Computers & Operations Research 43 (2014) 309–317

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Location-arc routing problem: Heuristic approaches and test instances

Rui Borges Lopes [a,*], Frank Plastria [b], Carlos Ferreira [a], Beatriz Sousa Santos [c]

[a] Department of Economics, Management and Industrial Engineering – C.I.O., University of Aveiro, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal

[b] Department of Mathematics, Operational Research, Statistics and Information Systems for Management, Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussel, Belgium

[c] Department of Electronics, Telecommunications and Informatics – I.E.E.T.A., University of Aveiro, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal

ARTICLE INFO

Available online 21 October 2013

Keywords:
Location-routing
Arc routing
Heuristics
Test instances

ABSTRACT

Location-routing is a branch of locational analysis that takes into account distribution aspects. The location-arc routing problem (LARP) considers scenarios where the demand is on the edges rather than being on the nodes of a network (usually a road network is assumed). Examples of such scenarios include locating facilities for postal delivery, garbage collection, road maintenance, winter gritting and street sweeping. This paper presents some heuristic approaches to tackle the LARP, as well as some proposals for benchmark instances (and corresponding results). New constructive and improvement methods are presented and used within different metaheuristic frameworks. Test instances were obtained from the capacitated arc routing problem (CARP) literature and adapted to address the LARP.

1. Introduction

Location-routing problems (LRP) deal with the combination of two types of decisions that often arise: the location of facilities and the design of the distribution routes. While most LRP papers address node routing (see for example [1,2]), one may consider several scenarios where the demand is on the edges rather than being on the nodes of a network (usually a road network is assumed). These problems are referred to in the literature as location-arc routing problems (LARPs), and are derived from the capacitated arc routing problem (CARP) [3].

The LARP is typically overlooked in the literature. It has been shown that node routing problems can be converted into arc routing problems (the capacitated vehicle routing problem – CVRP – can be transformed into the CARP [4]), and that the reverse is also possible, replacing each arc by three [5] or two vertices [6,7], making the two classes of problems equivalent (the same holds true for their location counterparts: the capacitated LRP and the LARP).

Still, for the three transformations of the CARP into the CVRP, the resulting instance requires either fixing of variables or the use of edges with infinite cost. Moreover, the resulting CVRP graph is a complete graph of larger size. Hence, the transformation increases the problem size and the planar structure of a usual CARP graph is lost [8] dramatically changing the number of edges from linear to quadratic. The same can be extrapolated to the LARP, motivating its study using dedicated methods and algorithms.

The first work on the LARP, by Levy and Bodin [9], intended to tackle a practical problem arising in the scheduling of postal carriers in the United States postal service. The developed algorithm used the location–allocation–routing (L–A–R) concept described by [10] for the LRP, which includes three steps: firstly, depots are to be located using a depot selection procedure; secondly, arcs with demand are to be allocated to depots; thirdly, an Euler tour route of minimum traverse cost is determined for each set of arcs allocated to depots.

Ghiani and Laporte [11] addressed an undirected LARP, called the location rural postman problem, in which depots are to be located and routes to be drawn (serving edges with demand), at minimum cost, in an undirected graph. The authors show that the problem can be transformed into a rural postman problem if there is a single depot to open or no bounds on the number of depots. Using an exact branch-and-cut approach they solve the transformed problem.

In subsequent work by Ghiani and Laporte [3] a set of common applications for the LARP is mentioned (mail delivery, garbage collection and road maintenance). Furthermore the authors define the LARP as an extension of one of the three classical arc routing problems: the Chinese postman problem, the rural postman problem, and the CARP. The authors also present some insight into heuristic approaches using the decomposition of the problem into location (L), allocation (A) and routing (R) [10]: location–allocation–routing (L–A–R) and allocation–routing–location (A–R–L).

* Corresponding author. Tel.: +351 234370361.
E-mail addresses: rui.borges@ua.pt (R.B. Lopes), Frank.Plastria@vub.ac.be (F. Plastria), carlosf@ua.pt (C. Ferreira), bss@ua.pt (B.S. Santos).

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.10.003

Muyldermans [12] presents a variant of the LARP: the p dead-mileage problem. In this problem, unlike the previously addressed LARPs, splitting of the demand is allowed, that is, the client can be serviced more than once. The objective is to minimize dead-mileage (deadheading) and the problem is solved exactly.

Finally, the works by Pia and Filippi [13] and Amaya et al. [14] address variants of the CARP with a structure similar to the LARP, respectively, the CARP with mobile depots and the CARP with refill points. In the first, two different types of vehicles are considered: compactors and satellites. Compactors can be seen as mobile depots for the satellites. The second problem considers servicing of arcs by vehicles that must be refilled at certain nodes (to be determined) in order to complete the service.

From the previously mentioned variants, the LARP addressed here is the one studied by Ghiani and Laporte [11] which can be seen as the arc routing equivalent to the capacitated LRP, and thus an extension to the CARP.

In this paper some heuristic approaches are presented to tackle the LARP, as well as some proposals for benchmark instances (and corresponding results). Regarding the heuristic approaches new constructive and improvement methods are developed and used within different metaheuristic frameworks. The test instances were obtained from the CARP literature and adapted to address the LARP.

The remainder of this paper is outlined as follows. In Section 2 a formal definition of the problem is given. Constructive methods and improvement heuristics are presented in Section 3, and used within different metaheuristic frameworks proposed in Section 4. The developed test instances are addressed in Section 5 as well as the corresponding computational results. Finally, conclusions and future research directions are presented in Section 6.

2. Problem definition

The LARP consists of determining simultaneously depot location and routes in a graph in order to serve a specified set of required arcs under given operational constraints. Muyldermans [12] has shown that, for this problem, an optimal solution exists with the facilities located on the vertices of the graph.

Formally, the LARP can be described on a weighted and directed graph $G=(V, A)$ with vertex set V and set of arcs A. The vertex set V contains a non-empty subset J of m potential depot locations ($J \subseteq V$) with a fixed cost f_j and an associated capacity b_j ($j \in J$). Every arc $a=(i, j)$ in the arc set A has a non-negative traversal cost c_a and a non-negative demand for service d_a. The arcs with positive demand form the subset R of the arcs required to be serviced, only once, by a fleet K of identical vehicles with capacity Q. Vehicles start and end their route in the same depot, and each new vehicle (or route, as it is assumed that each vehicle performs a single route) involves a fixed cost F. The movement from the end i of one required arc to the start j of another required arc without servicing the traversed arcs (either required or not) is known as "deadheading", and has an associated cost denoted by z_{ij} (usually the cost of the shortest path in G from i to j).

