RAIRO-Oper. Res. 53 (2019) 303–322
https://doi.org/10.1051/ro/2018087

RAIRO Operations Research
www.rairo-ro.org

EFFICIENT ALGORITHMS UNDER DYNAMIC GRAPHS TO SOLVE THE CAPACITATED ARC ROUTING PROBLEM WITH FEASIBLE SPARSE GRAPH

Sara Tfaili[1], Hamdi Dkhil[1], Abdelkader Sbihi[1,2,*] and Adnan Yassine[1,3]

Abstract. In this paper we develop several approaches to approximately solve the capacitated arc routing problem (CARP) on sparse graphs namely sparse CARP. First, we give a mathematical model for the sparse CARP and we present a brief survey about a transformation technique to transform the sparse CARP into a sparse capacitated vehicle routing problem namely sparse CVRP. Later, we propose several approaches to solve the sparse CARP by solving its equivalent obtained sparse CVRP. The first approach is a constructive heuristic (CH) used to construct an initial feasible solution. The second approach is an improving randomized procedure (IRP) used to improve the quality of the initial solution. Finally, we introduce the main adapted tabu search approach (TS) under a sparse dynamic graph. The algorithm starts by applying the first two procedures CH and IRP and attempts to compute a better solution for the sparse CARP. Extensive computational tests on randomly generated problem instances show the effectiveness of the proposed approach. The TS algorithm yields satisfactory results within reasonable computational time. The approach outperformed also the commercial solver Cplex v12.71 which was able to solve only small instances with relatively a big CPU time for medium size instances.

Mathematics Subject Classification. 90B06, 90C05, 90C10, 90C27, 90C59.

Received May 7, 2017. Accepted September 21, 2018.

1. Introduction

The routing problem is considered as one of the most important combinatorial optimization problems and integer programming due to its wide applications in many real-life situations. A first variant of the routing problem is the classical node routing problem or vehicle routing problem (VRP) which was first introduced by Dantzig and Ramser [10]. The VRP generally consists of determining a set of customers located on the nodes of the road network for some objective. The other variant is the arc routing problem (ARP) which was first introduced by Golden and Wong [20] and it consists of determining a set of arcs on a road network with a minimal cost to serve a set of customers located on the arcs of the network. Both variants generalize the well-known traveling salesman problem (TSP) [23] which was proved to be NP-Complete by Karp [28]. Due to the complexity of the ARP, and since VRP was more studied in the literature, then several transformation

Keywords. Routing problems, graph theory, sparsity, tabu search.

[1] Université Le Havre Normandie, LMAH, FR CNRS 3335, ISCN, 76600 Le Havre, France.
[2] Brest Business School (BBS), 29200 Brest, France.
[3] Université Le Havre Normandie, ISEL, 76600 Le Havre, France.
*Corresponding author: abdelkader.sbihi@brest-bs.com

techniques to transform ARP into VRP were developed so that a solution of the obtained VRP represents also a solution of the original ARP. In this paper, we are interested in solving a special capacitated arc routing problem that is the capacitated arc routing problem on sparse graphs.

As a major remark, one should not confuse between a sparse formulation of the CARP (*e.g.* see [3]) and a sparse CARP that we are considering in this work. Let $G = (V, E)$ be a graph over which we define a capacitated arc routing problem. A sparse formulation of the problem is a formulation whose number of variables is proportional to the number of edges or arcs of G and this was first introduced by Belenguer and Benavent [3]. On the other hand, a capacitated arc routing problem is said to be sparse if its graph or network G is sparse from graph theory point of view, *i.e.* a sparse graph is a graph which has a relatively small density which is defined to be the ratio of the number of actual edges of the graph to the maximum possible number of edges in the graph (density:$= \frac{2|E|}{|V|(|V|-1)}$). In our study, we deal with sparse capacitated arc routing problem such that the maximum degree held by any vertex of the network does not exceed 3, and with graphs whose densities range between 0.3 down to 0.001 and less. For more details and other definitions about sparsity, the reader may refer to [16, 34].

In this paper, we give a new mathematical model for the sparse CARP. We recall a transformation technique that we have proposed in a previous work [47, 48]. We give a first constructive heuristic noted CH to build an initial solution for the transformed sparse CARP that we improve by an improving randomized procedure noted IRP. Later we build a tabu search algorithm (TS) to solve the transformed sparse CARP. The computational results show that the approach was able to improve the quality of the solution and the encouraging results lead to develop future works on the sparse CARP.

The remainder of the paper is organized as follows: in Section 2, we give a literature review on the ARP and VRP. In Section 3, we present, first, a summary about a new transformation technique of the ARP into a VRP, then a new formulation of the sparse CARP is introduced. In Section 4, several approximate methods are developed: Section 4.3 details the constructive heuristic (CH), Section 4.4 details the improving randomized procedure (IRP) and Section 4.5 details the main steps of the tabu search algorithm (TS) for the sparse transformed CARP where the developed TS is applied over a sparse dynamic graph. In Section 5, numerical experiments are conducted on a set of instances of different sizes and densities. In Section 6, we give a conclusion and discuss some future research ideas for the sparse CARP.

2. Literature review

Routing problems constitute a wide field of research in optimization and operational research and they span a wide variety of routing type such as the vehicle (node) routing problems, inventory routing problems and arc routing problems. Many real-life applications are directly linked to the routing problems and especially in transportation and logistics context. In this section, we present a brief survey on the arc routing problem and its most well-known variants.

Arc routing problems aim at covering arcs on a transportation network [1]. In the literature, many variants of this problem were introduced and have been solved by different formulations and heuristic algorithms. These variants can be classified into three general classes: (i) the Chinese postman problem (CPP) and its variants [9, 14, 30, 41], (ii) the rural postman problem (RPP) and its variants [8, 17, 18, 32, 46] and (iii) the capacitated arc routing problem (CARP) [20]. In [36], the authors have developed an exact method for the ARP. In [7], cutting plane techniques were developed for the ARP. A branch-and-cut algorithm has been implemented in [17], while in [37–39] the authors have applied both the cutting plane techniques and branch-and-cut methods to solve the ARP. Let $G = (V, E)$ be a simple connected undirected graph where V is the set of vertices and E is the set of edges. In the CARP, we consider a subset $E_R \subset E$ of required edges and the remaining edges are said to be non-required edges as they do not require any service, but they can be traversed any number of times. The CARP consists of finding the tours with a minimal total cost while verifying the following conditions:

(1) each tour starts from the unique depot and returns to it;
(2) every required edge is serviced by one single vehicle;

(3) the sum of the collected demands by each vehicle must not exceed its capacity.

In [13], the CARP has been proved to be NP-Hard. To solve optimally the CARP, two types of approaches do exist. Exact methods that determine an optimal integer solution and prove optimality by showing that its cost is a lower bound and LP-based methods that use heuristic solutions. For the first type, a branch-and-bound algorithm was developed by Saruwatari *et al.* [43] to solve small problem instances. Later, a cutting planes algorithm was applied by Belenguer and Benavent [4]. For the other CARP variants, solutions were obtained either by mathematical formulations or by transforming the problem into node routing problem. In the literature, one can find different formulations of both the objective and the set of constraints such as the mixed CARP (MCARP) [5], the periodic CARP (PCARP) [29], the CARP with intermediate facilities (CARPIF) or the multi-depots CARP (MD-CARP).

Several formulations have been considered for the CARP. One can recall the two-index formulation which was proposed by Belenguer and Benavent [3], one-index formulations were proposed by Belenguer and Benavent [3, 4], Letchford [31] and a set partitioning formulation was introduced by Gòmez-Cabrero [21]. In [49], the authors have implemented a branch-and-bound algorithm for CARP. A branch-and-cut approach including capacity constraints, odd-cut constraints and a cutting plane algorithm has been developed by Belenguer and Benavent [3] for the super-sparse formulation of the CARP. A two ant colony approach for the multi-depot capacitated arc routing problem has been developed by Kansou and Yassine [26, 27].

A cut-first/branch-and-price-second algorithm for the CARP was given by Bode and Irnich [6] where the presented columns-generation formulation has strong lower bounds, symmetry elimination, efficient pricing and an effective branching scheme. This branching scheme was the first that ensures integral CARP solutions, while the structure of the pricing problem remains unchanged and where all open benchmarks of Belenguer and Benavent [3] were solved. Note that the graphs of the instances presented in [6] are not all sparse, but some of them are complete. Moreover, the sparse graphs of our study has a greater sparsity than that used before. There is no method in the literature that deals with the sparse CARP concerning the degree of sparsity that we are dealing with.

