegls4B	25	47873	26544	47	44.6
mdpoegle4C	20	23029	15756	15	31.6	mdpoegls4C	35	59951	37035	63	38.2

5.3. Parameters

The results in Tables 3, 4 and 5 are given by executing ten runs of the algorithm HACSA for each instance. For all runs, the HA algorithm parameters were set as: N_a =25, q0 = 0.5, p_r =0.7, α = 1, β = 1, ρ = 0.2 and iterH = 1000. We selected these parameters values after previous empirical tests.

5.4. The Effect of using Multi-Depots

Table 6 reports the results when executing our algorithm HACSA on Eglese's instances with 3 depots. The first column is the cost obtained by HACSA method while the second column is the running time in seconds. The third column reports the gap G36 between $HACSA_3$ (with 3 depots) and $HACSA_6$ (with 6 depots) which is computed as

$$G_{36}= \frac{\text{HACSA3}-\text{HACSA6}}{\text{HACSA3}} \text{ X } 100$$

The fourth column diffG63 = G_6 - G_3 reports the difference between the gap GE with 6 depots and the gap G_E with 3 depots.

Table 6: Comparison of HACSA algorithm with multiple depots on Eglese's instance set.

Instance	$HACSA_3$	T_3	G_E(%)	G_{36}(%)	DiffG63
mdpoegle1A	7919	9	36.9	5.531	3.4
mdpoegle1B	7532	7	38.9	4.395	2.7
mdpoegle1C	8822	8	17.5	15.393	12.7
mdpoegle2A	10589	11	46	2.115	1.2
mdpoegle2B	10978	17	33.6	6.24	4.1
mdpoegle2C	12610	19	34.8	12.974	8.5
mdpoegle3A	11580	19	45.8	2.288	1.2
mdpoegle3B	13012	16	41.9	7.155	4.1
mdpoegle3C	17783	19	12.9	20.823	18.1
mdpoegle4A	14891	25	42.3	12.968	7.5
mdpoegle4B	14444	23	34.9	7.671	5
mdpoegle4C	16289	17	29.3	3.272	2.3

Table 7: Comparison of HACSA algorithm with multiple depots on Eglese's instance set.

Instance	$HACSA_3$	T_3	G_E(%)	G_{36}(%)	DiffG63
mdpoegls1A	9583	13	46.3	5.875	3.1
mdpoegls1B	11575	15	43.2	9.685	5.5
mdpoegls1C	10934	11	27.5	12.557	9.1
mdpoegls2A	20951	52	49.8	1.785	0.9
mdpoegls2B	23148	50	36.2	11.703	7.4
mdpoegls2C	23083	56	34.1	8.049	5.3
mdpoegls3A	19658	59	46.9	3.617	1.9
mdpoegls3B	24212	49	36.6	8.244	5.3
mdpoegls3C	26903	57	32.8	5.955	4
mdpoegls4A	27050	64	48.1	10.429	5.4
mdpoegls4B	29510	66	38.4	10.051	6.2
mdpoegls4C	37810	96	36.9	2.05	1.3

Figure 1 and Figure 2 illustrates the importance of using multiple depots. It reports the difference of results between the use of 3 depots and 6 depots for the Eglese's test instances. Figure 1 illustrates a histogram that represents the difference of costs obtained by using 3 depots and 6 depots and Figure 2 shows a curve that represents the difference between the gap G_E when using 3 depots and 6 depots. Both figures emphasize that using 6 depots instead of 3 depots decreases significantly the obtained cost. The average gap G_E when using 3 depots is 37.15 while the average gap G_E when using 6 depots is 42.4. That yields that the average total distance improvement is 5.25%. Consequently, as the number of depots increases, the obtained average cost decreases.