The problem aims to determine the set of depots to be opened in J and the tracing of the distribution routes assigned to each open depot in such a way that the sum of fixed and traversal costs to serve all arcs in R is minimized.

Assuming G to be connected, it is possible to transform it into a complete graph $\hat{G}=(\hat{V}, \hat{A})$ where $\hat{V}$ is composed of the set V_R of vertices containing the extremities of the arcs in R ($V_R \subseteq V$), and J ($\hat{V}=V_R \cup J$). As $\hat{G}$ is a complete graph and $V_R \subseteq \hat{V}$, R is a subset of $\hat{A}$. Each arc $a=(i, j)$ in the arc set $\hat{A}$ has a non-negative cost $\hat{c}_a$ which takes on the value c_a if $a \in R$, z_{ij} otherwise.

For any subset S of vertices in $\hat{V}$, let $\delta^+(S)$ ($\delta^-(S)$), be the set of arcs leaving (entering) S, and $L(S)$ the set of arcs with both extremities in S. When S contains a single vertex v, $\delta^+(v)$ is a simplification for $\delta^+(\{v\})$. The following binary variables are used: x_{ak}, equal to 1 if and only if arc $a \in \hat{A}$ is used in the route performed by vehicle $k \in K$; y_j, equal to 1 if and only if depot j is to be opened; and w_{aj}, equal to 1 if and only if the arc $a \in R$ is assigned to depot j. The LARP can be formulated as:

$$\text{(LARP)} \quad \min \quad Z=\sum_{j\in J} f_j y_j+\sum_{a\in \hat{A}}\sum_{k\in K}\hat{c}_a x_{ak}+\sum_{k\in K}\sum_{a\in\delta^+(J)} F x_{ak} \tag{1}$$

$$\text{s.t.:} \quad \sum_{k\in K} x_{ak}=1 \quad \forall a\in R, \tag{2}$$

$$\sum_{a\in R} d_a x_{ak}\le Q \quad \forall k\in K, \tag{3}$$

$$\sum_{a\in\delta^+(i)} x_{ak}-\sum_{a\in\delta^-(i)} x_{ak}=0 \quad \forall i\in\hat{V},\ \forall k\in K, \tag{4}$$

$$\sum_{a\in\delta^+(J)} x_{ak}\le 1 \quad \forall k\in K, \tag{5}$$

$$\sum_{a\in L(S)} x_{ak}\le |S|-1 \quad \forall k\in K,\ \forall S\subseteq V_R, \tag{6}$$

$$\sum_{b\in\delta^+(j)\cap\delta^-(V_R)} x_{bk}+x_{ak}\le 1+w_{aj} \quad \forall a\in R,\ \forall j\in J,\ \forall k\in K, \tag{7}$$

$$\sum_{a\in R} d_a w_{aj}\le b_j y_j \quad \forall j\in J, \tag{8}$$

$$x_{ak}\in\{0,\ 1\} \quad \forall a\in\hat{A},\ \forall k\in K, \tag{9}$$

$$y_j\in\{0,\ 1\} \quad \forall j\in J, \tag{10}$$

$$w_{aj}\in\{0,\ 1\} \quad \forall a\in R,\ \forall j\in J. \tag{11}$$

The objective function (1) minimizes the sum of, respectively, the fixed costs of opening the depots, the costs of all traversed arcs, and the cost of acquiring vehicles. Constraints (2) ensure that each required arc is serviced once by exactly one vehicle. Capacity constraints are satisfied thanks to inequalities (3) and (8). Equalities (4) are the flow conservation constraints which, coupled with constraints (5), ensure the routes return to the departure depot. Constraints (6) are subtour elimination constraints while the set of constraints (7) specify that a required arc must be assigned to a depot in case there is a route linking them. Finally, constraints (9)–(11) define the variables. Note that integrality of w_{aj} can be relaxed to [0, 1] because if not pushed to 1 by (7) minimization will choose for 0 due to (8).

It can be noted that the LARP considered here can be seen as an extension of the CARP, where multiple depots are considered and an additional level of decision is for locating the depots.

3. Constructive methods and improvement heuristics

As the LARP results from the combination of a facility location problem and the CARP, both NP-hard problems [15,4], it is NP-hard. As a consequence, large instances can hardly be solved using exact methods; moreover, sharp bounds on the optimal value are typically hard to obtain. The best way to tackle these problems is then to use heuristic approaches [16] such as constructive methods and improvement heuristics.

Constructive methods are commonly used to obtain initial solutions from which improvement heuristics seek to attain better ones. Furthermore, these approaches are often used as the first step to many metaheuristics. In this section constructive methods (extended augment-merge and extended merge) and improvement heuristics

(reverse and relocate with both intra- and inter-route moves) are proposed to tackle the LARP.

3.1. Extended augment-merge

The augment-merge algorithm (illustrated in Fig. 1) was proposed for the CARP by Golden and Wong [4]. It starts with a trivial solution in which each arc with demand is serviced by a separate route. Then, after sorting the obtained routes in decreasing cost order and starting with the route with highest cost, it proceeds to the augment phase testing if the route already goes through demand arcs on less costly routes. If so, and provided vehicle capacity is preserved, the latter route is augmented into the former (the route with highest cost).

Afterwards the algorithm proceeds to the merge phase, where every feasible merge is evaluated for any two routes, merging the routes that provide the highest saving. This is done until no feasible saving exists. This last step is closely related to the well-known "savings" or Clarke and Wright algorithm [17].

The extended version (extended augment-merge – EAM) for the LARP obtains the initial solution by assigning, sequentially, each required arc to the closest depot in which it can fit, thus building a dedicated route. When all required arcs are assigned, the depots without demand to supply are closed. The augment phase is similar to the augment-merge algorithm, increasingly considering depot capacity constraints. In the merge phase of the EAM, the resulting route T, which may result from four different merges, is evaluated for reassignment to all depots (totaling $4m$ possible merges for each pair of routes). The resulting saving σ can be calculated as follows:

$$\sigma = F + z_{ri} + z_{jr} + z_{sk} + z_{ls} - z_{jk} - z_{ti} - z_{lt} + f_r\theta_r + f_s\theta_s - f_t(1-y_t). \quad (12)$$

θ_r (θ_s) is binary and equal to 1 if depot r (s), the depot of route R (S), supplies no more routes after the merger, and thus can be closed. y_t is a binary value (defined earlier for the formulation) equal to 1 if depot t (the depot currently evaluated to be assigned to T) is already open before the merge, and i, j, k, and l are the vertices of the arcs with demand which are connected to the depots in each route. This saving can be seen depicted in Fig. 2. The EAM ends when there is no more feasible merge with a positive saving. The time complexity of this method is $O(m|E|^2 \log |E|)$ due to the merge phase, where the list of $(m|E|^2)$ possible merges can be sorted using heapsort (the actual merging of the two routes can be done in linear time).