Considering the heuristics of CARP, simple constructive classical algorithms were developed as the construct-strike algorithm, the path scanning algorithm and the augment-merge algorithm [13, 51]. The problem has also been solved by two-phase constructive methods such as route first cluster second, cluster first route second and cycle assignment. Other heuristics such as simulated annealing algorithms (SA) [51] or tabu search algorithms [25] were successfully implemented for the CARP.

On the other hand, many ARPs have been studied by applying some transformation techniques to transform them into VRPs since the vehicle routing problem and its variants were well studied in the literature to the contrary of the ARPs (for more details about these transformations one can refer to [15, 33, 40]).

However and to the most of our knowledge, no transformation does exist in the literature that addresses the ARP over sparse graphs and conserves both the sparsity of the original graph and the structure of the problem too. The previous transformations consider the obtained VRP on its complete graph and this goes against our study that aims at studying the problem over special sparse graphs.

Moreover, the mathematical formulation that we present in the following section deals with our special CARP under specific sparse graphs, where there is no formulation in the literature dealing with such a problem and under such very sparse graphs. As our studied CARP is special and new due to the very high sparsity of its underlying graph, we have built the mathematical model by considering the following assumptions:

(1) the high sparsity of the graph which is characterized by having a low density (down to 0.001 and less in some cases);
(2) the maximum degree of any vertex is equal to 3 *i.e.* each vertex in the graph has at most 3 incident edges;
(3) the set of the required edges is relatively small.

3. Modeling and transforming the sparse CARP

In what follows, we give a new mathematical formulation for the sparse CARP and we briefly introduce a new technique to transform the sparse CARP into a CVRP with feasible sparse graph.

3.1. A mathematical model for the CARP with a sparse graph

Throughout the following, we present a new formulation of the capacitated arc routing problem over sparse graphs with homogenous fleet of vehicles. Let $G = (V, E)$ be a graph where V denotes the set of vertices and E the set of edges. We denote by $R \subseteq E$ the set of the required edges *i.e.* the set of edges having positive demands to be serviced. We consider the following data and variables:

- Parameters:
  - K: the total number of the homogenous fleet of vehicles;
  - Q: the capacity of each vehicle;
  - d_i: the demand of the edge i;
  - Δ_i: the total demand served by the vehicle arriving at the service i including the demand of i;
  - c_i: the cost of the edge i;
  - $N(i)$: the neighborhood of the edge i;
  - ω_i: the capacity of edge i = a constant maximum number of times an edge can be traversed;
- Decision variables:
  - $x_{i,j}^{e,f}$: a binary variable which is equal to 1 if and only if the service at f is successive to the service at e by the same vehicle and the chosen shortest path between e and f includes the consecutive adjacent edges i and j, and 0 otherwise;
  - $y_{e,f}$: a binary variable equal to 1 if f is serviced directly after e, and 0 otherwise.

We denote by 0 the edge representing the departure of the vehicles from the depot and by 1 the edge that represents the return to the depot. This means that the graph has only one departure (edge 0) and only one return (edge 1). Hence, the mathematical formulation of the model:

$$\text{Minimize} \sum_{e,f \in R, i \in E, j \in N(i)} c_i x_{i,j}^{e,f} \tag{3.1}$$

Subject to:

$$y_{e,f} + y_{f,e} \leq 1 \quad \forall e, f \in R \tag{3.2}$$

$$\sum_{f \in R} y_{0,f} = K \tag{3.3}$$

$$\sum_{f \in R} y_{1,f} = 0 \tag{3.4}$$

$$\sum_{f \in R} y_{e,f} = 1 \quad \forall e \notin \{0, 1\} \tag{3.5}$$

$$\sum_{e \in R} y_{e,1} = K \tag{3.6}$$

$$\sum_{e \in R} y_{e,0} = 0 \tag{3.7}$$

$$\sum_{e \in R} y_{e,f} = 1 \quad \forall f \notin \{0, 1\} \tag{3.8}$$

$$\Delta_f \geq \Delta_e + d_f + (d_f + Q) \times (y_{e,f} - 1) \quad \forall e, f \in R, e \neq f \tag{3.9}$$

$$\sum_{j\in N(i)} x_{i,j}^{e,f} - \sum_{j\in N(i)} x_{j,i}^{e,f} = 0 \quad \text{if} \quad i \neq e, i \neq f, e, f \in R \tag{3.10}$$

$$\sum_{j\in N(e)} x_{e,j}^{e,f} - \sum_{j\in N(e)} x_{j,e}^{e,f} = y_{e,f} \quad e, f \in R \tag{3.11}$$

$$\sum_{j\in N(f)} x_{f,j}^{e,f} - \sum_{j\in N(f)} x_{j,f}^{e,f} = -y_{e,f} \quad e, f \in R \tag{3.12}$$

$$\sum_{e,f\in R, j\in N(i)} x_{i,j}^{e,f} \leq \omega_i \quad \text{with} \quad \omega_i \geq 1 \quad \text{if} \quad i \neq 0 \tag{3.13}$$

$$\sum_{e,f\in R, j\in N(0)} x_{0,j}^{e,f} = K \tag{3.14}$$

$$\sum_{e,f\in R, j\in N(i)} x_{j,i}^{e,f} \leq \omega_i \quad \text{with} \quad \omega_i \geq 1 \quad \text{if} \quad i \neq 1 \tag{3.15}$$

$$\sum_{e,f\in R, j\in N(1)} x_{j,1}^{e,f} = K \tag{3.16}$$

$$x_{i,j}^{e,f}, y_{e,f} \in \{0,1\}, \Delta_e \leq Q \quad \forall e, f \in R, i, j \in E. \tag{3.17}$$

The objective function aims at minimizing the total costs. Constraints (3.2) show that either e is serviced before f or *vice versa*. Constraints (3.3) show that all the vehicles depart from the only departure of the depot which is represented by edge 0. Constraints (3.4) are used to identify that the depot has only one departure. The number of successors of each edge $e \notin \{0,1\}$ is determined by Constraints (3.5) where there exists one and only one successor. Constraints (3.6) show that all the vehicles have to return back to the depot after finishing their corresponding services. Constraints (3.7) determine that there is only one return to the depot and this return is given by the edge 1. The number of predecessors is determined by Constraints (3.8) where each edge $e \notin \{0,1\}$ has exactly one only predecessor. Constraints (3.9) assure that if f is served directly after e then the total demand at edge f is greater than or equal to the total demand at edge e. Otherwise, the difference between these demands is less than Q. Shortest path constraints are represented in (3.10)–(3.12). Constraints (3.13)–(3.16) determine the number of times an edge can be traversed, and this number is denoted by ω and assigned to be named as the capacity of the edge for each edge of the graph. In detail, Constraints (3.13) and (3.15) impose that the number of times of traversing an edge is bounded by some ω where this ω is constant for the edge and may vary from one edge to another. For the departure and the return of the depot in G, the maximum number of traversals is given by $\omega = K$ ensuring the journey of all the vehicles from and to the depot. Constraints (3.17) stand for decision variables.

In this model, we do not neither impose that the edges of the graphs are traversed only once nor count the number of times an edge is traversed by introducing aggregated variables like most CARP formulations as in [3, 49]. We impose a fixed number of maximum traversals for each edge for which the latter can be traversed less than or equal to this constant. Moreover, under this formulation we use a dynamic graph due to the modifications that happen upon changing the edge capacity once the corresponding edge is traversed. In other terms, each time an edge e of the graph is traversed, its capacity *i.e.* its corresponding ω is decremented by 1, and once this capacity reaches zero, this means that e cannot be traversed any more and thus we then consider the problem under $G\backslash\{e\}$. This is why we use a dynamic graph because the structure of this graph is changing while building the solution. This formulation may suffer from symmetry due to the use of non-oriented graphs in our study.

