Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 3201656, 12 pages
https://doi.org/10.1155/2019/3201656

Research Article

A New Ant Colony Optimization Algorithm to Solve the Periodic Capacitated Arc Routing Problem with Continuous Moves

Guilherme V. Batista,[1] **Cassius T. Scarpin**,[1] **José E. Pécora Jr.**,[1,2] **and Angel Ruiz**[2,3]

[1] *Research Group of Technology Applied to Optimization (GTAO), Federal University of Paraná (UFPR), Curitiba 81020-430, Brazil*
[2] *Interuniversity Research Centre on Enterprise Networks, Logistics and Transportations (CIRRELT), Québec G1V 0A6, Canada*
[3] *Department of Operations and Decisions Systems, Université Laval, Québec G1V 0A6, Canada*

Correspondence should be addressed to Guilherme V. Batista; guivbatista89@gmail.com

Received 14 September 2018; Revised 7 June 2019; Accepted 14 July 2019; Published 28 July 2019

Academic Editor: Konstantina Skouri

This paper describes a variant of the Periodic Capacitated Arc Routing Problem for inspections in a railroad network. Inspections are performed by vehicles over a time horizon on which some stretches need evaluation more frequently than others due to its use. Each car can evaluate one stretch per day without being attached to a depot; at each day, the shift may start and end at different locations. This characterizes the problem as the Periodic Capacitated Arc Routing Problem with Continuous Moves in which firstly the delays on attendances are minimized and, second, the displacement costs. We present a mathematical model and an Ant Colony Optimization algorithm to solve the problem. The use of a local search procedure and some principles of Granular Tabu Search is crucial for the algorithm's performance. The numerical results are promising, especially for critical situations where the arcs' needs are close to the total vehicles' capacity.

1. Introduction

Arc Routing Problems have high applicability and are widely studied; they consist of finding routes that cover streets, roadways, railways, roads in general. The Capacitated Arc Routing Problem (CARP), introduced in [1], is an essential problem of this class on which the creation of routes considers vehicles with a limited capacity that must collect or deliver demands. The CARP assists the decisions at the operational level, planning routes for only one period. If the problem involves a time horizon greater than one day, the complexity of the problem increases, and it becomes the Periodic Capacitated Arc Routing Problem (PCARP), first described in [2]. The PCARP is an upgrade of CARP considering multiple periods, involving tactical and operational decisions.

In this paper, we study a variant of the PCARP in the context of inspection and maintenance of railways. Homogenous vehicles with limited capacity are responsible for ensuring the good state of repair and the flux in the rail network. They travel over the rails making measures that may show any irregularity; this predictive job aims to prevent accidents due to wear and tear. In countries like Brazil, the railroad network is extensive and the workers need to travel with the vehicles during their shifts. At each day, they evaluate one stretch, and the car ends its turn at any city where the worker can rest. This characteristic is the same found in some pick-up and delivery problems [3], for example, in the maritime case where ships perform routes from city to city, or in the production and distribution of argon [4, 5]. In these problems, the moves are continuous because the vehicles do not have the obligation of returning to a depot. Thus, the inspection and maintenance of railroads are named Periodic Capacitated Arc Routing Problem with Continuous Moves (PCARP-CM).

Solving real-life problems is not an easy task even for commercial solvers. In the PCARP, exact algorithms usually prove optimal solutions only for limited and small instances [6, 7]. An alternative to larger instances is to tackle them using heuristic algorithms as Scatter Search [8], Memetic Algorithms [9–11], Adaptive Large Neighborhood Search [12], and Ant Colony Optimization (ACO) [13, 14]. The ACO has achieved good results for dynamic routing problems on which the costs may modify according to the moves [15]. The PCARP-CM is similar: it has demands updated at each period, which can interfere in the objective function. Provided that, our work presents an ACO Algorithm which achieves competitive results when compared to the result obtained by a commercial solver. It is a new approach of ACO that combines concepts of Granular Tabu Search [16] and Local Search.

The next section shows a brief literature review concerning the PCARP, its applications, and solving methods. The third section presents details concerning the PCARP-CM in the railroad inspection and its mathematical model. In the fourth section, the algorithm procedure of the ACO is described. Last but not least, the computational results demonstrate our algorithm performance. Finally, we show our conclusions.

2. Literature Review

The PCARP is an Operational Research problem that has been developed based on real-world applications. Besides introducing the PCARP as a general problem, earlier studies show applications in snow removal, on inspection of power lines, spreading herbicides in rails, and mainly in garbage collection [7, 9, 17]. The garbage collection is readily associated with some concepts: the streets are the arcs or links, the crossings are the nodes, there is a frequency during the week, the demand is the amount of garbage, and the vehicle's capacity is the maximum loading of waste allowed. The citizens produce garbage storing it in plastic bags or containers; the collects must be well planned considering that some regions may have higher garbage production than others, requiring more attendances.

Achieving the tactical decision's level is a valuable contribution of the PCARP for the scientific and business development because the problem covers a planning horizon composed of as many periods as needed. The periods must be a fraction of time. Conventional approaches worked with days and bound the number of routes that should compound the planning periods. Some developments set a combination of attending days to each arc [8, 9, 11, 13, 17, 18], others use time intervals [7], or just frequencies [6, 14] to generate solutions. Recent approaches have been discretizing the time in a way that the finishing time of one activity is the initial time for the next one [12, 19, 20].

The first proposed mathematical model appeared in [17], on which optimal solutions were reached for small instances with five periods and ten arcs. The resolution approach adopted used three heuristics: two insertion methods and one two-phase algorithm (cluster and routing). Usually, heuristics are the most viable option to solve the problem due to its NP-Hard nature. Some of these solving approaches are Evolutionary algorithms like the Memetic Algorithm [9–11]; Scatter Search [8]; two-phase methods [6]; Adaptative Large Neighborhood Search [12]; ACO [13, 14].

The context of surveillance, monitoring, and maintenance of roads [21] gave origin to the PCARP with Irregular Services [6], in this case, demand may have more than one pattern of frequency. When dealing with monitoring activities, minimizing the displacement costs may be irrelevant; on the other hand, monitoring as many times as possible may guarantee a safer road network. Other applications can even involve inventory constraints for vehicles responsible for watering open-pit mines [12, 20].

This research differs from previous works in both the modeling approach and the proposed solving method. The decisions consist of planning only a move for each car in each period, different from conventional approaches on which there is a complete route for each car in each period. Also, the frequency of attendances uses the time interval represented by periodicities. The main difference consists in the fact that the cars are not attached to a depot where they must start and finish their routes. Moreover, our ACO algorithm is also distinct from the ones already applied to PCARP: in [13] the ACO optimizes the order of tasks to be inserted into a solution, and in [14] the ACO is applied after the arc routing problem is transformed into a Vehicle Routing Problem.