3.2. Extended merge

In the augment-merge algorithm the use of the augment phase has been contested [18]. If all the arcs in A are required to be serviced, it performs well, as the arcs absorbed by the higher cost routes are often contiguous. However, if this is not the case, the deadheading distance created between absorbed arcs cannot be recovered in the merge phase, leading to degraded results. Belenguer et al. [18] further support this statement by presenting overall better results for the algorithm without the augment phase.

As the EAM derives from the augment-merge, a similar situation may occur. This suggests the development of an extended merge (EM) algorithm for the LARP, similar to the EAM, but without performing the augment phase. The merge phase is used in both the EAM and the EM algorithms.

The extension of the merge phase (in both algorithms) uses some concepts from the extension to the savings algorithm proposed by Prins et al. [19] for the capacitated LRP. Similarly to the EAM method, EM can be executed in $O(m|E|^2 \log |E|)$ time.

3.3. Reverse

The reverse improvement heuristic [20] performs inside the routes and the corresponding move can be seen as the arc routing equivalent of the 2-opt move [21].

The reverse move is identical to the one used in the CARP (Fig. 3). The algorithm replaces a subsequence of arcs by the reverse, always ensuring the required arcs are serviced. This may lead to other shortest paths (in the deadheading distance) linking the required arcs. The algorithm implements the first found feasible move that improves the solution. This is done sequentially until no more feasible moves can be found.

3.4. Relocate

In the relocate improvement heuristic (well-known from the node routing context) two possible variations are considered: relocate inside the routes and relocate between two routes. In both variations the concept is to relocate a given arc (or subsequence of arcs) that requires service to another position in the route, or in another route.

The relocate algorithm inside the routes is based on the CVRP algorithm by Savelsbergh [22], while the inter-route algorithm is based on the work by Beullens et al. [20] for the CARP. In the latter, not only vehicle capacity, but also depot capacity constraints have to be taken into account. In both algorithms the (subsequence of) required arc(s) or its reversal is reinserted, depending on which of the two provides the largest improvement.

4. Metaheuristics

Metaheuristics are general combinatorial optimization techniques most useful and efficient in solving hard problems [23,24]. While in theory they can handle any combinatorial optimization problem, it is often the case that an important effort must be put on finding the right way to adapt the general parameters of these methods to a particular problem [16].

This is the case in this section, where three general metaheuristics (tabu search combined with a variable neighborhood search – TS–VNS; greedy randomized adaptive search procedure – GRASP;

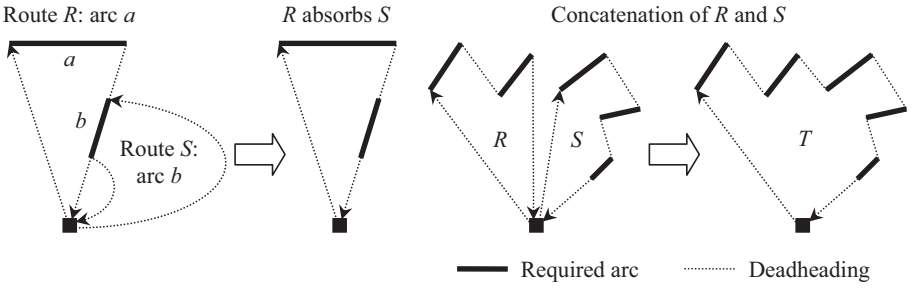

Fig. 1. Augment (left) and merge (right) moves in augment-merge [3].

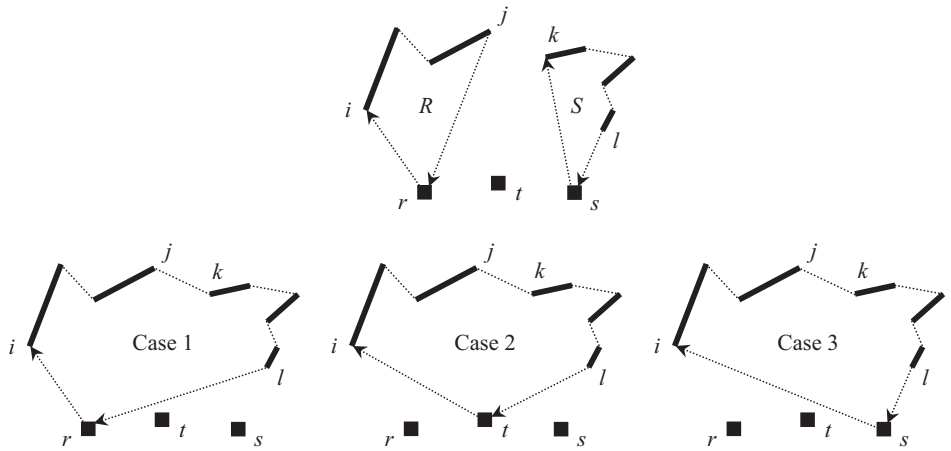

Fig. 2. Some merges in the merge phase of the EAM and EM algorithms.

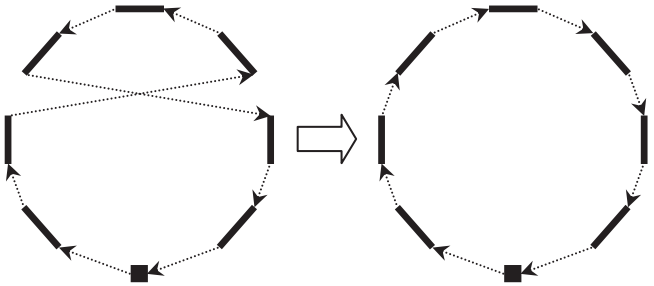

Fig. 3. The reverse move applied to a route.

and tabu search combined with a greedy randomized adaptive search procedure – TS–GRASP) are adapted and parameters tuned in order to tackle the LARP. The metaheuristics use the previously developed constructive and improvement methods in their specific framework.