3.2. A transformation of the sparse CARP into a CVRP with feasible sparse graph

In previous works, transformation techniques have been built to transform the ARP into VRP. However, all the previous studied transformations in the literature led to VRPs with complete graphs. Thus these transformations

cannot be adapted to the sparse ARP since they do not guarantee the sparsity property. In [2], the authors defined the VRP with a sparse feasibility graph if each customer has only a few other customers to which it is adjacent for a route which is similar to our problem. The idea is then to build a transformation for the sparse ARP and to take advantage of the results on the VRP with sparse feasibility graph. Another argument was stated in [35] that exploiting sparsity reduces the resolution computing time and allows more memory space for the tackled problem instance. In this section, we give a brief summary of the transformation technique that we developed in a previous work in [47,48].

An ARP defined over a graph $g = (V, E)$ is transformed into an equivalent VRP over a graph $G = (V', E')$ where $G = L(g)$ is the line graph of G [50]. Each edge $e = \{i, j\} \in E$ is represented by one node m_{ij} in G (not only the required edges are considered and transformed into required nodes, but also the non-required ones). Hence, the resulting VRP problem is defined on $G = L(g) = (V', E')$ in which $V' = \bigcup_{(i,j)\in E} \{m_{ij}\}$. This transformation is specific for the graphs whose number of edges does not exceed the number of vertices added to half the number of vertices *i.e.* suppose that $|V| = n$ and $|E| = m$, we are interested in the graphs where $n \leq m \leq n + \alpha$ with $0 \leq \alpha \leq \frac{n}{2}$ such that only 2α vertices has the maximum degree 3 and the degree of each of the remaining vertices is less than or equal to 2. The costs division in the new graph is done as follows. An edge $(m_{ij}, m_{i'j'})$ in the new graph $L(G)$ has the cost $\frac{1}{2}(c(i, j) + c(i', j'))$ where $c(i, j)$ is the cost of the arc (i, j) in G. Following this specific procedure to divide the costs of the edges in the obtained VRP and due to the transformation itself, a solution of the final VRP is proved to be also a solution of the initial problem.

Remark 3.1. The original sparse CARP and its transformed CVRP with sparse feasibility graph are equivalent problems. On one hand, if S is a solution of the CARP then S is a true cycle (closed walk respectively) which has an isomorphic cycle in the VRP (closed walk respectively) as its image due to the distribution of the costs. On the other hand, if T is a solution of the obtained CVRP, then as T is in the line graph of the initial graph, we consider the following two cases:

(1) T is a cycle, then T is a line graph of some graph in G and both are isomorphic. This is enough to show that as T is feasible (or optimal) then its inverse in G is also feasible (and optimal).
(2) T is a closed walk and since T does not contain any of the 9 subgraphs mentioned in [22,42,45] as induced subgraphs, then T admits an inverse that is isomorphic too. This shows that both solutions are feasible (and optimal).

This transformation allows the structure of the connected graph to be completely recovered from its line graph, where the edges of G are equivalent to the vertices in $L(g)$ and the incident ones in g are equivalent to the adjacent vertices in $L(g)$. Moreover, line graphs can be recognized in a linear time. This transformation conserves not only the structure of the problem, but also the sparsity of the original graph. More details on this work will appear in [48].

4. Solution approaches for the sparse CARP

Despite our literature research, we did not find any heuristic that has been used to solve the studied sparse CVRP problem, perhaps due to the different and special structures of the used graph. In the following, we will build tailored based heuristics to solve the studied problem.

4.1. An initial solution for the transformed CARP

We present a new tailored heuristic algorithm to solve our special sparse CARP. The constructive heuristic CH that we develop is applied on the CVRP which is obtained by the discussed transformation in Section 3.2. Prior to detail CH, we give a brief recall on the dynamic graphs [11] that we will use for the algorithmic design.

4.2. Dynamic graphs

For recall, a graph is said to be a dynamic graph if it is subjected to some discrete changes such as insertion or deletion of edges or vertices. Such graphs are important in graph algorithms to design algorithmic techniques and data structures. Deleting or inserting edges or vertices of the graph is known by "`update on a graph`". This update includes the changes that are associated with the edges or the vertices as the cost for example. The notion of the dynamic graph algorithm facilitates the computation of the solution of a problem efficiently instead of recomputing it from the scratch each time. For more details, readers may refer to [11, 12].

In our work, we use the dynamic graph concept due to the update of the edge capacities ω each time an edge is traversed.

4.3. The constructive heuristic CH

In the following, we give the heuristic algorithm, namely CH, for the sparse CARP. CH is a constructive based heuristic and is applied on the obtained CVRP which is equivalent to the original CARP where the used graph is a dynamic graph. In all what follows, we use the concatenation for each union denoted by $\cup^c$.

The procedure takes as input the original graph g with n vertices, m edges and $\omega_{\max}$ the maximal edge capacity held by the edges of g.

We introduce the following notations used by CH (Fig. 1):

- g: the original graph where the ARP is defined;
- $\tau(g)$: the transformation function to transform g of the sparse CARP into G of its corresponding sparse CVRP;
- G: the obtained line graph of g;
- $N(G)$: the set of nodes of G;
- Serv: the set of services;
- d: the total demand served at a current step;
- d': the total demand served at a current tour;
- dem(s): the demand of node s;
- D: the total demand to be served;
- Q: the capacity of each vehicle;
- $\omega_0(i)$: the initial capacity of node i;
- $\omega(i)$: the capacity of node i at a current step of construction;
- s: the current service;
- $next$: the next service returned by the procedure Next();
- P: all paths between the current service and the possible next service;
- $Dijkstra(current, G)$: gives all the shortest paths between a current service and all possible next services excluding the extremities of the paths;
- Finish(d', P): returns true if $d' + dem(S) > Q \quad \forall S \in S(P)$ and 0 else; where $S(P)$ is the set of services at the end extremities of Dijkstra paths;
- $path$: the current selected path between the current and the next service;
- $Select(s, next, P)$: it selects a path from P between the current and the next service;
- S: the solution (set of nodes taken by the vehicles during their tour).

CH is a constructive heuristic which uses an opportunism strategy to compute the distance between the current service and the next service. We detail in the following the main steps of CH.

- **Steps 6–8**: `while` there is unserved demand, add the service s to the set of services and reach to this service by a `Dijkstra path`.
- **Steps 9–12**: `If` the sum of the served demands does not exceed the capacity of the vehicle, `then` select a next service and consider all the paths connecting the current service to the next one. `Else`, return back to the depot.

Input: An original CARP instance with graph g and parameters (n, m, ω_{max})
Output: A feasible solution S with value V(S) for the transformed CARP

Initialization
1. $G \leftarrow \tau(g; n, m, \omega_{max})$;
2. $d \leftarrow 0$; $s \leftarrow 0$; $next \leftarrow -1$;
3. $Serv = \emptyset$; $P = \emptyset$; $\mathrm{S}_0 = \emptyset$
4. $\forall i \in N(G)\ \omega(i) \leftarrow \omega_0(i)$;
5. $\mathrm{S} \leftarrow \mathrm{S}_0$;
Main Steps
6. `While` $(d \vert D)$ `do`
7. $Serv \leftarrow Serv \cup^c \{s\}$;
8. $path \leftarrow Dijkstra(s, G)$;
9. `If (Not` Finish(d', P)`) then`
10. $next \leftarrow Next(s, P)$;
11. `Else`
12. $next \leftarrow 1$;
13. `EndIf`
14. $path \leftarrow$ Select$(s, next, P)$;
15. $\mathrm{S} \leftarrow \mathrm{S} \cup^c (path \setminus \{next\})$;
16. `For` $(i \in path \setminus \{next\})$ `do`
17. $\omega(i) \leftarrow \omega(i) - 1$;
18. `EndFor`
19. `For` $(i \in N(G))$ `do`
20. `If` $(\omega(i) = 0)$ `then`
22. $G \leftarrow G \setminus \{i\}$;
23. `EndIf`
24. `EndFor`
25. `If` $(next \neq 1)$ `then`
26. $s \leftarrow next$; $d \leftarrow d + dem(s)$; $d' \leftarrow d' + dem(s)$;
27. `Else`
28. $\mathrm{S} \leftarrow \mathrm{S} \cup^c \{1\}$;
29. $s \leftarrow 0$;
30. $d' \leftarrow 0$;
31. `EndIf`
32. `EndWhile`
33. Exit with S with value V(S).

FIGURE 1. A constructive heuristic for the transformed sparse CARP into sparse CVRP: the CH.

