Ant Colony System for the Periodic Capacitated Arc Routing Problem

Ali KANSOU* & Adnan YASSINE**

*Laboratoire de Mathématiques Appliquées du Havre (LMAH), Université du Havre, France
25 rue Philippe Lebon, B.P. 540, 76058 LE HAVRE cedex

**Institut Supérieur d’Études Logistiques (ISEL), Université du Havre, France
ali.kansou@yahoo.fr ; adnan.yassine@univ-lehavre.fr

Abstract

In this paper, we present a new Ant Colony System (ACS) approach for the Periodic Capacitated Arc Routing Problem (PCARP) on a mixed network. In the CARP problem the objective is to find routes starting from the depot such that each required arc/edge (called task) is serviced, capacity constraints are satisfied and total travel cost is minimized. The PCARP extends the classical CARP problem to a planning horizon of several days. Each task requires a certain number of visits within this time horizon. Hence, the objective of the problem is to choose the visit days for each task and to solve a CARP for each day in order to find efficient routes over the whole horizon. In our approach, we suggest a new technique for solving the PCARP problem. We have combined the ACS algorithm with a heuristic for insertion of tasks in a current solution. Hence, the ACS is used to optimize the order of tasks for insertion and the heuristic is used for insertion the tasks. The results of computational experiments carried on problem taken from literature indicate that the proposed approach outperforms existing algorithms in most cases. A comparison with the best memetic methods is presented and the results prove the robustness, the rapidity and the performance of our method.

Keywords: Periodic Capacitated Arc Routing Problem, Ant Colony System, Insertion Heuristic.

1 Introduction

The Periodic Capacitated Arc Routing Problem (PCARP) generalizes the Capacitated Arc Routing Problem (CARP) by extending the single work period (day) to a planning period $H$ of several days of work. The well know NP-hard problem CARP is introduced by Golden et al. in [10]. In this paper, the CARP is defined on a mixed network $M$ in which a fleet of identical vehicles, with limited capacity, is based at a depot node. Each link (arc or edge) in $M$ has a non-negative traversal cost and a non-negative service cost which are different and can be traversed at any time. A subset of required links, called tasks (a task has a demand strictly positive), must be serviced by one single vehicle. Then, the Mixed CARP (MCARP) consists of determining a set of feasible vehicle trips that minimizes the total cost of traversed links. Each vehicle trip starts and ends at the depot and the total demand serviced by any vehicle must not exceed the vehicle capacity. In this paper, the treated problem is the Periodic Mixed Capacitated Arc Routing Problem (PMCARP). Each task $t$ must be serviced $f_t$ times over $H$ in such a way that each time in a one single day and by one single vehicle in this day. The objective to PMCARP is to minimize the total distance travelled over all days of $H$. This problem is also NP-hard and arise in several applications like municipal waste collection, street sweeping, winter gritting, mail delivery, etc.
The first methodological paper devoted to a periodic arc routing problem was realized by Ghiani et al. in [9]. It presents a heuristic method for the periodic rural postman problem. Chu et al. proposed a linear programming model and constructive heuristics to solve the PCARP. The scatter search method is presented in [3]. Three heuristics methods are presented by Chu et al. in [4] to resolve this problem. The memetic algorithm is proposed by Lacomme et al. in [12] where the two decisions of planning (affectation of tasks to the days of $H$) and scheduling, are simultaneously treated. The principle of this memetic algorithm is based on a sophisticated crossover operator. The Periodic Vehicle Routing Problem PVRP is pioneered by Beltrami and Bodin in 1974 and several heuristics (Russell and Gribbin in [16], Chao et al. in [2]) and metaheuristics (see Cordeau et al. in [7], Vianna et al. in [17], and Hemmelmayr et al. in [11]) were used to solve this problem. All those methods were considered as multilevel combinatorial optimization problem, with the exception of Beltrami and Bodin procedures and the memetic algorithms [12]. In the first level an appropriate day is allocated to every task and the second level is dedicated to the solution of the classic VRP for the subset of tasks by taking into account every day of $H$.