4.1. Tabu search-variable neighborhood search

This approach (TS–VNS) is an iterative framework composed of a tabu search (TS) and a variable neighborhood search (VNS), respectively, for the location and (arc) routing phases. These two algorithms are performed iteratively until a stopping criterion is met, namely, a number *maxtsvns* of iterations without improvements to the solution (empirically found to be equal to 10). The TS–VNS approach starts by obtaining a solution using the VNS without constraints on the subset of depots *SD* to use ($SD \subseteq J$).

4.1.1. Variable neighborhood search

VNS is a metaheuristic proposed by Mladenović and Hansen [25] in which the main concept is to perform a systematic change of neighborhood within the local search, exploring increasing neighborhoods of the current solution. If an improvement is found, the search proceeds to the new solution and restarts. The steps of the basic VNS can be seen in Fig. 4.

In the adopted VNS: N_l is a finite set of pre-selected neighborhood structures inspired in the work by Polacek et al. [27] for the CARP; the initial solution is obtained by performing the EAM (preferred to EM since it is faster to obtain solutions, as the following steps behave similarly in both methods); and the stopping condition is a given number *maxvns* of iterations reached (equal to the number of arcs in the problem, multiplied by ten: $maxvns = 10|A|$).

The shaking step uses the CROSS-exchange operator proposed by Taillard et al. [28] for the VRP with time windows. It starts by randomly selecting two different routes, to which the CROSS-exchange operator is applied, swapping the sequence of consecutive required arcs. The number of required arcs which get swapped on each route is randomly obtained from an uniform distribution in the range $[1, \min(l, R_T)]$, R_T being the total number of required arcs for route T. When $l = l_{max}$ ($l_{max} = 6$, found empirically) the upper bound on the range is substituted by R_T. The described shaking step is biased to exchange smaller sequences of required arcs, while still allowing to perform large swaps.

The local search is applied on the two changed routes only, and is composed of the formerly proposed reverse and relocate improvement heuristics, performing intra-route (reverse and relocate) and inter-route (relocate) improvements, sequentially, until no additional improvement can be found. The newly obtained local optimum is only moved to when a cost reduction is obtained, thus making this a descent first improvement procedure which, according to Hansen and Mladenović [26], can be easily transformed into a descent–ascent method (although experimental trials with this variant proved unfruitful for the present approach).

At each iteration *it*, after obtaining the best solution S^{it} from the VNS using the subset of depots *SD* ($VNS(SD, S^{it})$), the TS–VNS approach proceeds to the location phase, performed by the TS algorithm.

4.1.2. Tabu search

The used TS algorithm is the one presented by Filho and Galvão [29], for the concentrator location problem, which provides near-optimal results in low CPU time, validating its use. TS was proposed by Glover [30] and has become one of the most widespread local search methods for combinatorial optimization. It uses a working memory called tabu list in which some attributes are stored (and forbidden to be used) for a number of moves.

In the used TS algorithm some adaptations were made to handle the LARP. Each route in the current best solution is collapsed into a single client (regarding demand), and the distance to the several feasible depots is the smallest insertion cost of the depot in the route, as in Barreto et al. [31]. The problem thus becomes a facility location problem and the algorithm tries to obtain the best depot location for the current routes ($TS(S^{it})$). Using the best obtained depot configuration (excluding the remaining depots from the problem), the TS–VNS approach proceeds to constructing the routes (routing phase) using the abovementioned VNS algorithm. If no improvement was found in the last (five) iteration(s), the approach provides some

Initialization. Select the set of neighborhood structures N_l, $l = 1,...,l_{max}$, that will be used in the search; find an initial solution S^i; choose a stopping condition.

Repeat the following steps until the stopping criteria is met.

(1) Set $l := 1$.

(2) Until $l = l_{max}$, repeat the following steps:

(a) *Shaking*. Generate a point S' at random from the l^{th} neighborhood of S^i ($S' \in N_l(S^i)$).

(b) *Local search*. Apply some local search method with S' as initial solution; denote with S'' the so obtained local optimum.

(c) *Move or not*. If this local optimum S'' is better than the incumbent S^i, move there ($S^i := S''$), and continue the search with N_1 ($l := 1$); otherwise, set $l := l + 1$.

Fig. 4. Steps of the basic VNS [26].

```
1:  it := 0
2:  ittsvns := 0
3:  SD := J
4:  cost(BestSol) := +∞
5:  repeat
6:     VNS(SD,S^it)
7:     TS(S^it)
8:     SD := depots(S^it) and small randomization
9:     if (cost(S^it) < cost(BestSol)) then
10:       BestSol := S^it
11:       ittsvns := 0
12:    else
13:       ittsvns := ittsvns + 1
14:    end if
15:    it := it + 1
16: until (ittsvns = maxtsvns)
17: return BestSol
```

Fig. 5. Overview of the proposed TS–VNS metaheuristic for the LARP.

diversification by randomly opening one (two) depot(s) and closing another from *SD*, always observing depot capacity constraints.

An overview of the pseudo code of the TS–VNS approach can be seen in Fig. 5. *ittsvns*, *BestSol*, *cost*(S^{it}), *cost*(*BestSol*), *depots*(S^{it}) respectively refers to the counter of successive iterations without improvement of the best solution, the best solution found, the cost of the current solution, the cost of the best solution found, and the depots to be opened in the current solution.

4.2. Greedy randomized adaptive search procedure

The EAM and EM are randomized to provide the greedy randomized search procedure producing the GRASP–EAM and GRASP–EM. The following features are valid for both variations and apply some concepts of the GRASP developed by Prins et al. [19] for the capacitated LRP.

For the randomized version of the constructive algorithms (*RCA*(*SD*, S^{it})) a restricted candidate list (RCL) of size φ is created from the savings calculated during the merge evaluations. The RCL contains the pairs of routes providing the best φ savings, from which one is randomly chosen to be performed. Changing the size of the RCL during the GRASP has been shown to often give better results [32]. So, at each merge, φ is randomly selected in [1, φ_{max}], where φ_{max} is the maximum RCL size allowed (found empirically: $\varphi_{max} = 7$).

The local search (LS), typically used in GRASP metaheuristics, is based on the reverse and relocate improvement heuristics previously presented, which performs intra-route (reverse and relocate) and inter-route (relocate) improvements, sequentially, until no additional improvement can be found.