- **Step 13**: `select` a path from the set of paths P.
- **Step 14**: `update` the solution to be the existing concatenated with the chosen path.
- **Steps 15–19**: `update` the capacity for all the vertices used in the chosen path.
- **Steps 20–21**: `allow` selecting a next service once the destination is not the depot. `Otherwise`, return back to the depot and start again with a new service where the served demand is initialized at zero. This is done in the remaining steps.

The general steps are divided into 3 stages. Each vertex starts with the initial capacity ω_0, *i.e.* each vertex starts with an initial maximal number of times to be traversed. Finally, the stopping criteria is attained when the served demand d is equal to the total demand D.

Stage 1

- compute all the paths between the current service and all other services that are different from the completed services;
- sort the completed services in the list "service".

Stage 2

- identification of "the end of tour": the termination of the tour is detected when the demand of each non-achieved service added to the served demand d' exceeds the vehicle capacity Q, or if each non-achieved service is not connected to the current service considering the configuration of the dynamic graph at the current step;
- if "the end of the tour" is not attained then we select the best service obtained by the Dijkstra's algorithm. Otherwise, the next service is associated to the return to the depot;
- in all cases we select the paths between the current and the next services which are calculated by Dijkstra's algorithm in a previous step;
- the nodes of this path are added to the end of the solution list except the node of the next service.

Stage 3

This stage can be considered as an "update stage".

- we decrease the capacity $\omega(i)$ of each node i in the selected path by 1 except for the last node of this path;
- each node i with a current capacity $\omega(i) = 0$ is removed from the dynamic graph. All the edges which are incident to this node are removed and any isolated node is removed too;
- the next service is associated to the current service. The demands d and d' are increased by $\text{dem}(i)$;
- the current tour is terminated by returning to the depot "node 1";
- the current service is updated as the depot exit "node 0" and the current served demand of the coming tour is 0.

A new tour begins if the current service s is set to 0. Each tour starts from node 0 of G (edge 0 of g) which represents the departure from the depot and ends at node 1 of G (edge 1 of g) which represents the return to the depot. Also each edge has a capacity less than or equal to $\omega_{\max}$. The parameters ω stand for the capacity of the nodes of the line graph G and the solution is represented by the concatenation of the paths of different vehicles (tours).

4.4. An improving randomized procedure based heuristic: IRP

The improving randomized procedure (IRP) consists of running for a fixed number of times the first constructive heuristic (CH) with a factor of randomness when selecting the next service. It looks like a GRASP algorithm where for each run, we select the best next services according to a probability p starting by the value 1, decreasing gradually by ε until reaching a minimal fixed value $\underline{p}$ and then increasing directly to its maximal value of 0.99. However (IRP) is not GRASP because there is no list of choices to choose the best, but we use the best according to some probability, and it does not end by a local search. The number of trials is initially fixed. (IRP) can be described as a combination of two strategies: (i) the opportunism strategy which gives the priority to the quality improvement and, (ii) the randomness which gives the priority to find a feasible solution by diversifying the search.

The search operation moves from one service to another depending on a set of the probability values which takes 1 as a first value while the trial decision is not included in a forbidden list called "list". Thus the next service is chosen by opportunism. According to both the probability value and the distance between these services, the next service is inserted in the solution if the next trial belongs to "list".

The number of runnings allows the solution to be improved progressively. The (IRP) procedure is detailed in the Figure 2.

Remark 4.1. For a given value of p, IRP(p) returns an improved solution if it is feasible, otherwise it returns an empty set. In what follows, we give some details of the randomized approach:

- Serv: the set of the required services;
- list: a list that contains a set of forbidden decisions for a solution S. We define a decision by the couple of service and its rank. If the service s has the index i in the set "Serv", then (s, i) represents a possible decision for the solution S;

Input: An original ARP instance with graph g and parameters (n, m, ω_{max})
Output: An improved feasible solution S^{best} with value $V(S^{best})$ for the transformed CARP

Initialization
1. G $\leftarrow \tau(g; n, m, \omega_{max})$;
2. S $\leftarrow \emptyset$; $S^{best} \leftarrow$S;
3. list $\leftarrow \emptyset$;
4. $p \leftarrow 1$;
5. $t \leftarrow 0$;
6. $t_{max} \leftarrow const$;

Main Steps
7. `While` $(t \leq t_{max})$ `do`
8. $t \leftarrow t + 1$;
9. S $\leftarrow$ CH(p);
10. `If` S $\neq \emptyset$ `then`
11. Serv$\leftarrow$ service(S);
12. `EndIf`;
13. list $\leftarrow$ list $\cup$ {Serv};
14. `If` |list| = maxSize `then`
15. list $\leftarrow$ list\{oldest};
16. `EndIf`;
17. `If` ((V(S) $\leq$ V(S^{best})) or ($S^{best} = \emptyset$)) `then`
18. $S^{best} \leftarrow$ S;
19. `EndIf`;
20. $p \leftarrow p - \varepsilon$;
21. `If` $(p = \underline{p})$ `then`
22. $p \leftarrow 1 - \varepsilon$;
23. `EndIf`;
24. `EndWhile`;
25. Exit with S with value V(S).

FIGURE 2. An improving feasible solution: the IRP approach.

- t: a counter representing the current trial (or run) for IRP(p);
- p: a probability used for IRP(p);
- $\underline{p}$: a minimal accepted value (threshold) of the probability p;
- maxSize: the maximal size of "list";
- oldest: an oldest forbidden trial decision;
- S^{best}: the best solution constructed up to the current trial;
- $V(S)$: the objective value of the solution S.

The difference between CH and IRP procedures resides in the return of the function "Next()". In IRP, at each stage of the routing, we select the next task to be visited by choosing the best non-forbidden next service that it is not in the list "list" (of a shortest path or a minimal cost) with a probability p. If not chosen, then we select the second best non-forbidden next service with the same probability. We reiterate until we obtain a best service to add to the solution.

Using different probabilities, we obtain different solutions with different qualities and then many trials are done. The current best solution is saved and returned at the termination of the process.

4.5. A tabu search algorithm for the sparse ARP

Tabu search (TS) is a metaheuristic based local search method which can be used for solving combinatorial optimization problems [19, 24]. The main concept of TS resides in the use of the adaptive memory that allows the search to guide the solution process by exploiting the search history. The adaptive memory is exploited to forbid the search to re-consider solutions that have already been visited. Our memory implementation employs

Input: An original CARP instance with a solution S and value V(S)
Output: An efficient solution S^{best} with value V(S^{best}) for the transformed CARP

Initialization
1. S ← IRP();
2. S^{best} ←S; V(S^{best}) ← V(S);
3. tlist ← ∅;
Main Steps
4. `While` (Not stopping condition) `do`
5. neighborhood ← FindNeighborhood(S); // constructs neighbors of the current solution
6. S ← Best(neighborhood$_G$)\tList; // selects the best neighbor that it is non-tabu
7. Update (tList); // updates the tabu list
8. Update (G); // updates the dynamic graph G
9. `If` (V(S) ≤ V(S^{best}) `then` S^{best} ← S;
10. `EndIf`;
11. `EndWhile`;
12. Exit with S^{best} with value V(S^{best}).

FIGURE 3. A tabu search based algorithm for the obtained CVRP with sparse feasibility graph: TS.

functions that encourage search diversification and intensification. These two TS components permit the search to escape from local optimal solutions and in many cases to find optimal solutions.

In this section, we present a tabu search algorithm for our sparse arc routing problem. This tabu search algorithm is adapted to work under a dynamic graph.

4.5.1. A neighborhood of the current solution

The neighborhood construction of this algorithm depends mainly on the service move in the graph. At each iteration we choose randomly a non-tabu service to be inserted in another position and evaluate the new shortest paths in the new dynamic graph. The TS algorithm starts with the solution found by the heuristic IRP, and then improvements take place to ameliorate the solution.

During the exploration, we use a dynamic graph (as for CH) initialized by the line graph $G = L(g)$ representing the obtained CVRP by the above transformation. We also use a variable node capacity noted by ω which determines the number of times a node can be traversed by the vehicles. These two parameters are updated at each new exploration. We consider a neighborhood structure based on one service move. This means, in order to build a neighbor of a current solution, we change one service schedule of the current solution (the rank of the service), then the service has two possibilities for its move: (i) a move to another schedule of the same vehicle or (ii) a move to another schedule of a different vehicle. We give in Figure 3 the main steps of the TS algorithm adapted to our problem.