3. Problem Definition

3.1. Inspecting Railroads. We are concerned by the planning of inspection and maintenance of railways where a set of homogenous vehicles with limited capacity travel over the rails searching for any irregularity. These inspections aim to prevent accidents due to wear and tear, guaranteeing the good state of repair and the flow in the rail network. We model this situation as a Periodic Capacitated Arc Routing Problem with Continuous Moves (PCARP-CM).

The railroad network is extensive in countries with continental dimensions, Brazil, for example, and the sequence of railroad stretches inspections must have proper planning over a time horizon. The vehicles must sweep vast paths daily at a slow velocity to achieve the examination accurately. Consequently, it is impracticable to return to a depot at the end of the working day. In classical problems, like garbage collection and winter gritting, the vehicles depart from a depot and cover a considerable number of streets with a fast-growing demand. Also, in most common cases, the vehicle capacity is related to the quantity of charge; nevertheless in our proposed model, the capacity is implicit in the displacements.

Each vehicle can perform one railroad stretch per day; a stretch is a direct link between two cities that offer the structure for the worker to rest. Work teams take turns of more than one day in this kind of service, and the vehicles never stop unless necessary. The length of each stretch is known a priori and is defined based on the capacity of the cars and on two variables that can differ from place to place: speed and traffic flow. Determining the starting and ending point of these stretches simplifies the modeling process of this problem concerning the vehicles capacity constraint.

A periodicity can represent the frequency of inspection of each stretch. It represents the maximum time interval required between two consecutive passages. This problem is cyclical, repeating from time to time, and occurs in a permanent regime. Thus, the routes must be planned over a limited time horizon that must be greater or equal than the maximum periodicity. On the last planned day, the same quantity of vehicles that had left each starting point must return to it. Consequently, the plan can be repeated as many times as necessary.

3.2. Mathematical Model. The formulation is based on an undirected graph $G = (X, E)$ composed of n nodes i representing the cities where the drivers could finish their daily turn and stay overnight, thus $X = \{1, 2, \ldots, n\}$. The stretches are represented by m edges, where $E = \{1, 2, \ldots, m\}$, which must be served by nk cars, $K = \{1, 2, \ldots, nk\}$. Each edge e is composed by a pair of nodes $x_{ij} = (i, j)$ or $[i, j]$; it may be traversed in two possible directions from i to j or from j to i and it has a cost $c_{ij} = c_{ji}$ related to the distance. The capacity of each car is to traverse one arc per day; this simplifies the model and fulfills the objectives of the problem.

The maximum number of periods in a row on which an arc must be attended at least once $MP(x_{ij})$ expresses the periodicity of each arc. For example, if an arc has a periodicity of 15 days, it must be attended at least once between the first and the fifteenth day, once between the second and the sixteenth, and so forth. Therefore, the time horizon to be planned must be greater or equal to the highest periodicity and it is composed by np periods p, $H = \{1, 2, \ldots, np\}$. The set S contains every edge that has a periodicity lower than np and the set R the edges with periodicity equals np, thus $E = R \cup S$. As this is a cyclical problem and the solution represents a permanent regime of work, after the last planned day the next moves must be the same as the ones from the first period. In general, for a period p greater than np, the move performed must be equivalent to the one from the day $p \bmod np$. No starting point is known a priori but the number of cars arriving in the last planned day at some point must be equal to the number of cars leaving this point in the first day.

To give more flexibility to the problem, a periodicity may be delayed increasing the objective function by a penalty cost of PU_{ij} per day of delay in the arc $[i, j]$. To avoid unnecessary moves and guarantee feasibility, the cars may stop at some scheduled point during how long necessary. After all, if the solution shows that it is necessary to stop at some point, it may be possible to perform the activities in less time than the time horizon. The following variables are used in the model:

$$\begin{aligned} x_{ijkp} &\begin{cases} 1, & \text{if the car } k \text{ goes from node } i \text{ to node to } j \text{ in the period } p \\ 0, & \text{otherwise} \end{cases} \\ pn_{ijp} &\begin{cases} 1, & \text{if the arc } (i,j) \text{ has not attended its periodicity in the period } p \\ 0, & \text{otherwise} \end{cases} \\ f_{ikp} &\begin{cases} 1, & \text{if the car } k \text{ stay stopped in the node } i \text{ during the day } p \\ 0, & \text{otherwise} \end{cases} \\ y_i &\quad \begin{matrix} \text{Represents the number of vehicles leaving the node } i \text{ in the day } 1 \\ \text{and arriving in the node } i \text{ in the day } np \end{matrix} \end{aligned} \tag{1}$$

The mathematical programming formulation is

$$\min \quad Z = \sum_{[i,j]\in E} \sum_{k=1}^{nk} \sum_{p=1}^{np} c_{ij} x_{ijkp} + \sum_{[i,j]\in E} \sum_{p=1}^{np} PU_{ij} pn_{ijp} \tag{2}$$

Subject to

$$\sum_{[i,j]\in E} x_{ijkp} + f_{jkp} = \sum_{[i,j]\in E} x_{jik,p+1} + f_{jk,p+1} \quad \forall j \in X,\ \forall k \in K,\ p = 1, \ldots, np-1 \tag{3}$$

$$\sum_{[i,j]\in E} \sum_{k=1}^{nk} x_{ijk1} + f_{ik1} = y_i \quad \forall i \tag{4}$$

$$\sum_{[i,j]\in E} \sum_{k=1}^{nk} x_{ijk,np} + f_{jk,np} = y_j \quad \forall j \tag{5}$$

$$\sum_{[i,j]\in E} x_{ijkp} + \sum_{[i,j]\in E} x_{jikp} + \sum_{i=1}^{n} f_{ikp} = 1 \quad \forall p \in H,\ \forall k \in K \tag{6}$$

$$\sum_{k=1}^{nk} \sum_{d=1}^{MP(i,j)} \left(x_{ijk,f(d+p-1)} + x_{jik,f(d+p-1)}\right) + pn_{ijp} \geq 1 \qquad \forall [i,j] \in S,\ \forall p \in H;\ f(r) = r,\ if\ r \leq np \text{ or } f(r) = (r) \bmod (np),\ if\ r > np \tag{7}$$

$$\sum_{k=1}^{nk} \sum_{p=1}^{np} \left(x_{ijkp} + x_{jikp}\right) \geq 1 \quad \forall [i,j] \in R \tag{8}$$

$$x_{ijkp}, pn_{ijp}, f_{ikp} \in \{0,1\} \quad \forall [i,j] \in E,\ \forall k \in K,\ \forall p \in H \qquad y_i \in N^+ \tag{9}$$

The objective function (2) minimizes the displacement costs and tries to avoid as many possible delays as possible during the time horizon, so $PU_{ij} \gg c_{ij}$. Constraints (3) assure the network flow conservation; in addition, there is the possibility for a vehicle to remain still at a node during a day. Equations (4) and (5) ensure that the number of cars leaving a node at the beginning of the first period is the same arriving at the end of the last one, respectively. Constraints (6) establish that a car must perform a move or remain still at some point for each planned day. Constraints (7) and (8) control the periodicity of each arc. In (7) all arcs that do not have the periodicity equal to the number of days in the time horizon have the possibility of delays reacting with a penalty in the objective function. It is important to notice that the index p in the variable x_{ijkp} cannot exceed the value of np, being converted by a correspondent value inside the time horizon; for example, at the day $np + 1$ the edges evaluated on the first period are performed again. Constraints (8) do not allow arcs with a periodicity equal to np to suffer delays, because these arcs would not be served. Finally, equations (9) define the domain of all the variables.