The Ant Colony Optimization (ACO) is a metaheuristic succesfully used for several combinatorial problems. Its principle is inspired from the behavior of ants who cooperate in search of food. The ACO method is applied to the PVRP [14] using two phases. The first one is the construction of a good infeasible solution to PVRP by ACO model, and the second phase is devoted to convert this solution into a feasible one. Another technique for using of ACO algorithm on the Bin packing problem with two dimensions (2BP/O/G) was presented by Yalaoui and Chu in [18] and by Levine and Ducatelle in [13]. The principle of this technique is the combination of ACO method and the SHF-FF heuristic (Ben Messaoud et al. 2003). The ACO algorithm is used to choose the object candidate to put it in the truck and the heuristic is used to place the object. We propose a similar hybrid approach which combines the ACO method to choose such a task with an insertion method, called Insertion In Best Combination IIBC. This insertion heuristic combines the idea for insertion tasks in the days service combination and the idea of choosing the task for insertion (Chu et al. in [4], Lacomme et al. in [12], ...).

The remainder of this paper is organised as follows: In Section 2, we describe the PMCARP problem and the notations used in this work. The Section 3 is devoted to the presentation of our insertion heuristic IIBC. A numerical analysis and experimental results are shown in Section 5. The general conclusions of our work will be presented in the Section 6.

2 Problem description

The academic CARP problem focuses on servicing edges of an undirected network graph. Our goal in this paper is to solve the periodic CARP problem based on more realistic network $G$, with the following three points: (1) $G$ is a mixed graph, comprises two types of links, arcs and edges; (2) two distinct costs per link, i.e. a deadheading cost (whether the link is traversed without treatment) and a treatment cost; (3) windy edge: if at least one of these two costs depends on the direction. Consider a network $G = (N, E \cup A)$, where $N$ is a vertex set including a depot (node 0), $E$ is a set of edges $(i,j)$ $(i, j \in N)$ including a subset $E_r$ of required edges and $A$ is a set of arcs $(i,j)$ $(i, j \in N)$ including a subset $A_r$ of required arcs. A vehicle fleet is given, all vehicles have identical capacity $Q$ and are stationed at the depot. Each link $(i,j) \in E \cup A$ has a deadheading cost $c_{ij}$ and a demand $d_{ij} \geq 0$. We call each link in $R = E_r \cup A_r = \{(i,j) \in E \cup A; d_{ij} > 0\}$ by task, where $|R| = m_r$ and each task $(i,j)$ has a treatment cost $w_{ij}$ ($w_{ij} \neq w_{ji}$ if $(i,j) \in E_c$).

We transforme $G$ into a graph $\Gamma = (V, L)$, where each edge is replaced by two opposite arcs. $V$ is a novel set of nodes remplaced $E \cup A$ and we add a fictitious arc 0 around node 0 which represents the depot. $V$ is composed of two other sets $V_a$ and $V_c$, where $V_a$ is a set of $a_r$ required arcs identified by indexes from 1 to $a_r$ and $V_c$ is a set of $2 * e_r$ arcs (the two directions of each required edge) identified by indexes from $a_r + 1$ to $a_r + 2 * e_r$. For any task $p$ in $\Gamma$ $(|V| = r = a_r + 2 * e_r)$, we denote by $inv(p) = \bar{p}$ its inverse such as $\bar{p} = 0$ if and only if $p \in V_a$, $\bar{p} \neq 0$ if and only if $p \in V_c$ (if $p \in V_c$ then $\bar{p} \in V_c$ and
3. The proposed heuristic II BC: Insertion In Best Combination