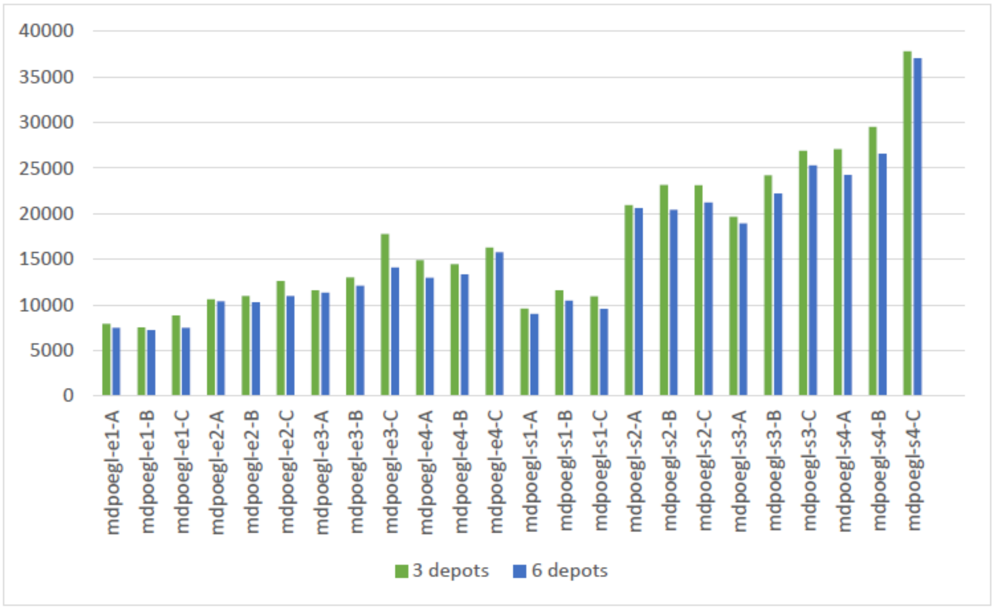


Figure 1: HACSA3's costs vs. HASCSA6's costs

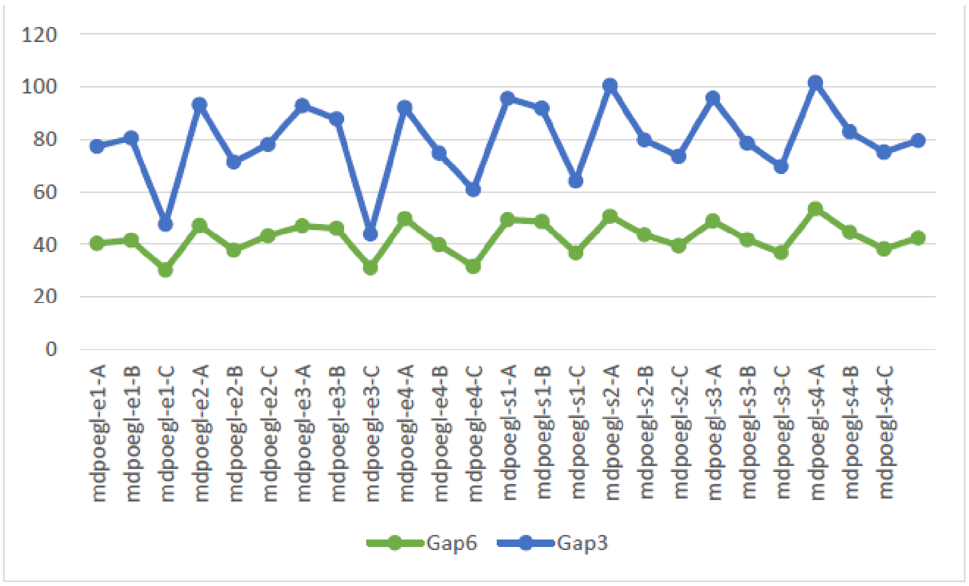


Figure 2: GE with 3 depots vs. GE with 6 depots

6. Conclusion

This work introduces for the first time the Multi-Depot Periodic Open Capacitated Arc Routing Problem (MDPOCARP) which is a NP-hard problem optimization problem deriving from the family of arc routing problem. Firstly, a constructive heuristic was developed for generating initial solution for MDPOCARP which relies on finding nearest neighbor. Secondly, a hybrid ant colony algorithm (HACSA) was proposed. It is composed by the ant colony optimization algorithm combined with both insertion procedure and local search algorithm. The insertion procedure is used to construct feasible ants. The local search uses swap, relocate, 2-opt and cross-exchange moves.

The computational experiments, using three set of benchmarks mdpogld, mdpoval and mdpoegl derived respectively from the CARP instances [23], [24], [25] has shown that HACSA generates good and promising solutions for MDPOCARP.

References

[1] B. L. G. a. R. T. Wong., «Capacitated arc routing problems,» 1981, p. 305–315.