The function Best(neighborhood$_G$) selects the best neighbor of the solution in the dynamic graph G at its current situation. When the solution changes, then ω changes and the number of traversing determines whether a traversal is allowed or not, for this we update the dynamic graph by the function Update (G).

4.5.2. The main principle of the search process

At each iteration of the exploration, we choose a random service then we remove it from its rank in the service schedule in the neighbor. We update the neighbor by removing the paths connecting this service to its successor and its predecessor services. We evaluate, using Dijkstra, the shortest path between the predecessor of the service and its successor, then we insert this path in the neighbor. We insert the service in a random new rank in the service schedule by selecting a new random predecessor. Finally, considering the previous update of the dynamic graph G, we apply Dijkstra to find two shortest paths, the first one is between the new predecessor service and the related service, and the second one is between the related service and the new successor service.

We update the rank of each service in the solution schedule for each new neighborhood construction. These updated ranks are saved in the list of services. At each construction, we update the remaining capacity of each node as well as the dynamic line graph G.

If a node's capacity decreases to 0, then this node is removed from G. If it increases to a positive value, we add this node to G. Through this exploration process and for each current solution, if the constructed neighbor improves the solution, it is then inserted in the current solution. If no improvement is possible, we keep the best found solution found so far and we construct a new neighbor.

Remark 4.2. The solution space is not always connected since while building the neighborhoods, some cases may cause a disconnection. Such cases are discussed below:

Let X be a solution such that $\forall$ service s in X we have $\omega(s) = 0$, and let Y be a neighbor of X *i.e.* $Y \in$ neighborhood(X). Suppose that s is the service moved from its position in X to a new position which builds the neighbor Y. Let s_{pred} (resp. s_{next}) be the predecessor of s in X (resp. its successor).

If $s \in \bigcap_{P \in Paths(s_{pred}, s_{next})} P$ (*i.e.* s belongs to the set of all the possible paths between s_{pred} and s_{next}) and for an updated ω in the dynamic graph after removing s from its position between s_{pred} and s_{next}, then there is no path between s_{pred} and s_{next} not containing s. Since $\omega(s) = 0$, s cannot be moved from its position between s_{pred} and s_{next} without disconnecting them definitely.

Generally, the solution space is disconnected because of the high sparsity of the graph. This disconnection increases with small values of average ω. To adapt our approach to this particularity of the problem, we use an exploitation process which consists to restart the tabu search from different initial solutions obtained by the IRP procedure.

4.5.3. Intensification and diversification strategies

During the search process, if after a certain number of constructions the current solution is not improved, we select the last explored neighbor to replace the current solution. We choose this strategy of neighborhood construction to reduce the combinatorial explosion of the solution search and the memory space use. To escape from local optima regions, at each exploration of a new best solution, we put tabu the move which was used to construct that solution where the tabu list size is equal to the number of services multiplied by 10.

If promising zones are detected, we intensify the search and if the current explored region becomes not interesting (because of stagnation), we diversify the exploration. Once we obtain a new best solution, we update "maxIter" by its value times 2 except if it is equal to its maximal value (2000) as an intensification process. However, when the number of explorations exceeds the tolerated value (20 000 iterations), we update "maxIter" by its minimal value (200 iterations) as a diversification process.

For the aspiration criteria, if a tabu move improves the found best solution, then we remove the tabu status from the move of the corresponding solution. The "maxIter" which is the size of the explored neighborhoods is variable and its value depends on the intensification and the diversification processes. Our TS uses a multi-start process to avoid being stuck in some areas of the solution space.

5. Numerical results

In this section, we detail the numerical experiments carried out in order to test the proposed approach. We have coded all the algorithms with C++ language on a Ubuntu Linux 14.04 HP ProBook Core i3, 2.4 Ghz and 6 Go of Ram. These results are compared to those obtained by Cplex V12.71.

In the benchmark, we consider 9 families of instances such that each family comprises instances with the original graph noted g for which the sparse CARP is defined. We study each family with the same number of vertices $|V(g)| = n$, the same number of edges $|E(g)| = m$, the same number of services $|R|$ and the same total demand D. We define $d(g) = \frac{2m}{n(n-1)}$ the density of the original graph g (rounded to the nearest 10^{-3}). We consider a homogenous fleet of K vehicles where their capacity is equal to Q for each vehicle. We also set the total available loading capacity (number of vehicles multiplied by their capacities) closer to the total required demand.

Note that the approach that we present is a specific-tailored one for the routing problems with graphs having a high sparsity (or a small density) and a maximum degree equal to 3. Despite our literature research, other approaches to solve such problems do not exist, so that, we have no results in the literature to compare with our computational experiments.

In Table 1 we detail the families of instances for which we run both Cplex and the CH algorithm. 10 families with different sizes and densities of graphs are presented. The first column gives the number of the family, the number of vertices and the number of edges are given in the second and third column respectively. Column 3 gives the number of services, and the density of the graph is given in column 4. The amount of the total demand is given in the last column.

Table 2 presents the obtained results by Cplex and CH respectively. In the first 4 columns, we give the number of the family, the average value of its corresponding edge capacity ω, the number of the homogeneous vehicles K and their capacity Q. The whole number of run instances is given in column 5 titled by All, and the number of feasible and optimal obtained solutions by Cplex is given in column 6, and by CH is given in column 7. Column 8 represents the gap between the solution values determined by Cplex and those determined by CH. This gap is given by:

$$\frac{\text{value given by CH} - \text{value given by Cplex}}{\text{value given by Cplex}} \times 100. \tag{5.1}$$

The last two columns give the consuming CPU time by Cplex and by CH.

The results in Table 2 show the effectiveness of the CH algorithm by comparing the results obtained by it to that obtained by Cplex especially for the families with big size instances where Cplex fails to perform. This is shown by the number of feasible and optimal solutions obtained by Cplex on one hand and by the CH heuristic on the other hand.

To detail the results between Cplex and CH, we present in Table 3 some instances of the first two families 1 and 2, the corresponding edge capacity ω, the number of vehicles K and its capacity Q, the gap given by Cplex, the gap of CH which is calculated as in (5.1).

The CPU consuming time by Cplex in seconds and the corresponding CPU time needed by the heuristic CH in seconds to solve the problem. We note that the optimality of the solution is attained by Cplex for the most of the instances of families 1 and 2 with a small CPU consuming time. However, CH performs well by giving solutions that are close to the optimal ones in a very small CPU consuming time. This gives an overview about the effectiveness of the CH algorithm which gives a feasible near optimal solution with a very small time.

Starting from family 3 which has a moderate-size instances (50 nodes and 70 edges), Cplex cannot perform the solution due to the lack of its memory. However, CH performs well and gives solutions.

TABLE 1. A first set of different families of CARP problem instances.

Family #	$\lvert V(g)\rvert$	$\lvert E(g)\rvert$	$\lvert R\rvert$	$d(g)$ (%)	D
Family 1	15	18	8	0.171	1000
Family 2	20	25	10	0.132	1000
Family 3	50	70	33	0.057	1000
Family 4	100	120	59	0.024	1000
Family 5	120	142	95	0.02	1000
Family 6	163	181	110	0.014	1000
Family 7	231	317	121	0.012	1000
Family 8	257	362	191	0.011	1000
Family 9	307	439	309	0.001	1000
Family 10	400	600	357	0.008	1000

TABLE 2. Comparison between Cplex and CH for different families of instances by CH.