This heuristic was used by Lacomme et al. [12], Chu et al. [4], etc. on the PCARP problem. They used a criteria ChooseTask to choose each task to be included in a solution as a descending order of frequency. In our heuristic II BC, the ChooseTask procedure used is very similar to that used by Paletta in [15] and by Bertazzi et al. in [1] on the PTSP problem (Periodic Traveling Salesman Problem). It makes a task \( p \) chosen for insertion, independently of the choice of combination by the following rule:

\[
p = \arg\max_{p \in \text{NO}} (f_p \cdot \text{Min}_{q \notin \text{NO}} (D(p, q) + w_p))
\]

where \( \text{NO} \) is the set of tasks in \( V \) and which are not yet included in a current solution \( S \). We represent a current solution \( S \) of PMCARP problem by \( S = (S_1, ..., S_{n_d}) \), where \( S_d \) is a sub-solution of \( S \) that represents a route associated to day \( d \). After choosing a task \( p \) by ChooseTask, the procedure ChooseCombination(\( p \)) is capable to make a combination \( k \in C_p \). This procedure was used in [12], [4], etc... on the indirected PCARP problem. Then, we insert \( p \) in current solution by InsertionTask(\( p, k, S \)) procedure. In the last step, we apply the Split() procedure made by Lacomme et al. in [12] on each sub-solution \( S_d \) to get the different trips of each day \( d \) in \( H \) and then a feasible solution \( S \) of PMCARP problem. The general algorithm used for this heuristic II BC is explained in Algorithm 1.

Each task existing in solution \( S \) is represented by \( S_{d,i} \) where \( i \) is its position in the sub-solution \( S_d \). Then, the best cost of inserting \( p \) in position \( i \) in day \( d \) of \( k \) is:

\[
I_{pdk} = \text{Min}_{i \in \{1, ..., |S_{d,-1}|-1\}} (D(S_{d,i-1}, p) + D(p, S_{d,i}) + w_p - D(S_{d,i-1}, S_{d,i}))
\]

where \( I_{pdk} \) is the best insertion cost of \( p \) in \( k \), such as \( I_{pdk} = \sum_{e \in k} \text{Min} \{ I_{pdk}, I_{pek} \} \) if \( p \in V_c \) and \( I_{pdk} \) otherwise. The existing of \( w_p \) in (1) and (2) is justified to treat the case of windy edge. Depending on the situation of \( p \) we have two forms of procedure ChooseCombination(), the first is the case if \( p \in V_c \) and the second case is done where \( p \in V_a \). The following algorithm illustrates the case where \( p \in V_c \) (in the other form, we remove the vector of directions \( VS \) and the cost \( I_{pek} \)).

**Procedure ChooseCombination(\( p \)):**

1. Let \( VP \) and \( VS \) respectively a vector of positions and directions of \( p \), of size \( f_p \).
   Let \( k^* \) which will be contain the value of combination to make, \( I = \text{infini} \).
2. For all \( k \in C_p \) repeat:
   a) \( I_{pk} = 0 \), \( \text{counter} = 0 \)
   b) for all \( d \in k \) repeat:
      find the best direction and position of \( \{ p, \overline{p} \} \) in \( d \),
      \( I_{pk} = I_{pk} + \text{Min} \{ I_{pdk}, I_{pek} \}, \text{counter} = \text{counter} + 1 \)
   c) if \( I_{pk} < I \) then
      \( I = I_{pk}, k^* = k \),
      update \( VP \) and \( VS \) by the values of the latest and best positions and directions of \( p \).
3. End ChooseCombination(\( p \)) and recovery \( k^* \).
After that we apply \textit{InsertionTask}(p, k, S) procedure, we insert a task \( p \) chosen by \textit{ChooseTask} in the combination \( k^* \) of \( C_p \), i.e. in the \( f_p \) days composing \( k \). Then for each \( d \in k^* \) we insert \( VS[i] \) in \( VP[i] \) of \( S_d \) and we increment \( i \) from 0 to \( f_p - 1 \) with \( d \) which follows an ascending order. We call a null solution, a solution that contains only the deposit in the begin and the end of each sub-solution.