4. The Ant Colony Optimization

Construction algorithms generate feasible solutions to problems by iteratively adding components until achieving a complete solution. Usually, there is no backtracking in this kind of process, e.g., at each step of the Greedy Construction Heuristics the component that gives a maximum myopic benefit is added to the solution. Greedy algorithms frequently generate solutions with better quality than a random solution but have the disadvantage of generating a limited number of solutions; also, the early choices effect in the latter ones, causing poor moves near the final phase. ACO is an alternative to create diversified solutions.

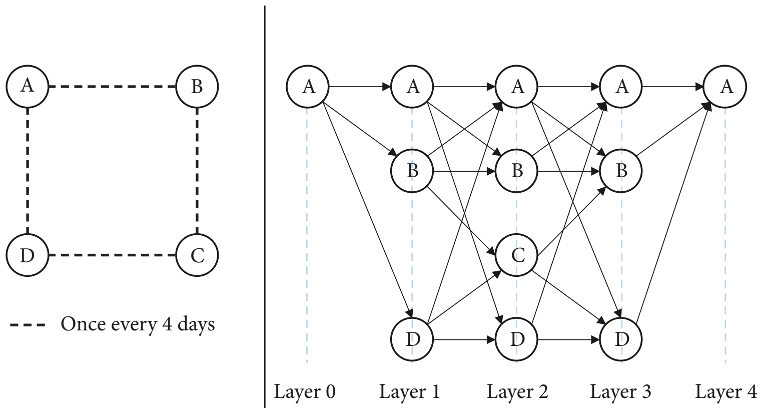
The behavior of real ants looking for food inspires the Ant Colony Optimization (ACO) algorithm. The ants spread pheromone during their search, creating a trail which influences the path choice of other ants. The algorithm uses numerical information to represent the pheromone trails and keeps information about the search experience. Each ant takes randomized decisions iteratively until achieving a complete solution; these decisions follow a probability that considers all information available at the moment.

Each artificial ant in ACO constructs a stochastic solution, conducting to a wide variety of different routes. The construction uses the information available at the moment and also the experience obtained with other solutions to add components, one by one, in the solution. If it is possible to create a procedure that generates a solution for a discrete optimization problem, then, in principle, it is possible to apply the ACO.

To increase the quality of the solutions produced by the ACO algorithm, it is possible to design hybrid ACO-local search algorithms. Local search algorithm improves a solution iteratively by looking on a defined neighborhood. During the search, if it finds a better solution, it replaces the current solution and the exploration moves to the new solution's neighborhood. The replacement can be done with the best solution found after exploring the whole neighborhood (best-improvement rule) or as soon as solution improving the best one so far is found (first-improvement rule). Needless to say, the choice of a good neighborhood is crucial for the algorithm's performance. The hybridization of a local search algorithm within the ACO scheme will be described in the subsection Improvement Phase.

4.1. The Problem Representation. In this paper, a solution is represented by a sequence of cities determining the starting and the ending city of each day's trip, so that the last city is the departure point for the next travel. Solutions with the same sequence of visits are symmetric and result at an equal value of the objective function, independently of the days and the direction on which the cities are visited. Generating reasonable solutions is not trivial for the PCARP-CM because of the continuity of the moves and the lack of a depot as a reference. Also, the cyclicity of the problem increases the number of symmetric solutions.

The ACO constructive algorithm proposed in this paper consists of finding a route into a directed graph. To this end, first an initial vertex is selected randomly but, in doing so, higher probability is given to the vertex having a higher



(a) Problem (b) Graph

Figure 1: An example problem (a) and its graph representation (b).

degree (number of edges connected). By choosing the initial vertex randomly, the constructive algorithm can generate a wider variety of solutions. Then, a graph is generated for each car based on layers; the initial layer contains just the possible vertex; the following layer includes all the adjacent vertices to the previous one, including itself. The process follows from layer to layer comprising all the possible connections with the past layer until achieving the final one. Not all the adjacent nodes are possible connections in all layers because of the cyclicity; the last layer will contain just the possible ending vertices based on the time distance. If only one car is available in the problem, the ending vertex must be the initial. If the problem comprises more than one car, it is necessary to evaluate the possibilities that respect the cycles during the time horizon. There will be $np + 1$ layers for each car. Figure 1 shows a toy problem and its associated graph.

Leaving a layer and going to the next one represents a move. The graph may give a false impression that the problem is simple, but after each move, the costs change dynamically because it may occur one delay. In periodical problems, the demands suffer influence by the period and the car's route. In our ACO algorithm, the ants start at the initial vertex and will probabilistically move until making a complete route for all the cars which, in other words, means going through all the layers. If in Figure 1, an ant moves from A in layer 0 to D in layer 1, which represents the car movement on the first day.

Defining the starting vertex helps the search in avoiding many symmetric solutions. To increase the ACO effectiveness, we enhanced it with some ideas of the Granular Tabu Search (GTS) [16]. The GTS is applied to optimization and graph-theoretic problems, and it uses what is named granular neighborhoods in a local search algorithm to reduce the computing time. In each iteration, the search is less myopic because it occurs among neighbors with desirable characteristics. For example, for the Vehicle Routing Problem, a sparse graph can be created to guide the search, it will contain just relevant arcs, e.g., with short distances.

For our ACO, we created two sparse graphs that avoid poor moves. The first one removes the arcs that maintain the car in the same node, for example, equivalent to removing the arcs A-A in Figure 1. The second will be modified at each move by eliminating some possibilities in specific days, the same as adding some arcs to a tabu list as follows. If an ant traverses an arc whose periodicity equals the time horizon, it will not be necessary to do it twice. Also, once an arc has been served, it does not seem reasonable to return to it soon, so we do not consider it for some time. Indeed, the arc is not considered for some periods corresponding to the arc periodicity divided by constant χ (we set $\chi = 1.5$ in our numerical experiments). Notice that although the arc should ideally be revisited precisely at the end of its periodicity, the constraints of the problem might force us to do it before. This process may block all the moves at some point, and when this happens, the algorithm will use just the first sparse graph, and if the problem continues, none are used.

