European Journal of Operational Research 255 (2016) 68–79

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Production, Manufacturing and Logistics

Local search heuristics for sectoring routing in a household waste collection context

Maria João Cortinhal[b,c,*], Maria Cândida Mourão[a,c], Ana Catarina Nunes[b,c]

[a] Instituto Superior de Economia e Gestão, Universidade de Lisboa, Rua do Quelhas 6, 1200-781 Lisboa, Portugal
[b] ISCTE-IUL – Instituto Universitário de Lisboa, Av. das Forças Armadas, 1649-026 Lisboa, Portugal
[c] CMAF-CIO, Universidade de Lisboa, 1749-016 Lisboa, Portugal

A R T I C L E I N F O

Article history:
Received 3 February 2015
Accepted 7 April 2016
Available online 20 April 2016

Keywords:
Routing
District design
Heuristics
Capacitated arc routing

A B S T R A C T

This paper addresses the problem of residential waste collection, as a real life application of a sectoring-arc routing problem (SARP). Tactical decisions comprise the partition of the service territory into a number of sectors so that each sector can be covered by a set of vehicle trips. In addition, operational decisions involving the design of the vehicle trips that minimize total routing time are to be made. Apart from supporting good vehicle trips, sectors should also be planned such that the workload time imbalance as well as the number of connected components are minimized. These later try to promote service areas (sectors) geographically concentrated and grouped into delimited regions. We propose two local search methods: a hill climbing and a tabu search based heuristic. A constructive heuristic for obtaining an initial solution is also suggested. By means of a normalized weighted sum of criteria for evaluating solutions, the local search heuristics were tailored to improve the features of the initial solution. The algorithms are tested on random instances and also on real life based instances. The results show that the proposed local search methods are an efficient way of obtaining good quality solutions to implement in practice. Results also highlight that the proposed function for evaluating solutions during the search phase plays an essential role.

1. Introduction

Waste collection systems cover different types of waste, such as residential, commercial, recyclable or skip waste. Depending on the characteristics and location of the waste containers, vehicle trips are tackled via node or arc routing problems.

This paper addresses a residential waste collection problem, in which waste is collected along the streets by a fixed number of capacitated vehicles. Thus, a capacitated arc routing approach is considered. Moreover, depending on streets size and on traffic rules, some streets can only be serviced in one direction, whereas others demand collection on both sides and in both directions. Large one-way streets need to be represented by multiple segments, and consequently the street network is represented by a mixed multi-graph.

This research was motivated by real life applications. Therefore, the design of vehicle trips is threefold. Firstly, it should minimize the total routing time, which represents the time required for the waste collection. Secondly, the street network must be partitioned into a given number of sub-regions (also called *sectors*, or *service areas*), which should be balanced and as connected as possible (i.e. with a small number of connected components). Thirdly, each sector must be serviced by a single capacitated vehicle that performs one or more trips within a limited workload time, due to labor regulations. The balance criterion is meant to reduce the differences among the work assigned to the vehicle crews. It should be noted that improving the connectivity and workload time balance of the sectors promotes solutions in which service areas are both geographically concentrated, grouped into delimited regions, and with similar services for different vehicles. Service connectivity and concentration enable not only specialization, but also the allocation of responsibilities to the vehicle crews.

Given that no bounds are known to limit neither the number of connected components nor the workload time imbalance, tackling together these two features gives rise to some challenging issues. For instance, if adding up a new task to a given sector is ruled by connectivity, it is enough to guarantee that the new task shares at least one node with those tasks already assigned to that sector. On

* Corresponding author. Tel.: +351 21 790 30 00; Fax: +351 21 796 4710.
E-mail addresses: maria.cortinhal@iscte.pt (M.J. Cortinhal), cmourao@iseg.ulisboa.pt (M.C. Mourão), catarina.nunes@iscte.pt (A.C. Nunes).

http://dx.doi.org/10.1016/j.ejor.2016.04.013
0377-2217/© 2016 Elsevier B.V. All rights reserved.

the other hand, sectors can be built up simultaneously and ruled by the workload time if their balancing is required.

Note that these two rules can be incompatible if, e.g. the sector with the smallest workload time has no unassigned adjacent tasks (i.e. unassigned tasks linked with tasks in that sector). For that reason, and despite this research stems from the one presented in Mourão, Nunes, and Prins (2009), constructive heuristics such as the ones suggested in the aforementioned paper, even if adapted, lack to produce effective solutions for the problem under analysis.

Alternatively, we propose two local search algorithms, one of which is based on hill climbing, and the other is a tabu search, being the initial solution provided by an improved constructive method. To simultaneously deal with the three optimization criteria (total routing time, imbalance and connectivity within service areas) both local search algorithms make use of a weighted normalized function to evaluate solutions during the search process.

The problem under analysis can be modeled as a sectoring-arc routing problem (SARP) (Mourão et al., 2009) with additional features required for the solutions. This is a NP-hard problem since it combines a sectoring problem (also known as districting, district design, or territory design) with a mixed capacitated arc routing problem (MCARP), as the street network is a mixed one and vehicle capacities are limited. In fact, if only one sector is considered, it reduces to the MCARP that is itself a NP-hard problem since it extends the capacitated arc routing problem (CARP), introduced by Golden and Wong (1981) for undirected networks.