Moreover, a learning process was included in the GRASP which reduces the computational time and improves the final solution [19]. The constructive algorithms, at each iteration *it*, provide a solution S^{it} which often has many open depots and, although the merges can close some, it may not be enough. In order to obtain better solutions, a subset *SD* of available depots is chosen to be used in the constructive algorithms ($SD \subseteq J$). In the first iteration of the GRASP all depots are used. Then, at each iteration, one of them is iteratively picked from *J* and the remaining depots in *SD* are randomly chosen (as well as their number) – *choose*(*SD*) – always ensuring there is enough capacity to serve all clients. This can be seen as a diversification.

Adding a memorization during the diversification mode enables the GRASP to learn about the good subset to open, and to possibly find better solutions. An intensification mode using this learning process was implemented, varying the GRASP iterations (using the boolean value *divmode* and reaching the maximum of $maxit = 75$) between:

- Diversification mode – applied for $maxdiv = 8$ iterations, in which the solution space is explored by varying the subset of open depots (as explained previously).
- Intensification mode – performed for $maxint = 7$ iterations, where an attempt is made to improve the routing for the selected depots (*SD*), obtained from the currently best found solution.

The parameters were set after a preliminary experimenting phase and allow five complete runs of the two modes returning, in the end, the best found solution (*BestSol*). Fig. 6 provides an overview of the GRASP approach, where *div* and *int* are the counters of successive iterations in, respectively, diversification mode and intensification mode.

4.3. Tabu search-greedy randomized adaptive search procedure

Attempting to integrate the best features of the previously described TS and GRASP, a TS–GRASP approach was developed. The TS handles the location phase while the GRASP addresses the (arc) routing phase. An iterative framework is used, in which the algorithms for both phases are performed iteratively until a stopping criterion is met (a number *maxtsgrasp* of iterations without improvements to the solution, empirically found: $maxtsgrasp = 10$). The required adaptations will be described below.

Similarly to the TS–VNS approach, the TS–GRASP starts by obtaining a solution for the routing phase without constraints on the subset of depots *SD* to use ($SD \subseteq J$), however, instead of the VNS, the following GRASP is used.

4.3.1. Greedy randomized adaptive search procedure

This GRASP uses the EM for the randomized constructive algorithm, as it provides bigger diversity and better results, returning a solution S^{it} at each iteration *it*. The randomization

```
1:  it := 0
2:  div := 0
3:  int := 0
4:  SD := J
5:  cost(BestSol) := +∞
6:  repeat
7:     RCA(SD,S^it)
8:     LS(S^it)
9:     if (cost(S^it) < cost(BestSol)) then
10:       BestSol := S^it
11:    end if
12:    if (divmode = true) then
13:       choose(SD)
14:       div := div + 1
15:       if (div = maxdiv) then
16:          SD := depots(S^it)
17:          divmode := false
18:          div := 0
19:       end if
20:    else
21:       int := int + 1
22:       if (int = maxint) then
23:          divmode := true
24:          int := 0
25:       end if
26:    end if
27:    it := it + 1
28: until (it = maxit)
29: return BestSol
```

Fig. 6. Overview of the proposed GRASP metaheuristic for the LARP.

performs similarly, using the RCL (with the same value of φ_{max}), as does the LS, with the reverse and relocate improvement heuristics (applied sequentially as long as there is improvement to the solution). Unlike the previous GRASP however, there is no diversification mode ($maxdiv = 0$), as the TS handles the location phase. Hence, this GRASP tries to obtain the best results (always in intensification mode) for the fixed depot configuration *SD*.

The *maxit* parameter also differs from the previous approach, varying it at each iteration according to:

$$maxit = \lceil 0.25 ittsgrasp \rceil + 1 \tag{13}$$

where *ittsgrasp* is the number of iterations without improvement. This further intensifies the search when fewer improvements are found (ensuring the GRASP is performed at least once).

The best obtained solution (*BestSol*) is used in the location phase (TS algorithm: *TS*(*BestSol*)) with which the TS–GRASP approach proceeds once the maximum number of iterations *maxit* is reached.

4.3.2. Tabu search

The TS algorithm in this approach works exactly as in the TS–VNS. Routes are collapsed into a single client and distances to depots are updated. Using the algorithm by Filho and Galvão [29], the best subset *SD* of depots to open is obtained (*depots*(*BestSol*)), to which a small diversification is then added: observing capacity constraints, if no improvement was found in the last (five) iteration(s), one (two) depot(s) is (are) randomly chosen to be opened and one to be closed.

The TS–GRASP approach then continues to the routing phase, restarting the GRASP and using only the new subset of depots *SD*, as seen in Fig. 7. In the figure, *newbestsol* is a boolean value indicating whether a new best solution was found in the routing phase.

5. Computational results

In order to ascertain the performance of the proposed methods and approaches for the LARP, experimental results were obtained.

```
1:  ittsgrasp := 0
2:  SD := J
3:  cost(BestSol) := +∞
4:  repeat
5:     it := 0
6:     newbestsol := false
7:     maxit := ⌈0.25 × ittsgrasp⌉ + 1
8:     repeat
9:        RCA(SD,S^it)
10:       LS(S^it)
11:       if (cost(S^it) < cost(BestSol)) then
12:          BestSol := S^it
13:          newbestsol := true
14:       end if
15:       it := it + 1
16:    until (it = maxit)
17:    TS(BestSol)
18:    SD := depots(BestSol) and small randomization
19:    if (newbestsol = true) then
20:       ittsgrasp := 0
21:    else
22:       ittsgrasp := ittsgrasp + 1
23:    end if
24: until (ittsgrasp = maxtsgrasp)
25: return BestSol
```

Fig. 7. Overview of the proposed TS–GRASP metaheuristic for the LARP.

Table 1
Data concerning the proposed instances for the LARP.