4.2. The Ants Moves. All ants start in the initial node at the first layer and complete a sequence of decisions allowing them to go forward until the last node in the last layer, generating thus a complete solution. Each ant's move represents a car movement from one node to another on a given day. The process of choosing a move considers data of the actual status of every arc and the pheromone trails left by previous ants; each move option has a probability that uses this information. Iteratively, the addition of movements composes the solution.

Some possibilities may be more attractive in this process. For example, let us assume that at the end of the 7th planned day an ant is at city A and must decide to go to either city B or city C. The following information is also known: the arc A-C was traversed 6 days before and it has a periodicity of 30 days; the arc A-B did not receive attendance yet, what should occur in every 15 days. Based on this information, it seems more promising to go to city B. Therefore, it is possible to influence the ant's random choice by giving a higher probability to city B as the next visit. Such heuristic information is handled by a visibility factor η_{ij} which is computed for every move option by looking at the past moves. It quantifies how good the arc choice is by reseting the periodicity of the one traversed and approaching the other arcs to the deadline. This calculus considers $last_{ij}$ the last day when the edge (i, j) received attendance, d the day when the move is happening, and $g(ij)$ an auxiliary function that will help in the calculus:

$$g(ij) = MP(i, j) + \sum_{[k,l] \in E[i,j]} (last_{kl} + MP(k, l) - d) \tag{10}$$

$$\eta_{ij} = \begin{cases} g(ij) & if\ g(ij) > 0 \\ 1 & if\ g(ij) = 0 \\ -g(ij)^{-1} & if\ g(ij) < 0 \end{cases} \tag{11}$$

The higher the value of $g(ij)$ in (10), the more attractive becomes going from i to j. A negative value means that some periodicity, given by MP, is being violated. The visibility η_{ij} must be greater than zero in the algorithm. Therefore, some intervals were defined as in (11). This computation is required for all options that respect the two sparse graphs created and it must occur in every decision (for each day and car). The heuristic information brings specific knowledge for the problem. Also, the computation at runtime is important in dynamic problems [15].

The probability is also influenced by the experience accumulated with past routes using the pheromone trails left by ants. At the beginning, $t = 0$, the quantity of pheromones is low and the same for every arc. Running the algorithm, the ants spread pheromones in the used path, so each arc (i, j) traversed receives an amount of it. As time goes, the pheromone also evaporates. $\tau_{ijkp}(t)$ is the numerical information that represents the quantity of pheromone in the arc (i, j) using the car k in the day p in the iteration t. The more used an arc is, the more attractive it becomes for an ant. There are different ways to compute the probability of traversing the arc. The most widely used rule is the Ant Systems (AS) [22]. The following is based on it.

$$P_{ijkp}(t) = \frac{[\tau_{ijkp}(t)]^{\alpha} [\eta_{ij}]^{\beta}}{\sum_{j=1}^{n} [\tau_{ijkp}(t)]^{\alpha} [\eta_{ij}]^{\beta}} \tag{12}$$

The probability that an ant goes from a node i to another j using a car k during the day p in an iteration t, $P_{ijkp}(t)$, results from the pheromones and the heuristic information (visibility). It shows how attractive an arc is. Moreover, parameters α and β represent the relative importance given to the pheromones and to the heuristic information, respectively. A random draw decides the moves of each ant respecting the probabilities. In this process, the ant moves from layer to layer until completing the route for each car. If there are two or more cars, the algorithm generates a new graph for each one, with a new starting point when necessary, and the same quantity of layers to represent the moves on each day. If a vehicle initiates its route in a specific node and finishes it in the same node, the following car can select a new point to start. This selection evaluates the possibilities in the sparse graphs, where nodes with more possible adjacencies have more probability to be chosen. On the other hand, if a car finishes its route in a node different than the initial, the next car must start at this last node to guarantee the cyclicity of the problem.

4.3. Improvement Phase. It is common to hybridize the ACO with other heuristics aiming better performance. Once a complete solution is obtained, a Local Search (LS) algorithm can improve it. The two approaches are complementary: the ACO explores the solution space in a coarse-grained way and the LS refines the solution found. This combination may be crucial to achieving state-of-the-art performance. We developed a simple local search algorithm, which evaluates two neighborhoods and iteratively improves the solution with the best neighbor found among them, using a best-improvement rule.

The first neighborhood is the well-known 2-opt algorithm. It removes each pair of nonconsecutive arcs from the solution so that a reconstruction procedure is able to create a different route. Two new arcs link the free stretch inverting its sense. In this problem, this also means changing the days of attendance. This is replicated to each car. It is possible to mix two routes, but it increases the search space and implies an extra difficulty that may change the length of the routes, violating the capacity.

The second neighborhood consists of removing every three consecutive arcs and evaluating a better way to reconstruct the route. Removing it means to cease to cross two following cities in the solution, so the reconstruction determines the replacement of it by a different pair. It allows the elimination of unnecessary moves and the inclusion of an arc not attended. It also respects the starting and ending point of every route. If the ending point for the first car is the next car's starting one, then this point can be modified.

4.4. The Complete Algorithm. Let m be the total number of ants, each one will move from node to node with a probability measured by (12) respecting the sparse graphs when possible. When obtaining a complete solution, the local search algorithm improves it. All the visited arcs will receive an amount of pheromone laid by the ant. Furthermore, some part of the pheromones evaporates with a given rate. Thus, for the next iterations, the pheromones in each arc must be updated.

Each completed and improved ant's solution quantifies the amount of pheromone to be spread between the iterations t and $t + 1$. The objective function value is composed of penalties for delays and the displacement costs. As the delays must be avoided and are the priority in the model, we used the number of delays nd^s in the solution s as criteria to define this quantity. The displacement costs do not interfere in the decisions because they are secondary objectives.

$$\Delta\tau_{ijkp}^{s}(t, t+1) = \begin{cases} \dfrac{1}{nd^s} & if\ nd^s > 0,\ and\ the\ s^{th}\ solution\ uses\ the \\ & arc\ (i, j)\ with\ the\ car\ k\ in\ the\ day\ p \\ \dfrac{1}{\psi} & if\ nd^s = 0,\ and\ the\ s^{th}\ solution\ uses\ the \\ & arc\ (i, j)\ with\ the\ car\ k\ in\ the\ day\ p \\ 0 & otherwise \end{cases} \tag{13}$$