Family #	ω	K	Q	All	$\text{Cplex}_{\text{sol}}$	CH_{sol}	Av. gap (%)	Av. $\text{CPU}_{\text{Cplex}}$	Av. CPU_{CH}
Family 1	2	3	400	60	9 F – 21 O	9 F – 3 O	4.799	0.47 s	<1 s
Family 1	5	3	400	54	16 F – 18 O	13 F – 2 O	6.315	0.95 s	<1 s
Family 2	2	3	400	10	6 F – 0 O	0 F – 0 O	31.84	15 min 32 s	<1 s
Family 2	10	3	400	10	10 F – 0 O	0 F – 0 O	30.554	8 min 51 s	<1 s
Family 2	10	4	300	10	10 F – 0 O	2 F – 0 O	32.13	8 min 9 s	<1 s
Family 3	5	3	400	10	0 F – 0 O	2 F – 0 O	–	–	<1 s
Family 3	10	3	400	10	0 F – 0 O	6 F	–	–	<1 s
Family 3	12	3	400	10	0 F – 0 O	6 F	–	–	<1 s
Family 4	2	3	400	10	0 F – 0 O	0 F – 0 O	–	–	<1 s
Family 4	5	3	400	10	0 F – 0 O	0 F – 0 O	–	–	<1 s
Family 4	10	3	400	10	0 F – 0 O	2 F	–	–	<1 s
Family 4	12	3	400	10	0 F – 0 O	2 F	–	–	<1 s
Family 4	12	4	300	10	0 F – 0 O	1 F	–	–	<1 s
Family 5	2	3	400	300	0 F – 0 O	0 F – 0 O	–	–	<1 s
Family 5	5	3	400	144	0 F – 0 O	3 F	–	–	<1 s
Family 5	10	3	400	107	0 F – 0 O	19 F	–	–	<1 s
Family 5	12	3	400	115	0 F – 0 O	26 F	–	–	<1 s
Family 5	12	4	300	133	0 F – 0 O	17 F	–	–	<1 s
Family 6	5	3	400	106	0 F – 0 O	0 F – 0 O	–	–	<1 s
Family 6	10	3	400	111	0 F – 0 O	1 F	–	–	<1 s
Family 6	12	3	400	125	0 F – 0 O	3 F	–	–	<1 s
Family 6	15	3	400	118	0 F – 0 O	12 F	–	–	<1 s
Family 6	15	4	300	154	0 F – 0 O	6 F	–	–	<1 s
Family 7	5	3	400	101	0 F – 0 O	22 F	–	–	<1 s
Family 7	10	3	400	102	0 F – 0 O	33 F	–	–	<1 s
Family 7	15	3	400	102	0 F – 0 O	36 F	–	–	<1 s
Family 8	5	3	400	175	0 F – 0 O	26 F	–	–	<1 s
Family 8	10	3	400	160	0 F – 0 O	52 F	–	–	<1 s
Family 9	5	3	400	108	0 F – 0 O	26 F	–	–	<1 s
Family 9	10	3	400	115	0 F – 0 O	35 F	–	–	<1 s
Family 10	5	3	400	106	0 F – 0 O	19 F	–	–	<1 s
Family 10	10	3	400	110	0 F – 0 O	31 F	–	–	<1 s
Family 10	17	3	400	108	0 F – 0 O	49 F	–	–	<1 s

Notes. F: Feasible; O: Optimal.

In Table 4, we present the average results of different instances of sub-families of families 1 and 2 solved by Cplex and by IRP. For the IRP approach, we remark that more the average capacity of the edges increases, the algorithm needs less time to solve the problem. Moreover, this increase is faced by a decrease in the number of the IRP failures which validates that the complexity of the problem increases as the sparsity of the graph increases.

The IRP approach realizes better solutions than CH since it starts with the same initial solution and improves it gradually with a number of improvements.

Consider the following notations:

- "All": the number of all tested instances.
- " #Feas": the number of the feasible instances out of the all tested instances;
- "Average gap": the average gap between the CPLEX and the heuristic IRP;
- "Best gap": the best gap between the heuristic IRP and the CPLEX;
- "Worst gap": the worst gap between the heuristic IRP and the CPLEX;

TABLE 3. Results of different instances of families 1 and 2 by Cplex and by CH.

Family #	ω	K	Q	$\text{Cplex}_{\text{sol}}$	CH_{sol}	Cplex gap(%)	CH gap(%)	$\text{CPU}_{\text{Cplex}}$	CPU_{CH}
1	2	3	400	823.75	967.04	0	17.4	0.16	0.04
1	2	3	400	994.08	994.08	0	0	0.3	0.05
1	2	3	400	1308.96	1363.13	0	4.1	0.22	0.09
1	2	4	300	1594.64	1594.64	0	0	0.25	0.06
1	2	4	300	1217.44	1752.12	0	43.9	16.46	1.74
1	5	3	400	1204.97	1371.75	0	13.8	0.97	0.04
1	5	3	400	643.54	643.54	0	0	0.14	0.03
1	5	3	400	836.83	941.008	7.4	12.4	0.92	0.05
1	5	3	400	1290.28	1668.05	0	29.3	0.38	0.03
1	10	3	400	817	838.41	4.22	2.6	0.48	0.04
1	10	3	400	1297.08	1496.25	0	15.4	0.3	0.05
1	10	3	400	935.31	1241.82	1.44	32.8	0.53	0.03
2	3	3	400	805.6	1238.37	16.65	53.7	2.08	0.07
2	3	3	400	1169.39	1236.03	0	5.69	0.24	0.07
2	3	3	400	1601.11	1743.96	7.33	8.92	7.86	0.07
2	6	3	400	1383.52	1591.66	4.31	15.04	3.4	0.08
2	6	3	400	1796.84	1879.53	4.27	4.6	30.37	0.07
2	6	3	400	1284.38	1369.62	7.58	6.64	15.26	0.07
2	6	4	300	1782.05	1824.11	5.39	2.36	24.28	0.09
2	6	4	300	1071.92	1270.03	0.05	18.48	4	0.07
2	10	3	400	1535.99	1576.34	8.25	2.63	5.51	0.09
2	10	3	400	1210.22	1652.57	7.39	36.55	9.27	0.07
2	10	3	400	1367.82	1463.09	4.69	6.97	1.25	0.07
2	10	3	400	1337.27	1680.85	2.73	25.69	5.51	0.07

TABLE 4. Results of different sub-families of families 1 and 2 performed by Cplex and by IRP.

Family #	ω	All	#Feas.	Av. gap(%)	Best gap(%)	Worst gap(%)	#Zero gap	Av. $\text{CPU}_{\text{Cplex}}$	Av. CPU_{IRP}	#Fails.	#Imp.
1	2	280	50	0.0897	0	0.328	4	1.173	0.007	35	1
1	3	121	50	0.1262	0.0172	0.283	0	1.815	0.00928	24	1
1	4	88	50	0.0949	0	0.2213	4	5.386	0.00916	17	2
1	5	64	50	0.1076	0.0189	0.2956	0	3.6604	0.0083	13	2
1	6	62	50	0.0898	0	0.2703	3	3.1774	0.0095	5	2
2	3	53	30	0.1711	0.0103	0.3251	0	52.21118	0.2402	14	1
2	4	39	30	0.171	0.0256	0.4015	0	177.6804	0.026	9	2
2	5	36	30	0.1475	0.0323	0.2654	0	141.1997	0.0366	7	2
2	6	34	30	0.1282	0.0152	0.2499	0	61.5477	0.0355	5	3

– " #Zero gap": the number of instances where both Cplex and IRP obtain the same value;
– "Av. $\text{CPU}_{\text{Cplex}}$": the average Cplex time taken to solve the problem;
– "Av. CPU_{IRP}": the IRP consuming time;
– " #Fails": the number of instances where the heuristic failed to solve the problem;
– " #Imp": the number of improvements done by IRP.

For the first two families, we give the results obtained by CPLEX and IRP. The experiments were done according to the different values of ω, K and Q. We run a group of instances of a sub-family of each main family of Table 4. Each sub-family the main characteristics of its corresponding main family, and has its own fixed parameters ω, K, and Q. Every instance of this sub-family has the same parameters of the sub-family itself.

TABLE 5. Results obtained by IRP for the families 3, 4, 5, 8, 9 and 10.