Instance		$\|V\|$	$\|E\|$	$\|R\|$	m	$\bar{f}$	Q	F
1	gdb-20	11	22	22	3	10	27	1
2	gdb-21	11	33	33	3	8	27	1
3	gdb-22	11	44	44	3	5	27	1
4	gdb-23	11	55	55	3	10	27	1
5	gdb-1	12	22	22	3	20	5	5
6	gdb-2	12	26	26	3	25	5	5
7	gdb-12	13	23	23	3	50	35	5
8	gdb-5	13	26	26	3	20	5	2
9	bccm-2A	24	34	34	5	20	180	5
10	bccm-2C	24	34	34	5	50	40	2
11	bccm-3A	24	35	35	5	15	80	2
12	bccm-3C	24	35	35	5	25	20	5
13	bccm-1B	24	39	39	5	15	120	2
14	bccm-1C	24	39	39	5	25	45	8
15	bccm-8A	30	63	63	5	10	200	5
16	bccm-6B	31	50	50	5	40	120	5
17	bccm-5A	34	65	65	5	20	220	5
18	bccm-5C	34	65	65	5	30	130	2
19	bccm-7A	40	66	66	5	5	200	2
20	bccm-4C	41	69	69	5	25	130	2
21	bccm-9B	50	92	92	5	30	175	5
22	bccm-10D	50	97	97	5	20	75	5
23	eglese-E1-A	77	98	51	10	250	305	20
24	eglese-E2-B	77	98	72	10	750	200	20
25	eglese-E3-C	77	98	87	10	1000	135	50
26	eglese-E4-B	77	98	98	10	650	180	75
27	eglese-S1-A	140	190	75	10	400	210	50
28	eglese-S2-B	140	190	147	10	2500	160	100
29	eglese-S3-C	140	190	159	10	1500	120	100
30	eglese-S4-A	140	190	190	10	1000	230	100

First, we discuss implementation and propose a new set of benchmark instances. This is followed by a comparative analysis on the algorithmic proposals using the newly devised instances.

5.1. Implementation and benchmark instances

All the aforementioned approaches were implemented in C♯ and the results obtained using a 3.00 GHz Intel Xeon E5450 Quad Core CPU with 8 GB of RAM and Windows XP (without parallel processing). To obtain the deadheading distances (z_{ij}), in both the constructive methods and improvement heuristics (and

consequently in the metaheuristic approaches), the well-known Dijkstra algorithm [33] was used.

A new set of instances is proposed in order to compare results and times (justified by the absence of benchmark instances in the literature). The instances were drawn from the CARP literature (the original sets can be found in http://www.uv.es/belengue/carp.html) and adapted to the LARP. These are the well-known and widely used instances from: Golden et al. [34], named "gdb"; Benavent et al. [35], called "bccm"; and Eglese [36], named "eglese". The first two sets were generated on a graph, with all edges being required, while the latter originates from a real-world (winter gritting) problem for the road network of Lancashire (UK). In the latter set of instances two graphs were obtained and different instances created by changing the set of required edges and the capacities of the vehicles.

From the previously mentioned CARP instances, some were chosen (promoting diversity) and adapted to support more than one depot and a cost structure in which location costs range from 30% to 70% of the total cost (with deadheading distance used as distribution costs). Table 1 displays basic data about the proposed instances named after the original CARP set followed by the instance number/name in the original set (e.g. eglese-E1-A is the instance "E1-A" from [36]). The first columns of the table display the name of the instance and the number of the vertices (V), edges (E), and required edges (R) (all instances use undirected graphs). Then follows the number of potential depot locations (m) and the average depot fixed cost ($\bar{f}$). Finally, columns "Q" and "F" refer, respectively, to the vehicles capacity and fixed cost. The proposed instances are available at http://lore.web.ua.pt/.

Table 2
Average and median results for the EAM and EM constructive methods with and without LS.

Method	CPU (s)		Gap_{LB} (%)	
	Average	Median	Average	Median
EAM	0.02	0.01	41.48	37.91
EAM+LS	0.02	0.01	38.18	35.93
EM	0.12	0.02	39.74	39.94
EM+LS	0.12	0.02	38.48	37.22

5.2. Comparative analysis

Results for the constructive methods (with and without the local search – LS – consisting of the improvement heuristics applied sequentially until no more improvements can be found) and for the metaheuristic approaches were obtained for the newly devised instances. For the constructive methods, a single run for each instance is performed (as there is no randomization), and the corresponding result and computing time found. For the metaheuristic approaches, twenty runs were made for each instance, from which the average and best result (and the time to obtain it) were acquired.

Average and median results for the EAM and EM constructive methods (with and without LS) can be seen in Table 2: the first column shows the method name, followed by average and median values of CPU times (in seconds) and gap to lower bounds Gap_{LB} (in percentage). The lower bounds were obtained by running the optimization solver CPLEX 12 where constraints (6) (see Section 2) were relaxed, including them iteratively, as they would get violated, until running out of memory (typically running for a few days). Table 3 shows the results regarding the metaheuristic

Table 3
Results for the TS–VNS, the GRASP (using EAM and EM) and the TS–GRASP metaheuristic approaches.