\begin{algorithm}
1. \( NO \leftarrow V, S = (S_1, ..., S_{n_d}) \) a null solution, \( counter = 0 \)
2. While \( counter < m_r \) repeat:
   a) \( p \leftarrow \text{ChooseTask} \),
   b) \( k^* \leftarrow \text{ChooseCombination}(p) \),
   c) Apply \textit{InsertionTask}(p, k^*, S),
   d) \( NO \leftarrow NO - \{p, \} \) if \( p \in V_c \) and \( NO \leftarrow NO - \{p\} \) if \( p \in V_a \),
   e) \( counter = counter + 1 \),
3. Apply \textit{Split}(S) and recovery \( S \) the final feasible solution of PMCARP problem.
\end{algorithm}

4 The proposed hybrid approach \textit{ACOBC} to \textit{PMCARP} problem

In this approach, the PMCARP is divided into two phases: choose tasks order phase and insertion tasks phase. We propose an hybrid approach based on the Ant Colony Optimization ACO and an insertion method to solve the PCARP problem. The ACO is a metaheuristic proposed by Colorini and Dorigo ([6], 1992) to solve the travelling salesman problem. It is inspired by the ability of ants to find the shortest path between their nests and food sources. Naturally, ants deposited chemical quantity, called pheromones, on the tracks they borrowed. Pheromones are visible by ants and therefore more a road is used by ants there are more deposited pheromones and then it becomes more attractive to other ants. The ACO is used to find the order to insert the tasks in a current solution, and the \textit{IIBC} heuristic is used to insert each task choosed by ACO in this current solution.

We propose a colony of \( n_f \) ants where each ant \( f \) represents the order of tasks, called \textit{sequencef}, which will be included in the associated solution \( S_f \). We note by \( \tau_{i,j} \) the amount of pheromone filed between two tasks \( i \) and \( j \), \( \eta_{i,j} \) is the visibility measure which is the inverse of the travel cost between \( i \) and \( j \). At the iteration 0 of ACOBC algorithm (see \textbf{Algorithm 2}), we have \( \tau_{i,j} = 1/C(S_0) \) for all \((i,j) \in V \times V \) where \( S_0 \) and \( C(S_0) \) are respectively the initial solution calculated by \textit{IIBC} heuristic and the associated cost.

The first phase is called at each iteration, where each ant \( f \) is positioned on a task randomly and it constructs \textit{sequencef} by successively choosing a task to insert, continuing until each task has been in \textit{sequencef}. \( f \) chooses the task \( j \) after \( i \) from a set \( NO_i \) which contains the tasks that are not yet in \textit{sequencef}, using the following rule of probability:

\[
 j = \begin{cases} 
 \arg\max_{J \in NO_f} \left[ \tau_{i,j}(t) \right]^\alpha \left[ \eta_{i,j} \right]^\beta & \text{if } q \le q_0 \\ 
 J & \text{otherwise} 
\end{cases}
\]  

(3)

With \( 0 < q, q_0 < 1 \) and \( J \) is a task calculated by the following standard probability:

\[
p^f_{i,j}(t) = \begin{cases} 
 \frac{\left[ \tau_{i,j}(t) \right]^\alpha \left[ \eta_{i,j} \right]^\beta}{\sum_{i \in NO_f} \left[ \tau_{i,j}(t) \right]^\alpha \left[ \eta_{i,j} \right]^\beta} & \text{if } j \in NO_f \\
 0 & \text{otherwise} 
\end{cases}
\]

After the construction of all \textit{sequencef} the second phase is called. For each task \( p \) in \textit{sequencef} (by order of \textit{sequencef}) we will obtain a combination \( k \) using \textit{ChooseCombination}(p) procedure and we apply \textit{InserterTask}(p,k,\textit{Sf}). After that, we will obtain \( n_f \) giants solution \( S_f \) (capacity constraints not satisfied on each day of \( H \)).

At each iteration, the \textit{Improvement} process (local search) is applied. With a probability \( p_r \), we improve a solution \( S \) obtained by an ant using five procedures: 1) Change each task \( p \in V_c \) by its inverse \( p \); 2) Change the current combination for each task \( p \) by another combination in \( C_p \); 3) Exchange
combinations of two tasks $p$ and $q$ who have the same frequency and two different combinations used by them in $S_f$. 4) Exchange positions tasks $p$ and $q$ who are in the same $S_d$ of $S$ for each $d \in H$ and finally 5) Remove a task $p$ from its position and put it elsewhere in the same $S_d$ of $S$ and for every day $d \in H$.