Equation (13) computes the variation of pheromones between iterations caused by the kth solution. A desirable solution

```
Algorithm: ACO for the PCARP-CM
1:  procedure ACO(cars, arcs, cities)
2:  Select an initial point
    Iteration $t = 1$
    Spread pheromones for the first iteration
3:  Do until reaching the Stopping Criteria
4:        For each ant // ($i = 1$ $to$ $m$)
5:            For each car // ($k = 1$ $to$ $K$)
6:                If car $\neq 1$ then evaluate the initial point end if
7:                Load possible ending points in the last day for car $k$
8:                Generate the graph for the problem
9:                For each day // ($d = 1$ $to$ $np$)
10:                   Computes the visibility $\eta_{ij}$ for each possible move
11:                   Draw the next city to be visited considering
                        the probabilities $P_{ijkp}(t)$ from the equation (12)
12:                   Update the sparse graphs
13:               Next day
14:           Nest car
15:           Apply the Local Search Procedure
16:           Save the solution if it is the best one
17:           Computes the quantity of pheromones to be spread by the ant $i$ in each
                arc, car and day $\Delta\tau^{s}_{ijkp}(t, t+1)$ using the equation (13)
18:       Next ant
19:       Update the pheromones using the variation and $\rho$ for $\tau_{ijkp}(t+1)$ in the
            equation (14)
            $t \longleftarrow t + 1$
20:   Loop
21:   Return the best solution found
```

Algorithm 1: The complete ACO algorithm for the PCARP-CM.

does not contain penalties; if there are delays, fewer pheromones are laid. On the other hand, more pheromones will attract more ants in the future. When no delays happen, we set $\psi = 0.1$ but could be of any value in $]0, 1]$. Supposing a solution for a problem where an arc should had received one attendance every 15 days and at some moment it had stayed 18 days without it, it had 3 days of delay nd^s. It is necessary to compute the delays of all arcs and accumulate it in the variable nd^s to decide how much pheromones to spread. The maximum amount happens when there are no delays.

The final amount of pheromone for the next iteration results from all m solutions and $\in$ $[0, 1]$, where $(1 - \rho)$ represents the evaporation coefficient, as shown in (14). It means that the next ants will probabilistically choose to move in an arc based on the contribution of every ant and in the historic of use.

$$\tau_{ijkp}(t+1) = \rho\tau_{ijkp}(t) + \sum_{s=1}^{m} \Delta\tau^{s}_{ijkp}(t, t+1) \quad (14)$$

Algorithm 1 provides a pseudocode of the complete algorithm. Firstly, the procedure receives all the information concerning the problem. The same initial point is for all ants; this might help in the convergence because it eliminates many symmetric solutions. Each ant performs its route, selecting moves for every car each day. The Local Search Algorithm improves the solution; if it finds the best objective value, then the Best Solution is updated. It is computed the quantity of pheromone to be spread; and after every ant completes its route, the pheromones are updated. The process repeats until achieving the stopping criteria, which may be a time limit or a determined number of iterations.

5. Computational Results

No benchmark instances are available for this problem, so we generated new ones based on the 23 *gdb* for the capacitated arc routing problem proposed by Golden [1]. These instances were also transformed in [8, 9, 12, 17, 20, 23], originating the *pgdb* instances. They have been modified to suit the PCARP-CM features, and they are labeled *pgdbcm*. Since the PCARP-CM is quite different from the others, we just kept the graph structure, the number of nodes, and the edges with its costs from the original instances. The number of different periodicities could variate from two to four; the number of cars oscillates from one to six. Each arc received a coherent periodicity considering: the graph, frequencies, and the number of vehicles. The time horizon was defined as the greatest periodicity. After all those transformations, we duplicated the instances in which the greatest periodicity was not equal to the least common multiple among the periodicities. This last process changed some periodicities and the time horizon to values that were multiple, resulting in the instances from *pgdbcm*24 to *pgdbcm*39. It allowed us to have a more significant diversity of problems, resulting

Table 1: PCARP-CM characteristics and the result obtained with the mathematical model.

Problem	Nodes	Arcs	Nb. of Periodic.	Cars	Req.	Time Horizon	Incumb. Solution	Lower Bound	GAP
*pgdbcm*1	12	22	2	1	32	38	19437,0	410,9	97,9%
*pgdbcm*2	12	26	3	1	44	53	39576,0	510,3	98,7%
*pgdbcm*3	12	22	2	2	31	19	344,0	315,0	8,4%
*pgdbcm*4	11	19	3	2	34	21	8525,0	447,0	94,8%
*pgdbcm*5	13	26	4	2	53	32	14755,0	659,0	95,5%
*pgdbcm*6	12	22	2	3	34	14	456,0	418,0	8,3%
*pgdbcm*7	12	22	3	3	37	45	569,0	491,0	13,7%
*pgdbcm*8	27	46	4	3	67	27	75367,0	325,0	100,0%
*pgdbcm*9	27	51	3	4	78	24	70456,0	439,0	99,4%
*pgdbcm*10	12	25	4	4	70	21	4791,0	684,0	85,7%
*pgdbcm*11	22	45	2	1	50	60	-	411,4	-
*pgdbcm*12	13	23	3	1	33	40	545,0	469,5	13,8%
*pgdbcm*13	10	28	2	2	33	20	620,0∗	620,0	0,0%
*pgdbcm*14	7	21	3	2	32	20	145,0∗	145,0	0,0%
*pgdbcm*15	7	21	4	2	48	29	144,0	137,0	4,9%
*pgdbcm*16	8	28	2	3	40	24	182,0	176,0	3,3%
*pgdbcm*17	8	28	3	3	52	21	175,0	161,0	8,0%
*pgdbcm*18	9	36	4	3	81	33	9410,0	369,0	96,1%
*pgdbcm*19	8	11	3	4	29	12	125,0∗	125,0	0,0%
*pgdbcm*20	11	22	4	4	53	16	14265,0	244,0	98,3%
*pgdbcm*21	11	33	2	5	46	12	212,0	204,0	3,8%
*pgdbcm*22	11	44	4	5	88	22	482,0	423,0	12,2%
*pgdbcm*23	11	55	2	6	82	17	340,0	329,0	3,2%
*pgdbcm*24	12	26	3	1	44	54	8586,0	517,6	94,0%
*pgdbcm*25	12	22	2	2	31	20	348,0	315,0	9,5%
*pgdbcm*26	11	19	3	2	34	24	548,0	447,0	18,4%
*pgdbcm*27	13	26	4	2	50	32	723,0	639,0	11,6%
*pgdbcm*28	12	22	3	3	33	16	444,0	402,0	9,5%
*pgdbcm*29	27	46	4	3	63	30	2405,0	304,0	87,4%
*pgdbcm*30	27	51	3	4	78	24	46441,0	375,0	99,2%
*pgdbcm*31	12	25	4	4	70	24	4838,0	684,0	85,9%
*pgdbcm*32	13	23	3	1	33	42	527,0	474,0	10,1%
*pgdbcm*33	7	21	3	2	36	20	157,0	156,0	0,6%
*pgdbcm*34	7	21	4	2	42	30	120,0	117,0	2,5%
*pgdbcm*35	8	28	3	3	54	24	174,0	167,0	4,0%
*pgdbcm*36	9	36	4	3	81	36	383,0	369,0	3,7%
*pgdbcm*37	11	22	4	4	53	36	1284,0	246,0	80,8%
*pgdbcm*38	11	44	4	5	88	24	427,0	423,0	0,9%
*pgdbcm*39	11	55	2	6	82	18	338,0	329,0	2,7%

in 16 new instances. The time horizon being a multiple of all periodicities is recurrent in the real context but not mandatory.