[2] C. P. a. W. R.-C. P. Lacomme, «Evolutionary algorithms for periodic arc routing problems,» European Journal of Operational Research, n° %1Project Management and Scheduling, p. 165(2):535 – 553, 2005.

[3] Y. C. a. J. Hao., «Two phased hybrid local search for the periodic capacitated arc routing problem,» European Journal of Operational Research, 06 2017.

[4] P. F. a. A. F. F. Usberti, «The open capacitated arc routing problem,» Computers & Operations Research, p. 38(11):1543 – 1555, 2011.

[5] R. A. a. F. Usberti, «Hybrid genetic algorithm for the open capacitated arc routing problem,» Computers & Operations Research, p. 90:221 – 231, 2018.

[6] R. L. a. Z. J. R.Y.K. Fung, « A memetic algorithm for the open capacitated arc routing problem,» Transportation Research Part E: Logistics and Transportation Review, p. 50:53 – 67, 2013.

[7] A. K. a. A. Yassine, «New upper bounds for the multi-depot capacitated arc routing problem,» IJMHeur, p. 1(1):81–95, 2010.

[8] E. A. a. M. Speranza, «The periodic vehicle routing problem with,» European Journal of Operational Research,, p. 137:233–247, 2002.

[9] E. F. a. D. H.-M. C. Archetti, «The flexible periodic vehicle routing problem,» Computers & Operations Research, p. 85, 2017.

[10] F. B. a. J. R. F. Cornillier, «Heuristics for the multi-depot petrol station replenishment problem with time windows,» European Journal of Operational Research, p. 220:361–369, 07 2012.

[11] F. B. G. L. a. J. R. F. Cornillier, «A heuristic for the multiperiod petrol station replenishment problem.,» European Journal of Operational Research, p. 191(2):295 – 305, 2008.

[12] M. V. a. G. R. D. Popovic, «Variable neighborhood search heuristic for the inventory routing problem in fuel delivery.,» Expert Systems with Applications, p. 39(18):13390 – 13398, 2012.

[13] T. C. M. G. N. L. a. W. R. T. Vidal, « A hybrid genetic algorithm for multidepot and periodic vehicle routing problems>,» Operations Research, p. 60:611–624, 2012.

[14]T. C. M. G. a. W. R. A. Rahimi-Vahed, «Fleet-sizing for multidepot and periodic vehicle routing problems using a modular heuristic algorithm.,» Computers & Operations Research, p. 53:9 – 23, 2015.

[15] T. C. a. M. T. P. Nguyen, «A hybrid generational genetic algorithm for the periodic vehicle routing problem with time windows,» Journal of Heuristics, p. 20:383–416, 08 2014.

[16] S. G. S. M. a. F. V. P. Carotenuto, «Periodic capacitated vehicle routing for retail distribution of fuel oils.,» vol. volume 10, p. 735–744, 2015.

[17] Z. J a. N. G. R. Liu, «A hybrid genetic algorithm for the multi-depot open vehicle routing problem,» OR Spectrum, p. 36(2):401–421, Mar 2014.

[18] C. I. S. T. a. S. V. E. Lalla-Ruiz, «An improved formulation for the multi-depot open vehicle routing problem,» OR Spectrum, p. 38:1–13, 07 2015.

[19] M. V. C. A. Dorigo M., «Ant system: optimization by a colony of cooperating agents.,» IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 26, pp. 29 - 41, 1996.

[20] T. D. M. Stutzle, «A short convergence proof for a class of ant colony optimization algorithms.,» IEEE Transactions on Evolutionary Computation, vol. 6, pp. 358 - 65, 2002.

[21] P. J. M. A. E. H. L. L. J. van Laarhoven, «Job shop scheduling by simulated annealing.,» Operational research, vol. 40, pp. 113-25, 1992.

[22] I. P. C. Osman, «Simulated annealing for permutation flow-shop scheduling,» Omega, vol. 17, pp. 551 - 17, 1989.

[23] B. D. J. B. E. Golden, « Computational experiments with algorithms for a class of routing problems,» Computers & Operations Research, vol. 10, pp. 47 - 59, 1983.

[24] E. C. V. C. A. M. E. Benavent, «The capacitated arc routing problem: Lower bounds.,» Networks, vol. 22, pp. 669 - 90, 1992.

[25] R. W. Eglese, «Routing winter gritting vehicles.,» Discrete Appl. Math., vol. 48, pp. 231 - 44 , 1994.