Then, we apply the Split() procedure of Lacomme et al. [12] to obtain $n_f$ feasible solutions and the cost of each solution $S_f$ is calculated by the same procedure. Finally, there is the pheromone update by (4). Suppose that we have $u^{f}_{i,j}(t) = 1$ if $(i, j)$ is in sequence $f$ at iteration $t$ and equal to 0 otherwise, then the rule for global updating pheromones is given as follows:

$$
\tau_{(i,j)}(t+1) \leftarrow \rho \tau_{(i,j)}(t) + \sum_{f=1}^{n_f} \Delta \tau_{(i,j)}^{f}(t) \quad \text{with} \quad \Delta \tau_{(i,j)}^{f} = u^{f}_{i,j}(t) \frac{1}{C_f} 
$$

where $\rho$ is the evaporation rate of pheromones ($0 < \rho < 1$), $n_c$ represents the number of elitist ants and $C_f$ is the cost of obtained solution $S_f$ using sequence $f$ for insertion phase. The second term represents the amount of pheromones added to the next iteration between tasks for the $n_c$ best ants.

**Algorithm 2.** Outline of the ACOBC method.

1. Find $S_0 \leftarrow S_{test}$ an initial solution by IIBC heuristic, $C_{best} \leftarrow C(S_0)$ its cost, Initialize $\tau(p,q) = 1/C_{best} \forall (p,q) \in L$, $n_f$ empty ants, $iter = 1$.
2. While $iter \leq iter_{max}$ repeat:
   a. for all $f = 1, ..., n_f$ do
      a.1 $NO^f \leftarrow V$, prepare a null solution $S_f$
      a.2 choose randomly $p$ by $k \leftarrow \text{ChooseCombination}(p)$ and apply $\text{InsererTask}(p,k,S_f)$,
      a.3 $NO^f \leftarrow NO^f - \{p, \overline{p}\}$ if $p \in V_e$ and $NO^f \leftarrow NO^f - \{p\}$ if $p \in V_a$
      a.4 while $NO^f$ is not empty
         a.4.1 choose a task $q$ for $f$ by (3)
         a.4.2 $k \leftarrow \text{ChooseCombination}(q)$ and apply $\text{InsertionTask}(q,k,S_f)$,
         a.4.3 $NO^f \leftarrow NO^f - \{q, \overline{q}\}$ if $q \in V_e$ and $NO^f \leftarrow NO^f - \{q\}$ if $q \in V_a$
      a.5 apply process Improvement
      a.6 apply Split($S_f$) and calculate $C(S_f)$
      a.7 if $C(S_f) < C_{best}$ then $S_{best} \leftarrow S_f$ and $C_{best} \leftarrow C(S_f)$
   b. do update pheromones by (4) after finding the $n_c$ best ants
3. End While and recovery $S_{best}$ the final solution achieved with the cost $C_{best}$.

So the ACO algorithm chooses each task to insert in a current solution and the IIBC heuristic chooses the combination and the suitable positions and directions of the task by ChooseCombination() and by InsertionTask() we insert it in each day and position selected in the current solution. The optimization of the order of insertion of tasks by IIBC is assured then by the Ant Colony.

## 5 Computational results

Our method has been implemented in C language and tested on a 1.4 GHz Pentium 4 with windows 98. Detailed results are given in Table 1, for the 23 periodic instances of Golden manufactured by Lacomme et al. (see [12], 2005) which contain basic graphs for the PCARP problem. To prove its effectiveness we compared his results with those of the better memetic methods, noted BestMA and presented in [12]. The parameters of the ACOBC for the problem are: $iter_{max} = 185$, $\alpha = \beta = 1$, $\rho = 0.9$, $n_f = 20$, $n_c = 10$, $q_0 = 0.5$, $p_r = 0.75$, $n_a = 5$ and $n_b = 24$ ($n_c$ is the total number of combinations). In Table 1., the columns Nom, $n$ and $n_a$ concern respectively a name of instance, a size of vertex set, and a sum of frequencies (number of task occurrences in a solution).