Family #	ω	K	Q	All tested	Feas.	Av. CPU_{IRP}	#Imp
3	4	3	400	29	20	0.2038	$0 < I < 3$
3	5	3	400	27	20	0.2106	$1 < I < 4$
3	10	3	400	20	20	0.2008	3
3	10	6	190	135	20	0.199	$0 < I < 3$
4	7	3	400	56	20	0.7802	$0 < I < 3$
4	12	3	400	25	20	0.8645	$0 < I < 3$
4	7	4	300	87	20	0.8227	$0 < I < 3$
4	7	6	190	912	20	0.8418	$0 < I < 3$
5	5	3	400	142	20	1.0126	$0 < I < 3$
5	12	3	400	27	20	1.2105	$0 < I < 4$
5	7	5	240	222	20	1.0651	$0 < I < 3$
5	7	6	190	325	10	1.1589	$0 < I < 3$
8	7	3	400	28	20	19.9628	$0 < I < 3$
8	12	3	400	24	20	20.303	$0 < I < 4$
8	7	6	190	204	20	18.5081	$0 < I < 3$
9	7	3	400	31	20	54.0664	$0 < I < 3$
9	10	3	400	23	20	54.3984	$0 < I < 3$
9	12	3	400	23	20	52.6137	$0 < I < 3$
9	7	5	240	130	20	47.6028	$0 < I < 3$
9	7	6	190	229	20	46.136	$0 < I < 3$
10	10	3	400	28	20	115.8591	$0 < I < 3$
10	12	3	400	34	20	119.8183	$0 < I < 3$
10	12	5	240	33	20	102.522	$0 < I < 3$
10	12	6	190	58	20	100.1831	$0 < I < 3$

For the first two families, we give the results obtained by CPLEX and IRP. The experiments were done according to the different values of ω. The first column of Table 4 gives the number of the main family, the second gives the corresponding average ω, the third column stands for the total number of instances done in one sub-family, and the number of the obtained feasible instances is given in column 4. The average gap between the IRP and Cplex is given in column 5. Columns 6 and 7 give the best and the worst gap between IRP and Cplex. The number of instances where Cplex and IRP produce the same solution is shown in column 8. The average consuming times of Cplex and IRP are given in columns 9 and 10. Column 11 shows the number of instances where IRP failed to improve, and finally in the last column, the number of improvements done by IRP is given. For the gap between Cplex and IRP, it is similar to that given in (5.1) replacing CH by IRP.

In Table 5, we present the results obtained for some sub-families of each of families 3, 4, 5, 8, 9 and 10 by the IRP approach. We give the number of obtained feasible instances out of all the run ones. The average CPU time and the number of improvements are also given.

In Table 6, we present 4 families of different properties. Detailed results for some instances of families 1 and 2 are given in Table 7 and the average results for these two families are given in Table 8. In Table 7 we present the number of the family, the number of the selected instance, the best lower bound given by Cplex, the solution determined by Cplex, by IRP and by TS, and the CPU consuming times needed by Cplex and by TS in the last two columns. Similarly, some detailed instances of families 3 and 4 are given in Table 9, and the average results of each family are given in Table 10. We give the average gap between the solution obtained by Cplex and the one obtained by IRP as given in (5.1) replacing CH by IRP.

We also compute the deviation between the IRP and TS algorithms which is calculated according to:

$$\frac{\text{value given by IRP} - \text{value given by TS}}{\text{value given by TS}} \times 100. \tag{5.2}$$

TABLE 6. Family instances for TS.

Family #	$\|V(g)\|$	$\|E(g)\|$	$\|R\|$	$d(g)$ (%)	D	ω	K	Q
Family 1	15	18	8	0.171	100	6	3	40
Family 2	20	25	8	0.132	100	6	3	40
Family 3	50	70	33	0.057	1000	7	3	400
Family 4	100	120	59	0.024	1000	4	3	400

TABLE 7. Comparative results between Cplex, IRP and TS for instances of Family 1 and Family 2.

Family #	Instance	BLB	Cplex	IRP	TS	$\text{CPU}_{\text{Cplex}}$ (s)	CPU_{TS} (s)
Family 1	13	845	967	972	972	0.37	1
Family 1	26	1232	1232	1259	1232	0.39	<1
Family 1	39	1031	1279	1484	1423	1	<1
Family 1	42	996	996	1617	1617	0.12	<1
Family 1	72	740.88	838	1075	886	0.19	<1
Family 2	16	1103	1208	1394	1296	2	1
Family 2	26	1429	1493	2080	2080	1	1
Family 2	49	710	806	1020	1020	1	1
Family 2	55	898	961	1061	1021	0.43	1
Family 2	79	1531.125	1590	1890	1546	1	1

Notes. BLB: Best Lower Bound.

TABLE 8. Average results of families 1 and 2.

Family	Av. gap(%)	Best gap(%)	Worst gap(%)	Av. deviation(%)	Biggest deviation(%)	Smallest deviation(%)
Family 1	0.0422	0	0.384	0.0259	0.4261	0
Family 2	0.0411	0	0.2822	0.0146	0.2225	0

In the following, BLB stands for the best lower bound determined by Cplex. Average gap represents the gap between the solution determined by Cplex and that determined by the heuristic IRP. On the other hand, the deviation represents the deviation between the heuristic IRP and the TS solutions.

The results of the tabu search algorithm (TS) are given in Tables 7–10. TS algorithm gives better results than Cplex which is big time consuming especially for big size instances. It also improves the solutions obtained by IRP. Table 7 shows that for example the problem instance 79, TS gives a solution (of value 1546) better than the one found by Cplex (of value 1590) and better than the one obtained by IRP (of value 1890). Also, if we consider for example the problem instance 26 of Family 1, the solution found by TS (of value 1232) is optimal and equal to solution obtained by Cplex (of the same value) and better than the one obtained by IRP (of value 1259). For the big size instances presented in Tables 9 and 10, TS improves the results of IRP with a small CPU time.

6. CONCLUSION

In this paper, we introduce three different approaches to study the arc routing problem over a feasible sparse graph for which we propose, and present a new mathematical formulation for the sparse capacitated arc routing problem. The contribution of our work is that it studies the CARP over a special class of graphs. This problem

TABLE 9. Comparative Results between IRP and TS for instances of Family 3 and Family 4.

Family #	Instance	IRP	TS	Deviation(%)	CPU_{TS} (s)
Family 3	29	2644	2644	0	7
Family 3	55	4185	3078	0.3596	9
Family 3	62	3255	3149	0.0337	3
Family 3	75	3729	3384	0.1019	11
Family 3	85	7404	3775	0.9613	15
Family 4	35	5342	5342	0	8
Family 4	41	6983	7169	0.0266	7
Family 4	48	8022	7526	0.0659	43
Family 4	72	7041	7006	0.005	10
Family 4	93	5830	5754	0.0132	15

TABLE 10. Average results of families 3 and 4.

Family #	Av. deviation(%)	Biggest deviation(%)	Smallest deviation(%)
Family 3	0.0715	0.3596	0
Family 4	0.0079	0.0659	0

has not been studied before and this is why it is not trivial to compare the results obtained by our algorithms to those obtained by the classical ones and this despite our deep literature search. Our model carries new terms of "edge capacity" in CARP and "node capacity" in CVRP which impose the number of traversing an edge in CARP and a node in CVRP to be limited to some maximal value. This concept is introduced for the first time, whereas in the previous studies, the number of traversals of an arc was either counted or imposed to be 1. The contribution of our supposition shows its efficiency in the area of applications in real life where sometimes, the number of traversals of an arc is forced to not exceed some values. Such applications may appear in the domain of steganography for instance.

To solve the problem approximately, we built a constructive heuristic algorithm (CH) to obtain an initial feasible solution for the obtained CVRP with sparse feasibility graph. Later, we proposed an improving randomized procedure (IRP) for a first improvement of the solution obtained by CH. The IRP is based on a factor of randomness and the strategy tries to improve the initial solution by a fixed number of CH runs. Finally, an adapted tabu search algorithm (TS) under a dynamic graph was proposed to solve the problem. In our TS approach, we adapted the aspiration criteria and the intensification and diversification components especially because the nature of the solution space is disconnected in general.

Our algorithm improves the quality of the obtained solutions by the first 2 heuristics (CH and IRP). The efficiency of TS is measured by the quality of the obtained solution and by the CPU computational time. In many cases, TS obtained optimal solutions and all the cases highly improved quality ones. These outputs were especially obtained big size instances.

The problem that we have studied is a new one and as a future work, we are interested in studying the sparse arc routing problem under the uncertainty case for which we consider a set of scenarios (or data uncertainty) of arcs traversals. Robust optimization as a min–max optimization [44] with scenarios on the costs is in this case a good alternative approach to deal with the robust arc routing problem with a graph sparsity characteristic.

Acknowledgements. The authors would like to thank the anonymous referees for their helpful comments and insightful suggestions to improve the content of this paper.

References

[1] A.A. Assad and B.L. Golden, Arc routing methods and applications, edited by M.G. Ball, T.L. Magnanti, C.L. Monma and G.L. Nemhauser. In: Network Routing, Handbooks in Operations Research and Management Science. Elsevier (1995) 375–483.

[2] J.E. Beasley and N. Christofides, Vehicle routing with a sparse feasibility graph. *Eur. J. Oper. Res.* **98** (1997) 499–511.

[3] J.M. Belenguer and E. Benavent, The capacitated arc routing problem: valid inequalities and facets. *Comput. Optim. Appl.* **10** (1998) 165–187.