	Instance	LB	TS–VNS		Gap_{LB}	GRASP–EAM		Gap_{LB}	GRASP–EM		Gap_{LB}	TS–GRASP		Gap_{LB}
			Best	CPU		Best	CPU		Best	CPU		Best	CPU	
1	gdb-20	135	139	0.36	2.96	135	0.03	0.00	135	0.10	0.00	135	0.04	0.00
2	gdb-21	170	176	0.77	3.53	173	0.08	1.76	173	0.28	1.76	171	0.13	0.59
3	gdb-22	210	216	1.38	2.86	215	0.29	2.38	213	0.74	1.43	213	0.29	1.43
4	gdb-23	247	264	1.23	6.88	258	0.57	4.45	254	1.22	2.83	254	0.52	2.83
5	gdb-1	353	359	0.61	1.70	353	0.02	0.00	359	0.10	1.70	353	0.05	0.00
6	gdb-2	379	400	0.47	5.54	400	0.04	5.54	390	0.17	2.90	390	0.11	2.90
7	gdb-12	532	607	0.35	14.10	607	0.16	14.10	543	0.09	2.07	537	0.05	0.94
8	gdb-5	397	419	0.43	5.54	419	0.03	5.54	407	0.16	2.52	405	0.13	2.02
9	bccm-2A	247	249	2.02	0.81	255	0.05	3.24	247	0.40	0.00	247	0.18	0.00
10	bccm-2C	358	384	0.77	7.26	384	0.04	7.26	370	0.35	3.35	370	0.17	3.35
11	bccm-3A	96	99	1.17	3.13	97	0.06	1.04	96	0.45	0.00	96	0.20	0.00
12	bccm-3C	158	174	0.98	10.13	172	0.04	8.86	166	0.44	5.06	162	0.16	2.53
13	bccm-1B	209	220	1.19	5.26	216	0.09	3.35	211	0.59	0.96	211	0.39	0.96
14	bccm-1C	310	348	1.40	12.26	340	0.07	9.68	332	0.53	7.10	325	0.24	4.84
15	bccm-8A	404	433	5.41	7.18	427	0.56	5.69	425	3.07	5.20	424	0.97	4.95
16	bccm-6B	320	335	1.62	4.69	336	0.14	5.00	329	1.38	2.81	329	0.64	2.81
17	bccm-5A	441	464	2.95	5.22	459	0.56	4.08	456	3.02	3.40	450	1.41	2.04
18	bccm-5C	448	488	2.68	8.93	475	0.53	6.03	462	2.88	3.13	463	1.04	3.35
19	bccm-7A	290	304	4.69	4.83	302	0.44	4.14	297	3.01	2.41	298	1.18	2.76
20	bccm-4C	454	467	5.51	2.86	458	0.52	0.88	458	3.35	0.88	458	1.36	0.88
21	bccm-9B	392	415	5.60	5.87	413	1.30	5.36	406	9.52	3.57	408	4.78	4.08
22	bccm-10D	521	553	6.51	6.14	549	1.63	5.37	552	11.17	5.95	544	3.83	4.41
23	eglese-E1-A	2716	3256	6.16	19.88	3231	0.43	18.96	3014	2.07	10.97	2964	2.14	9.13
24	eglese-E2-B	5070	5811	4.97	14.62	5856	0.87	15.50	5584	5.29	10.14	5469	2.20	7.87
25	eglese-E3-C	7727	9147	5.07	18.38	9123	1.42	18.07	8566	10.31	10.86	8665	4.13	12.14
26	eglese-E4-B	7148	8017	7.90	12.16	8100	1.52	13.32	7751	13.40	8.44	7669	6.52	7.29
27	eglese-S1-A		4586	8.06		4404	1.10		4378	8.16		4315	2.78	
28	eglese-S2-B		15,820	33.21		15,592	6.76		15,307	55.04		15,069	22.64	
29	eglese-S3-C		17,090	40.96		16,933	7.88		16,109	65.01		16,029	24.21	
30	eglese-S4-A		12,722	24.29		12,491	10.76		11,977	97.13		11,879	52.35	
			Average	5.96	7.41		1.27	6.52		9.98	3.82		4.49	3.23
			Median	2.35	5.70		0.44	5.37		1.73	2.87		0.81	2.79

approaches (TS–VNS, GRASP–EAM, GRASP–EM, and TS–GRASP): the first columns display the instances name, followed by the lower bounds (LB). For each algorithm are also shown: the best results – Best; CPU time, in seconds; and the gap (Gap_{LB}), in percentage, between the lower bound and the algorithm best result. Average and median values are provided for CPU time and Gap_{LB}, as skewed distributions and/or outlying data points were obtained.

Looking at the results for the constructive methods, it can be concluded that, overall, EM performs better than EAM. This may lead to conclude that the claim by Belenger et al. [18] for the CARP, suggesting the augment phase generally leads to poorer results, is valid for the LARP. However, as the EAM obtains faster results (and with somewhat similar final results) it may be an interesting constructive method to be used in metaheuristics.

Regarding the metaheuristic approaches, computing times are, on average, less than 10 s, the GRASP–EAM being the fastest, followed by the TS–GRASP, the TS–VNS, and the GRASP–EM (although the difference among them is negligible as this is a strategic problem).

The TS–VNS approach presents the worst results which, looking at the results of the TS–GRASP, suggest that this is due to the VNS (used in the routing phase) not performing as well as the GRASP on the route building/improvement. Comparing the results of the two GRASP (GRASP–EAM and GRASP–EM), GRASP–EM performs better by providing an average as well as a median improvement of around 2.5%. This may be due to the approach allowing a greater diversity in the possible merges by not having an augment phase, thus easing the finding of better solutions.

Looking at the TS–GRASP, when compared with the GRASP–EM results, it can be concluded that the TS, used in the location phase, performs better than the diversity mechanism used in the choosing of open depots in the GRASP (as the route building works similarly). Moreover, this difference is further stressed in the last eight instances, which have more possible depot locations. Thus, as the TS–GRASP has a better location algorithm, it does not waste so much time trying to obtain the best depot configuration, resulting in smaller computing times and allowing further intensification of the routing phase (enabling to find better results).

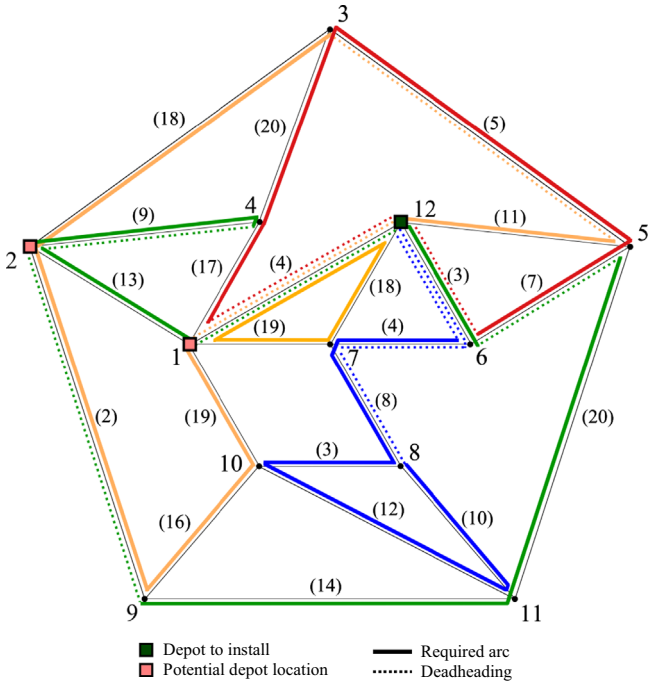

Fig. 8. Graphical representation of an optimal solution for instance "gdb-1".

A graphical representation of an optimal solution for instance "gdb-1" can be seen in Fig. 8, where routes are displayed using different colors and a single depot is to be installed (in node 12).

6. Summary and conclusions

In this paper heuristic approaches to tackle the LARP, as well as some proposals for benchmark instances (and corresponding results) are presented. Regarding the heuristic approaches new constructive and improvement methods are proposed and used within different metaheuristic frameworks (two constructive methods, three heuristic improvements and, using these, four metaheuristic approaches). Due to the lack of benchmark instances in the literature a new set of instances was devised, adapted from the CARP literature.