Three methods in [12] are applied on Golden instances of the PCARP problem which are called PMA, DMA and IPMA. The values in column BestMA in Table 1., are the best values obtained by all three methods and after changing all the parameters on each forum of these 23 instances of Golden, for example by increasing the number of cross-over applied. Then in terms of the costs achieved by our
solutions with the ACOBC method, we managed to beat the best method BestMA in 13 instances (the costs are marked in bold in the Table 1.). Note also that our method providing the initial solution is more efficient than the BIH method (see [12]) in most cases.

<table>
<thead>
<tr>
<th>Nom</th>
<th>n</th>
<th>e_r</th>
<th>Q</th>
<th>n_a</th>
<th>BIH</th>
<th>IIBC</th>
<th>BestMA</th>
<th>ACOBC</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>G_1</td>
<td>12</td>
<td>22</td>
<td>5</td>
<td>65</td>
<td>1109</td>
<td>1080</td>
<td>890</td>
<td>886</td>
<td>7</td>
</tr>
<tr>
<td>G_2</td>
<td>12</td>
<td>26</td>
<td>5</td>
<td>76</td>
<td>1339</td>
<td>1129</td>
<td>1030</td>
<td>1007</td>
<td>10</td>
</tr>
<tr>
<td>G_3</td>
<td>12</td>
<td>22</td>
<td>5</td>
<td>61</td>
<td>1147</td>
<td>926</td>
<td>794</td>
<td>757</td>
<td>7</td>
</tr>
<tr>
<td>G_4</td>
<td>11</td>
<td>19</td>
<td>5</td>
<td>52</td>
<td>1117</td>
<td>973</td>
<td>858</td>
<td>806</td>
<td>5</td>
</tr>
<tr>
<td>G_5</td>
<td>13</td>
<td>26</td>
<td>5</td>
<td>75</td>
<td>1454</td>
<td>1247</td>
<td>1097</td>
<td>1121</td>
<td>13</td>
</tr>
<tr>
<td>G_6</td>
<td>12</td>
<td>22</td>
<td>5</td>
<td>67</td>
<td>1282</td>
<td>1054</td>
<td>1001</td>
<td>979</td>
<td>7</td>
</tr>
<tr>
<td>G_7</td>
<td>12</td>
<td>22</td>
<td>5</td>
<td>65</td>
<td>1209</td>
<td>1162</td>
<td>905</td>
<td>909</td>
<td>7</td>
</tr>
<tr>
<td>G_8</td>
<td>27</td>
<td>46</td>
<td>27</td>
<td>143</td>
<td>1447</td>
<td>1370</td>
<td>1000</td>
<td>1206</td>
<td>43</td>
</tr>
<tr>
<td>G_9</td>
<td>27</td>
<td>51</td>
<td>27</td>
<td>155</td>
<td>1318</td>
<td>1194</td>
<td>934</td>
<td>1098</td>
<td>57</td>
</tr>
<tr>
<td>G_{10}</td>
<td>13</td>
<td>23</td>
<td>35</td>
<td>65</td>
<td>925</td>
<td>854</td>
<td>743</td>
<td>720</td>
<td>10</td>
</tr>
<tr>
<td>G_{11}</td>
<td>22</td>
<td>45</td>
<td>50</td>
<td>133</td>
<td>1501</td>
<td>1403</td>
<td>1172</td>
<td>1207</td>
<td>49</td>
</tr>
<tr>
<td>G_{12}</td>
<td>12</td>
<td>25</td>
<td>10</td>
<td>67</td>
<td>1669</td>
<td>1525</td>
<td>1218</td>
<td>1371</td>
<td>9</td>
</tr>
<tr>
<td>G_{13}</td>
<td>10</td>
<td>28</td>
<td>41</td>
<td>81</td>
<td>1813</td>
<td>1745</td>
<td>1597</td>
<td>1667</td>
<td>11</td>
</tr>
<tr>
<td>G_{14}</td>
<td>7</td>
<td>21</td>
<td>21</td>
<td>64</td>
<td>350</td>
<td>351</td>
<td>298</td>
<td>292</td>
<td>6</td>
</tr>
<tr>
<td>G_{15}</td>
<td>7</td>
<td>21</td>
<td>37</td>
<td>64</td>
<td>200</td>
<td>182</td>
<td>182</td>
<td>176</td>
<td>5</td>
</tr>
<tr>
<td>G_{16}</td>
<td>8</td>
<td>28</td>
<td>24</td>
<td>85</td>
<td>452</td>
<td>406</td>
<td>372</td>
<td>370</td>
<td>9</td>
</tr>
<tr>
<td>G_{17}</td>
<td>8</td>
<td>28</td>
<td>41</td>
<td>85</td>
<td>299</td>
<td>281</td>
<td>267</td>
<td>263</td>
<td>11</td>
</tr>
<tr>
<td>G_{18}</td>
<td>9</td>
<td>36</td>
<td>37</td>
<td>106</td>
<td>574</td>
<td>585</td>
<td>502</td>
<td>523</td>
<td>32</td>
</tr>
<tr>
<td>G_{19}</td>
<td>8</td>
<td>11</td>
<td>27</td>
<td>30</td>
<td>195</td>
<td>203</td>
<td>173</td>
<td>173</td>
<td>9</td>
</tr>
<tr>
<td>G_{20}</td>
<td>11</td>
<td>22</td>
<td>27</td>
<td>63</td>
<td>455</td>
<td>393</td>
<td>374</td>
<td>363</td>
<td>6</td>
</tr>
<tr>
<td>G_{21}</td>
<td>11</td>
<td>23</td>
<td>27</td>
<td>101</td>
<td>609</td>
<td>552</td>
<td>525</td>
<td>519</td>
<td>20</td>
</tr>
<tr>
<td>G_{22}</td>
<td>11</td>
<td>44</td>
<td>27</td>
<td>129</td>
<td>663</td>
<td>623</td>
<td>598</td>
<td>606</td>
<td>38</td>
</tr>
<tr>
<td>G_{23}</td>
<td>11</td>
<td>55</td>
<td>27</td>
<td>165</td>
<td>798</td>
<td>743</td>
<td>690</td>
<td>712</td>
<td>75</td>
</tr>
</tbody>
</table>