[4] J.M. Belenguer and E. Benavent, A cutting plane algorithm for the capacitated arc routing problem. *Comput. Oper. Res.* **30** (2003) 705–728.

[5] J.M. Belenguer, E. Benavent, P. Lacomme and C. Prins, Lower and upper bounds for the mixed capacitated arc routing problem. *Comput. Oper. Res.* **33** (2006) 3363–3383.

[6] C. Bode and S. Irnich, Cut-first branch-and-price-second for the capacitated arc-routing problem. *Oper. Res.* **60** (2012) 1167–1182.

[7] A. Corberán and J.M. Sanchis, The general routing problem polyhedron: facets from the RPP and GTSP polyhedra. *Eur. J. Oper. Res.* **108** (1998) 538–550.

[8] A. Corberán, A. Letchford and J.M. Sanchis, A cutting plane algorithm for the general routing problem. *Math. Program. Ser. A* **90** (2001) 291–316.

[9] A. Corberán, R. Martí and J.M. Sanchis, A GRASP heuristic for the mixed Chinese postman problem. *Eur. J. Oper. Res.* **142** (2002) 70–80.

[10] G.B. Dantzig and J.H. Ramser, The truck dispatching problem. *Manage. Sci.* **6** (1959) 80–91.

[11] C. Demetrescu, I. Finocchi and G. Italiano, Dynamic graphs. Chapter 10.2, edited by J. Gross and J. Yellen. In: Handbook of Graph Theory. CRC Press (2003) 985–1014.

[12] C. Demetrescu, I. Finocchi and G. Italiano, Dynamic graphs. Chapter 22, edited by D.P. Mehta and S. Sahni. In: *Handbook of Data Structures and Applications*. CRC Press (2004).

[13] M. Dror, editor, Arc Routing – Theory, Solutions and Applications. Kluwer Academic Publishers (2000).

[14] H.A. Eiselt, A historical perspective on arc routing, edited by M. Dror. In: Arc Routing: Theory, Solutions and Applications. Springer, Boston, MA (2000) 1–16.

[15] L. Foulds, H. Longo and J. Martins, A compact transformation of arc routing problems into node routing problems. *Ann. Oper. Res.* **226** (2015) 177–200.

[16] Z. Fekete and L. Szegö, A note on $[k, l]$-sparse graphs. Egrerváry Research Group on Combinatorial Optimization, Pázmány P. sétány 1/C, H1117, Budapest, Hungary www.cs.elte.hu/egres (2005).

[17] G. Ghiani and G. Laporte, A branch-and-cut algorithm for the undirected rural postman problem. *Math. Program.* **87** (2000) 467–481.

[18] G. Ghiani, R. Musmanno, G. Paletta and C. Triki, A heuristic for the periodic rural postman problem. *Comput. Oper. Res.* **32** (2005) 219–228.

[19] F. Glover, Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.* **13** (1986) 533–549.

[20] B.L. Golden and R.T. Wong, Capacitated arc routing problems. *Networks* **11** (1981) 305–315.

[21] D. Gòmez-Cabrero, J.M. Belenguer and E. Benavent, Cutting plane and column generation for the capacitated arc routing problem. Presented at ORP3Valencia, Spain (2005).

[22] J.T. Gross and J. Yellen, Graph Theory and its Applications, 2nd edition. CRC Press, Boca Raton, FL (2006).

[23] G. Gutin and A.P. Punnen, The Traveling Salesman Problem and its Variations. Springer, Boston, MA (2006).

[24] P. Hansen, The steepest ascent mildest descent heuristic for combinatorial programming. Presented at the Congress on Numerical Methods in Combinatorial Optimization. Capri, Italy (1986).

[25] A. Hertz, G. Laporte and M. Mittaz, A tabu search heuristic for the capacitated arc routing problem. *Oper. Res.* **48** (2000) 129–135.

[26] A. Kansou and A. Yassine, A two ant colony approaches for the multi-depot capacitated arc routing problem. In: *Proc. of International Conference on Computers and Industrial Engineering* (2009) 1040–1045.

[27] A. Kansou and A. Yassine, New upper bounds for the multi-depot capacitated arc routing problem. *Int. J. Metaheuristics* **1** (2010) 81–95.

[28] R.M. Karp, Reducibility among combinatorial problems, edited by R.E. Miller and J.W. Thatcher. In: Complexity of Computer Computations. Plenum, New York, NY (1972) 85–103.

[29] P. Lacomme, C. Prins and W. Ramadane-Chérif, Evolutionary algorithms for multiperiod arc routing problems. In: Proc. of the 9th Int. Conf. on Information Processing and Management of the Uncertainty in Knowledge-Based systems (IPMU 2002). ESIA-Univ. of Savoie (2002) 845–852.

[30] G. Laporte, Modeling and solving several classes of arc routing problems as travelling salesman problems. *Comput. Oper. Res.* **24** (1997) 1057–1061.

[31] A.N. Letchford, *Polyhedral results for some constrained arc-routing problems*. Ph.D. thesis, Lancaster University, Lancaster, UK (1997).

[32] A.N. Letchford and R.W. Eglese, The rural postman problem with deadline classes. *Eur. J. Oper. Res.* **105** (1998) 390–400.

[33] H. Longo, M. Poggi de Aragão and E. Uchoa, Solving capacitated arc routing problems using a transformation to the CVRP. *Comput. Oper. Res.* **33** (2006) 1823–1837.

[34] J. Nešetřil and P.O. Mendez, Sparsity: Graphs, Structures and Algorithms. Springer Berlin Heidelberg (2012).
[35] A. Oukil, *Exploiting sparsity in vehicle routing algorithms*. Ph.D. thesis, Lancaster University, Lancaster, UK (2008).
[36] M.W. Padberg and M.R. Rao, Odd minimum cut'sets and b-matchings. *Math. Oper. Res.* **7** (1982) 67–80.
[37] M.W. Padberg and G. Rinaldi, Optimization of a 532 city symmetric traveling salesman problem by branch and cut. *Oper. Res. Lett.* **6** (1987) 1–7.
[38] M.W. Padberg and G. Rinaldi, Facet identification for the symmetric traveling salesman polytope. *Math. Program.* **47** (1990) 219–257.
[39] M.W. Padberg and G. Rinaldi, A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Rev.* **33** (1991) 60–100.
[40] W.L. Pearn, A. Assad and B.L. Golden, Transforming arc routing into node routing problems. *Comput. Oper. Res.* **14** (1987) 285–288.
[41] W.L. Pearn, Solvable cases of the k-person Chinese postman problem. *Oper. Res. Lett.* **16** (1994) 241–244.
[42] A. van Rooij and H. Wilf, The interchange graph of a finite graph. *Acta Math. Acad. Sci. Hungar.* **16** (1965) 263–269.
[43] Y. Saruwatari, R. Hirabayashi and N. Nishida, Node duplication lower bounds for the capacitated arc routing problem. *J. Oper. Res. Soc. Jpn.* **35** (1992) 119–133.
[44] A. Sbihi, A cooperative local search-based algorithm for the Multiple-Scenario Max–Min Knapsack problem. *Eur. J. Oper. Res.* **202** (2010) 339–346.
[45] S. Skiena, Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica. In Line graph, Section 4.1.5. p. 128 (1990) 135–139.
[46] J.S. Sniezek, *The capacitated arc routing problem with vehicle/site dependencies: an application of arc routing and partitioning*. Ph.D. thesis, University of Maryland (2001).
[47] S. Tfaili, *Contribution aux graphes creux pour le problème de tournées sur arcs déterministe et robuste: théorie et algorithmes*. Ph.D. thesis, Université Le Havre Normandie, Le Havre (2017).
[48] S. Tfaili, H. Dkhil, A. Sbihi and A. Yassine, A transformation technique for arc routing problems into node routing problems with a sparse feasible graph. Working Paper (2016).
[49] S.A. Welz, *Optimal solutions for the capacitated arc routing problem using integer programming*. Ph.D. thesis, University of Cincinnati, Cincinnati (1994).
[50] D.B. West, *Characterizing line graphs*, 2nd edition. In: Introduction to Graph Theory. Prentice-Hall, Englewood Cliffs, NJ (2000) 279–282.
[51] S. Wøhlk, *Contributins to arc routing*. Ph.D. thesis, University of Southern Denmark, Odense (2005).