We compared our algorithm to sub-optimal or optimal results produced by the linear programming model. The solver Gurobi 7.0.1 64 bits, with a time limit of one hour and the default parameters, was used for this comparison. All algorithms were implemented in Visual Basic.NET. Table 1 lists all the 39 instances with their respective characteristics and the results. The first column shows the name of the problem, and the next ones present the number of nodes, arcs, periodicities, cars, requirements, and time horizon. The number of requirements represents the smallest number of needed passages over the graph during the time horizon. The last three columns show the answer obtained with the solver after the time limit. Firstly, there is the incumbent

Table 2: Results of the tests to define parameters for the ACO Algorithm for the PCARP-CM.

α; β; ρ	Average	Standard Deviation	α; β; ρ	Average	Standard Deviation	α; β; ρ	Average	Standard Deviation
0.5;0.5;0.5	15859.2	13475.7	1;3;0.7	14156.6	12647.3	2;1;0.9	16600.6	13848.6
0.5;0.5;0.7	16183.0	13854.6	1;3;0.9	15025.6	13005.7	2;2;0.5	18858.3	15455.2
0.5;0.5;0.9	16865.6	14362.9	1;5;0.5	13867.7	12361.2	2;2;0.7	18088.9	14983.7
0.5;1;0.5	16338.2	13864.6	1;5;0.7	14220.6	12677.9	2;2;0.9	16895.0	14258.3
0.5;1;0.7	16366.0	13904.9	1;5;0.9	15311.4	13212.6	2;3;0.5	18889.3	15366.4
0.5;1;0.9	16448.7	13743.4	1.2;0.5;0.5	14694.5	12995.8	2;3;0.7	18308.4	15481.3
0.5;2;0.5	16179.6	13586.8	1.2;0.5;0.7	14017.7	12392.5	2;3;0.9	16936.0	13945.6
0.5;2;0.7	16568.2	13880.8	1.2;0.5;0.9	13964.2	12529.1	2;5;0.5	19562.6	16174.2
0.5;2;0.9	16399.8	13921.1	1.2;1;0.5	14887.4	13169.0	2;5;0.7	18496.9	14940.5
0.5;3;0.5	16186.0	13569.8	1.2;1;0.7	13984.9	12547.5	2;5;0.9	16932.2	13884.1
0.5;3;0.7	16282.8	13841.7	1.2;1;0.9	14585.2	13018.0	5;0.5;0.5	22404.5	18139.3
0.5;3;0.9	16575.3	14138.1	1.2;2;0.5	14826.1	12995.7	5;0.5;0.7	21988.9	17097.7
0.5;5;0.5	16400.4	13687.8	1.2;2;0.7	14220.1	12561.7	5;0.5;0.9	21437.5	16956.4
0.5;5;0.7	16543.2	14035.1	1.2;2;0.9	14482.6	12947.9	5;1;0.5	22394.5	18020.2
0.5;5;0.9	16782.8	13877.7	1.2;3;0.5	14689.6	13037.8	5;1;0.7	22074.8	17453.2
1;0.5;0.5	13692.0	12598.6	1.2;3;0.7	14149.5	12313.9	5;1;0.9	21361.2	16944.4
1;0.5;0.7	14824.3	13243.7	1.2;3;0.9	14387.4	12872.5	5;2;0.5	22379.4	17334.7
1;0.5;0.9	14973.1	13057.2	1.2;5;0.5	14964.4	13208.6	5;2;0.7	21950.8	16971.1
1;1;0.5	**13218.2**	12314.9	1.2;5;0.7	14426.6	12720.8	5;2;0.9	21665.8	17300.3
1;1;0.7	14019.0	12449.8	1.2;5;0.9	14424.1	12987.1	5;3;0.5	22677.1	17584.1
1;1;0.9	15032.9	13283.5	2;0.5;0.5	19080.2	15207.4	5;3;0.7	21998.5	17490.6
1;2;0.5	13334.1	11985.9	2;0.5;0.7	18460.9	15324.0	5;3;0.9	21537.6	16456.3
1;2;0.7	14036.5	12858.2	2;0.5;0.9	16795.2	13861.4	5;5;0.5	22644.0	17977.5
1;2;0.9	15299.7	13331.7	2;1;0.5	19165.3	15803.9	5;5;0.7	21955.4	16996.4
1;3;0.5	13567.2	11982.5	2;1;0.7	18345.1	15311.8	5;5;0.9	21402.6	16361.2

solution, then the lower bound, and, finally, the gap between the incumbent solution and the lower bound.

The scalability of penalties in the objective function leads a single delay to provoke large gaps. The value of each penalty is of 1000 while the displacement costs are of an average of 7.1 with 6.49 of standard deviation for each edge. Three problems, *pgdbcm*13, *pgdbcm*14, and *pgdbcm*19, have the optimal value proved, and the others were stopped after 3600 seconds. For problem *pgdbcm*11, this time limit was not enough to find a feasible solution. So, we used the same time limit for our ACO algorithm.

5.1. Parameter Tunning. The ACO uses parameters that concern the relative importance given to the pheromone trails $\alpha \geq 0$, and to the visibility or heuristic information $\beta \geq 0$ combined with the coefficient $0 \leq \rho \leq 1$ which determines the evaporation. To determine them, we ran 29250 tests combining some promising values used in [22, 24] that were $\alpha = \{0.5, 1, 1.2, 2, 5\}$, $\beta = \{0.5, 1, 2, 3, 5\}$, and $\rho = \{0.5, 0.7, 0.9\}$. A large number of ants allow the exploration of more solutions and more information to update the pheromones, so we decide to use $m = 100$ ants. Furthermore, for these tests, we used the stopping criteria, on which the number of iterations is equal to 100. The Local Search procedure may take a long time improving the solution when the initial result is poor. So, we removed it from the algorithm aiming to find one combination of parameters that brings more productive solutions. The results of these tests are available in Table 2, which brings the average and the standard deviation value for all tests.

The results presented in Table 2 are the average value of the objective function for all instances tested ten times for each particular combination of parameters with the standard deviation. It means that each combination of parameters was tested 390 times, and each instance was solved 750 times. Each test took an average of 32 seconds to be completed. Hence, we could evaluate proper parameters considering the variability inherent to the algorithm. The α, β, and ρ parameters resulting from the test were 1, 1, and 0.5, respectively.