The running times require some remarks. The last column Time in the above table presents the execution time needed to obtain solutions by our method ACOBC (Time in second). The average of time on all instances required by ACOBC is almost 0.34 minutes. But the average time by the three methods genetic algorithm [12] are respectively: \(\text{Time(PMA)} = 2.98\) minutes, \(\text{Time(DMA)} = 4.70\) minutes and \(\text{Time(IPMA)} = 7.78\) minutes. The average for the method BestMA is almost 748.7 minutes and this is justified by the very large number of cross-over applied and the change of lots of other parameters. So our method is much faster, because in our simple method, we apply a certain number of times \(\text{iter}_{\text{max}}\) the two steps \(\text{ChooseCombination()}\) and \(\text{InsertionTask()}\) of the IIBC heuristic and the duration of each time is a lot smaller than a second.

Hence, our approach is very simple but very effective and quick. The robustness of ACOBC method is even assured, because unlike the BestMA method, we used the same parameters on all instances to get our results.

6 Conclusion

We proposed a new hybrid approach ACOBC based on the Ant Colony Optimization ACO combined with an insertion heuristic IIBC for the Periodic Capacitated Arc Routing Problem on a mixed network. The problem is interesting and the applied methodology looks appropriate. This paper shows that the ACOBC is able to provide high quality solutions to the PCARP problem with a very limited computing time and a very efficient manner. The presented results demonstrate the good performance, rapidity and robustness of the ACOBC compared to memetic algorithms approach BestMA of Lacomme et al. [12].

Our approach has found thirteen new best solutions compared to the solutions obtained in [12] on the 23 instances of Golden.
References