Concerning the proposed heuristics, the TS–GRASP outperformed the other approaches results wise, and was extremely competitive regarding computing times.

A limitation of this work is that conclusions on the proposed approaches were based on the results obtained for the newly devised set of benchmark instances which, in some cases, have high gap values; yet, the computational analysis of the results allowed to validate the proposed instances as they appear to be balanced (regarding location and routing costs) and representative of several different cost configurations.

Notice that the proposed approaches use the same local search methods (although within different frameworks), and thus the obtained upper bounds may be not entirely independent. On the other hand, the quality assessment of the presented results is based upon the lower bound values obtained using a commercial software package.

As future work, the development of new sets of benchmark instances may prove useful, either based on real-world problems, or generated using a structure that allows to draw conclusions on which real-world situations may be tackled with a specific approach. Finally, new methods for obtaining lower and upper bounds to tackle the LARP should be developed as this is a highly applicable problem which requires dedicated methods and algorithms.

References

[1] Nagy G, Salhi S. Location-routing: issues, models and methods. Eur J Oper Res 2007;177(2):649–72.
[2] Lopes RB, Ferreira C, Santos BS, Barreto S. A taxonomical analysis, current methods and objectives on location-routing problems. Intl Trans Oper Res 2013;20(6):795–822, http://dx.doi.org/10.1111/itor.12032.
[3] Ghiani G, Laporte G. Location-arc routing problems. Opsearch 2001;38 (2):151–9.
[4] Golden BL, Wong RT. Capacitated arc routing problems. Networks 1981;11 (3):305–15.
[5] Pearn WL, Assad A, Golden BL. Transforming arc routing into node routing problems. Comput Oper Res 1987;14(4):285–8.
[6] Baldacci R, Maniezzo V. Exact methods based on node-routing formulations for undirected arc-routing problems. Networks 2006;47(1):52–60.
[7] Longo H, Aragão MPd, Uchoa E. Solving capacitated arc routing problems using a transformation to the CVRP. Comput Oper Res 2006;33(6):1823–37.
[8] Wøhlk S. A decade of capacitated arc routing. In: Golden BL, Raghavan S, Wasil E, editors. The vehicle routing problem: latest advances and new challenges. New York: Springer; 2008. p. 29–48.
[9] Levy L, Bodin L. The arc oriented location routing problem. INFOR 1989;27 (1):74–94.
[10] Laporte G. Location-routing problems. In: Golden BL, Assad AA, editors. Vehicle routing: methods and studies. Amsterdam: North-Holland; 1988. p. 163–98.
[11] Ghiani G, Laporte G. Eulerian location problems. Networks 1999;34 (4):291–302.
[12] Muyldermans L. Routing, districting and location for arc traversal problems [Ph.D. thesis]. Leuven: Katholieke Universiteit Leuven; 2003.
[13] Pia AD, Filippi C. A variable neighborhood descent algorithm for a real waste collection problem with mobile depots. Intl Trans Oper Res 2006;13 (2):125–41.

[14] Amaya A, Langevin A, Trépanier M. The capacitated arc routing problem with refill points. Oper Res Lett 2007;35(1):45–53.
[15] Mirchandani PB, Francis RL, editors. Discrete location theory. New York: Wiley; 1990.
[16] Hertz A, Widmer M. Guidelines for the use of meta-heuristics in combinatorial optimization. Eur J Oper Res 2003;151(2):247–52.
[17] Clarke G, Wright J. Scheduling of vehicles from a central depot to a number of delivery points. Oper Res 1964;12(4):568–81.
[18] Belenguer JM, Benavent E, Lacomme P, Prins C. Lower and upper bounds for the mixed capacitated arc routing problem. Comput Oper Res 2006;33 (12):3363–83.
[19] Prins C, Prodhon C, Wolfler Calvo R. Solving the capacitated location-routing problem by a GRASP complemented by a learning process and a path relinking. 4OR 2006;4(3):221–38.
[20] Beullens P, Muyldermans L, Cattrysse D, Van Oudheusden D. A guided local search heuristic for the capacitated arc routing problem. Eur J Oper Res 2003;147(3):629–43.
[21] Lin S, Kernighan BW. An effective heuristic algorithm for the traveling salesman problem. Oper Res 1973;21(2):498–516.
[22] Savelsbergh M. The vehicle routing problem with time windows: minimizing route duration. INFORMS J Comput 1992;4(2):146–54.
[23] Glover FW, Kochenberger GA, editors. Handbook of metaheuristics. Norwell: Kluwer Academic Publishers; 2003.
[24] Talbi EG. Metaheuristics: from design to implementation. Hoboken: Wiley; 2009.
[25] Mladenović N, Hansen P. Variable neighborhood search. Comput Oper Res 1997;24(11):1097–100.
[26] Hansen P, Mladenović N. Variable neighborhood search: principles and applications. Eur J Oper Res 2001;130(3):449–67.
[27] Polacek M, Doerner KF, Hartl RF, Maniezzo V. A variable neighborhood search for the capacitated arc routing problem with intermediate facilities. J Heuristics 2008;14(5):405–23.
[28] Taillard É, Badeau P, Gendreau M, Guertin F, Potvin JY. A tabu search heuristic for the vehicle routing problem with soft time windows. Transp Sci 1997;31(2): 170–186.
[29] Filho VJMF, Galvão RD. A tabu search heuristic for the concentrator location problem. Locat Sci 1998;6(1):189–209.
[30] Glover F. Future paths for integer programming and links to artificial intelligence. Comput Oper Res 1986;13(5):533–49.
[31] Barreto S, Ferreira C, Paixão J, Santos BS. Using clustering analysis in a capacitated location-routing problem. Eur J Oper Res 2007;179(3):968–77.
[32] Resende MGC, Ribeiro CC. Greedy randomized adaptive search procedures. In: Glover FW, Kochenberger GA, editors. Handbook of metaheuristics. Norwell: Kluwer Academic Publishers; 2003. p. 219–49.
[33] Dijkstra EW. A note on two problems in connexion with graphs. Numerische Math 1959;1(1):269–71.
[34] Golden BL, DeArmon JS, Baker EK. Computational experiments with algorithms for a class of routing problems. Comput Oper Res 1983;10(1):47–59.
[35] Benavent E, Campos V, Corberán Á, Mota E. The capacitated chinese postman problem: lower bounds. Networks 1992;22(7):669–90.
[36] Eglese RW. Routeing winter gritting vehicles. Discrete Appl Math 1994;48 (3):231–44.