5.2. Algorithm Results. Table 3 shows the results obtained compared with the ones from the solver. The first column shows the problem name and the three following ones, the results from the solver. The fourth column starts the results of the ACO Algorithm presenting first the objective value achieved, followed by the number of iterations, time spent, and the gap relative to the lower bound obtained in the solver. Finally, the last column shows the best result found so far, also in bold, and the last row some average values.

We set a time limit of 3600 seconds to have a fair comparison; this time is more than the necessary to have many iterations when the Local Search procedure is not present. Unfortunately, not all problems have good solutions before

Table 3: Results obtained with the ACO algorithm for the PCARP-CM.

Problem	Solver			ACO Algorithm				Best
	Incumb. Solution	Lower Bound	GAP	Objective Value	Iterations	Time (seconds)	GAP	
*pgdbcm*1	19437,0	410,9	97,9%	**451,0**	**312**	**3606**	**8,9%**	451,0
*pgdbcm*2	39576,0	510,3	98,7%	**3594,0**	**10**	**3680**	**85,8%**	3594,0
*pgdbcm*3	**344,0**	**315,0**	**8,4%**	**344,0**	**89**	**3628**	**8,4%**	344,0
*pgdbcm*4	8525,0	447,0	94,8%	**2519,0**	**44**	**3644**	**82,3%**	2519,0
*pgdbcm*5	14755,0	659,0	95,5%	**9764,0**	**12**	**3832**	**93,3%**	9764,0
*pgdbcm*6	**456,0**	**418,0**	**8,3%**	**456,0**	**312**	**3606**	**8,3%**	456,0
*pgdbcm*7	**569,0**	**491,0**	**13,7%**	576,0	32	3675	14,8%	569,0
*pgdbcm*8	75367,0	325,0	100,0%	**22364,0**	**4**	**3870**	**98,5%**	22364,0
*pgdbcm*9	70456,0	439,0	99,4%	**31446,0**	**3**	**4492**	**98,6%**	31446,0
*pgdbcm*10	4791,0	684,0	85,7%	**3801,0**	**4**	**3839**	**82,0%**	3801,0
*pgdbcm*11	-	411,4	-	**442,0**	**10**	**3737**	**6,9%**	442,0
*pgdbcm*12	545,0	469,5	13,8%	**536,0**	**306**	**3606**	**12,4%**	536,0
*pgdbcm*13	**620,0***	**620,0**	**0,0%**	**620,0**	**18**	**3704**	**0,0%**	620,0
*pgdbcm*14	**145,0***	**145,0**	**0,0%**	149,0	12	3742	2,7%	149,0
*pgdbcm*15	**144,0**	**137,0**	**4,9%**	1152,0	4	4010	88,1%	144,0
*pgdbcm*16	**182,0**	**176,0**	**3,3%**	204,0	2	5796	13,7%	182,0
*pgdbcm*17	**175,0**	**161,0**	**8,0%**	183,0	3	4963	12,0%	175,0
*pgdbcm*18	9410,0	369,0	96,1%	**2447,0**	**1**	**10322**	**84,9%**	2447,0
*pgdbcm*19	**125,0***	**125,0**	**0,0%**	131,0	45	3632	4,6%	131,0
*pgdbcm*20	14265,0	244,0	98,3%	**4278,0**	**8**	**3939**	**94,3%**	4278,0
*pgdbcm*21	**212,0**	**204,0**	**3,8%**	222,0	5	4013	8,1%	212,0
*pgdbcm*22	**482,0**	**423,0**	**12,2%**	1492,0	1	13001	71,6%	482,0
*pgdbcm*23	**340,0**	**329,0**	**3,2%**	357,0	1	18064	7,8%	340,0
*pgdbcm*24	8586,0	517,6	94,0%	**620,0**	**10**	**3927**	**16,5%**	620,0
*pgdbcm*25	**348,0**	**315,0**	**9,5%**	370,0	127	3612	14,9%	348,0
*pgdbcm*26	**548,0**	**447,0**	**18,4%**	573,0	39	3602	22,0%	548,0
*pgdbcm*27	**723,0**	**639,0**	**11,6%**	4789,0	11	3898	86,7%	723,0
*pgdbcm*28	**444,0**	**402,0**	**9,5%**	447,0	30	3674	10,1%	444,0
*pgdbcm*29	**2405,0**	**304,0**	**87,4%**	3415,0	3	4581	91,1%	2405,0
*pgdbcm*30	46441,0	375,0	99,2%	**25424,0**	**3**	**4625**	**98,5%**	25424,0
*pgdbcm*31	**4838,0**	**684,0**	**85,9%**	4906,0	3	4621	86,1%	4838,0
*pgdbcm*32	**527,0**	**474,0**	**10,1%**	536,0	208	3600	11,6%	527,0
*pgdbcm*33	**157,0**	**156,0**	**0,6%**	158,0	13	3638	1,3%	157,0
*pgdbcm*34	**120,0**	**117,0**	**2,5%**	131,0	4	4512	10,7%	120,0
*pgdbcm*35	**174,0**	**167,0**	**4,0%**	209,0	2	5145	20,1%	174,0
*pgdbcm*36	**383,0**	**369,0**	**3,7%**	2479,0	1	13780	85,1%	383,0
*pgdbcm*37	**1284,0**	**246,0**	**80,8%**	4284,0	8	3969	94,3%	1284,0
*pgdbcm*38	**427,0**	**423,0**	**0,9%**	525,0	1	16899	19,4%	427,0
*pgdbcm*39	**338,0**	**329,0**	**2,7%**	380,0	1	22137	13,4%	338,0
Average	9365,0		38,6%	3507,0			42,8%	3184,8

starting the local search and, as was said, the improvements might take a long time. To have a better idea of the algorithm performance, we let the algorithm free to finish its current iteration after reaching the time limit. By doing so, it may happen that some problems run only for one iteration.

On average, the objective values of the ACO Algorithm were better than the ones obtained by the solver, but the average gap calculated is a little worst. The lower bounds calculated do not include any delay until they reach the time limit; it means that just displacement costs have been

Table 4: Comparison between performances in tight cases.

Ratio $\left(\dfrac{Nb.\ of\ Requirements}{Nb.\ Cars \times Time\ horizon}\right)$	Number of Instances	Best performance		
		Gurobi	ACO	Draw
> 80%	20	4	**13**	3
≤ 80%	19	**19**	0	0

considered. In most cases, one delay causes gaps over than 70%, and more than four delays result in values greater than 90%, so this proportion does not reflect how good the solution that has priority to reduce delays is. Analyzing the relation of requirements and the number of moves available, computed by the number of cars times time horizon, 20 instances need to occupy at least more than 80% of the vehicles' capacity. These instances are some tight cases of difficult resolution. In these situations, the cars do not have much freedom to stop, and a single decision of move can make the difference between a good or a poor solution. Table 4 shows the results considering these cases. In 80% of them, our algorithm was better or equal than the solver. In the other situation, Gurobi was unbeatable, but the ACO achieved close results.

To evaluate other aspects of the ACO, we tested other strategies or parameters as follows: (1) we used 1000 ants, (2) the ants did not have the same initial node, or (3) the local search was done at the end of the algorithm in a pool of best solutions and randomly selected solutions instead of doing it for every ant. Those changes sped up the algorithm and increased the diversification. The behavior of the algorithm was similar to the first one proposed, having the best performance in 13 instances and tying in 3 when compared to the solver. The average value improved to 3053,64, but the GAP worsened to 45,4% on average. The ACO variability is high, so when comparing both algorithms we find that this alteration had an improvement of values in a few instances and less better solutions.

6. Conclusions and Future Work

In this paper, we proposed a mathematical model for the PCARP with continuous moves applied to the inspection of railroads. For this purpose, the foremost objective is to minimize the number of delays on each stretch. Another secondary aim is to reduce the displacement costs. Finding a feasible solution may be a difficult task depending on the problem characteristics.

The problem was tacked by an exact method using the mathematical model in a solver and a new Ant Colony Optimization Algorithm. Commercial solvers combine heuristics with exact methods and have shown an excellent performance for many problems, including the PCARP-CM. Our algorithm showed itself to be competitive, achieving better results in some cases. In summary, the ACO Algorithm showed advantages in critical situations where the demands are close to the vehicle's capacity. The Local Search is crucial for the algorithm's performance but can take an extended computational time depending on the solution obtained.

As the ACO and the solver produced good results in different situations, in the future, we will intend to hybridize them within a matheuristic scheme. This process may allow us to improve the final results and increase the software's efficiency.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by CAPES, Coordination for the Improvement of Higher Education Personnel, Brazil.

References

[1] B. Golden, J. Dearmon, and E. Baker, "Computational experiments with algorithms for a class of routing problems," *Computers & Operations Research*, vol. 10, no. 1, pp. 47–59, 1983.

[2] P. Lacomme, C. Prins, and W. Ramdane-Chérif, "Evolutionary algorithms for multiperiod arc routing problems," in *Proceedings of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based System*, pp. 1–8, 2002.

[3] H. Andersson, A. Hoff, M. Christiansen, G. Hasle, and A. Løkketangen, "Industrial aspects and literature survey: combined inventory management and routing," *Computers & Operations Research*, vol. 37, no. 9, pp. 1515–1536, 2010.

[4] M. Savelsbergh and J. Song, "Inventory routing with continuous moves," *Computers & Operations Research*, vol. 34, no. 6, pp. 1744–1763, 2007.

[5] M. Savelsbergh and J. Song, "An optimization algorithm for the inventory routing problem with continuous moves," *Computers & Operations Research*, vol. 35, no. 7, pp. 2266–2282, 2008.

[6] I. Monroy, C. Amaya, and A. Langevin, "The periodic capacitated arc routing problem with irregular services," *Discrete Applied Mathematics*, vol. 161, no. 4-5, pp. 691–701, 2013.

[7] F. Chu, N. Labadi, and C. Prins, "The Periodic Capacitated Arc Routing Problem linear programming model, metaheuristic and lower bounds," *Journal of Systems Science and Systems Engineering*, vol. 13, no. 4, pp. 423–435, 2004.

[8] F. Chu, N. Labadi, and C. Prins, "A Scatter Search for the periodic capacitated arc routing problem," *European Journal of Operational Research*, vol. 169, no. 2, pp. 586–605, 2006.

[9] P. Lacomme, C. Prins, and W. Ramdane-Chérif, "Evolutionary algorithms for periodic arc routing problems," *European Journal of Operational Research*, vol. 165, no. 2, pp. 535–553, 2005.

[10] Y. Zhang, Y. Mei, K. Tang, and K. Jiang, "Memetic algorithm with route decomposing for periodic capacitated arc routing problem," *Applied Soft Computing - Journal*, pp. 17–21, 2016.

[11] Y. Mei, K. Tang, and X. Yao, "A memetic algorithm for periodic capacitated arc routing problem," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 6, pp. 1654–1667, 2011.

[12] J.-P. Riquelme-Rodríguez, M. Gamache, and A. Langevin, "Adaptive Large Neighborhood Search for the Periodic Capacitated Arc Routing Problem with Inventory Constraints," *Networks*, vol. 47, no. 1, pp. 26–36, 2014.

[13] A. Kansou and A. Yassine, "Ant colony system for the periodic capacitated arc routing problem problem," in *Proceedings of the International Network Optimization Conference*, pp. 1–7, 2009.

[14] S. Huang and T. Lin, "Using Ant Colony Optimization to solve Periodic Arc Routing Problem with Refill Points," *Journal of Industrial and Production Engineering*, vol. 31, no. 7, pp. 37–41, 2014.

[15] M. Dorigo and T. Stützle, "Ant colony optimization: overview and recent advances," in *Handbook of Metaheuristics*, M. Gendreau and J.-Y. Potvin, Eds., vol. 272, pp. 227–263, Springer, 2nd edition, 2019.

[16] P. Toth and D. Vigo, "The granular tabu search and its application to the vehicle-routing problem," *INFORMS Journal on Computing*, vol. 15, no. 4, pp. 333–346, 2003.

[17] F. Chu, N. Labadi, and C. Prins, "Heuristics for the periodic capacitated arc routing problem," *Journal of Intelligent Manufacturing*, vol. 16, no. 2, pp. 243–251, 2005.

[18] Y. Zhang, Y. Mei, K. Tang, and K. Jiang, "Memetic algorithm with route decomposing for periodic capacitated arc routing problem," *Applied Soft Computing*, vol. 52, pp. 1130–1142, 2017.

[19] J. Riquelme-Rodríguez, M. Gamache, and A. Langevin, "Location arc routing problem with inventory constraints," *Computers & Operations Research*, vol. 76, pp. 84–94, 2016.

[20] J. Riquelme-Rodríguez, M. Gamache, and A. Langevin, "Periodic capacitated arc-routing problem with inventory constraints," *Journal of the Operational Research Society*, vol. 65, no. 12, pp. 1840–1852, 2017.

[21] F. Marzolf, M. Trépanier, and A. Langevin, "Road network monitoring: Algorithms and a case study," *Computers & Operations Research*, vol. 33, no. 12, pp. 3494–3507, 2006.

[22] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 1, pp. 1–13, 1996.

[23] F. Chu, N. Labadi, and C. Prins, "A scatter search for the periodic capacitated arc routing problem," in *Project Management and Scheduling (PMS)*, pp. 415–420, 2004.

[24] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed Optimization by Ant Colonies," in *Proceedings of Proceedings of ECAL91 - European Conference on Artificial Life*, pp. 132–142, January 1991.