To the best of the authors' knowledge, the local search (LS) methods suggested in this paper have never been addressed to solve the SARP. For these LS methods we propose tailored moves that favor the connectivity of the service areas. Moreover, and following the literature on what concerns evaluation measures to similar problems, we also analyze the benefits of integrating different measures for evaluating solutions during the search process. The computational results show that the generated solutions are attractive from a practical point of view.

The remainder of this paper is organized as follows. In Section 2 we review the related literature and highlight the position of this research, and in Section 3 the problem is described. The local search solution methods are detailed in Section 4. Lastly, the computational results are summarized and analyzed in Section 5 before the conclusions, which are given in Section 6.

2. Literature review

The construction of trips has been widely addressed in the literature, most of which is devoted to node routing approaches, as may be confirmed in Golden, Raghavan, and Wasil (2008), and in Toth and Vigo (2014). Extensive surveys regarding arc routing can be found in Dror (2000), Perrier, Langevin, and Campbell (2006a), Perrier, Langevin, and Campbell (2006b), Wøhlk (2008), Corberán and Prins (2010), and Corberán and Laporte (2014).

The design of sectors has been considered for multiple purposes, such as political districting (Bação, Lobo, & Painho, 2005; Bozkaya, Erkut, & Laporte, 2003), commercial territory design (Jarrah & Bard, 2012; Ríos-Mercado & Fernández, 2009; Salazar-Aguilar, Ríos-Mercado, González-Velarde, & Molina, 2012), road maintenance (Muyldermans, Cattrysse, & Van Oudheusden, 2003; Perrier, Langevin, & Campbell, 2008), meter reading (Assis, Franca & Usberti, 2014), and also waste collection (Constantino, Gouveia, Mourão, & Nunes, 2015; Hanafi, Freville, & Vaca, 1999; Lin & Kao, 2008; Male & Liebman, 1978; Mourão et al., 2009; Teixeira, Antunes, & Sousa, 2004).

Some routing problems, as those related with road maintenance or waste collection, require not only the design of sectors, but also sectors with some specific features such as connectivity and balancing. Sectors balance is a feature which is required to obtain sectors similar in size or in the workload required.

Several balance measures have been proposed, such as the ones based on: (i) trips duration (Hanafi et al., 1999; Kim, Kim, & Sahoo, 2006; Mourão et al., 2009) or on their estimates (Gonzalez-Ramírez, Smith, Askin, Miranda, & Sánchez, 2011); (ii) the length of the links serviced (Perrier et al., 2008); (iii) the quantity serviced (Male & Liebman, 1978; Mourgaya & Vanderbeck, 2007; Salazar-Aguilar et al., 2012); (iv) a relationship between quantity and traveled distance (Lin & Kao, 2008; Teixeira et al., 2004), or; (v) on the number of customers (Salazar-Aguilar et al., 2012). Despite being different, all these measures are meant to evaluate the workload assigned to each sector. Therefore, sectors balance increases to some extent the chance of a fair distribution of workload amongst the different crew teams.

Sectors balance can be promoted through different ways, namely: (i) by constraints that impose upper bounds on cost (Haugland, Ho, & Laporte, 2007), on demand (Mourgaya & Vanderbeck, 2007), or on the length of each sector (Perrier et al., 2008); (ii) by tailored evaluation functions (Assis et al., 2014; Gonzalez-Ramírez et al., 2011; Hanafi et al., 1999; Lin & Kao, 2008; Ríos-Mercado & Fernández, 2009; Salazar-Aguilar et al., 2012), or; (iii) by the solution method as a whole (Kim et al., 2006; Male & Liebman, 1978; Mourão et al., 2009; Teixeira et al., 2004).

The balancing requirement is also referenced in the routing literature that solely discusses the construction of vehicle trips. For instance, both Jozefowiez, Semet, and Talbi (2009) and Oyola and Løkketangen (2014) address the capacitated vehicle routing problem with trip balancing. In this extension of the problem, two minimization objectives are tackled: the difference between the longest and the shortest trip length, and also the usual total length.

The connectivity within each sector is related to the contiguity of its demand units (i.e. units to be serviced and usually represented by nodes or links in a network), and therefore to the possibility of reaching each other within their service area. For this purpose, some authors suggest heuristics that consider specific rules to assign demand units to sectors, which additionally favor the concentration of each service area in a geographical subregion (Hanafi et al., 1999; Haugland et al., 2007; Mourão et al., 2009; Muyldermans et al., 2003). Additionally, Constantino et al. (2015) propose exact models and heuristics to attain the concentration as well as the connectivity of the service zones.

Other authors consider connectivity explicitly as a constraint, as is the case of the local search based methods of Ríos-Mercado and Fernández (2009) and Lei, Laporte, and Guo (2012). On the other hand, the heuristic solution methods proposed by Perrier et al. (2008) consist of solving mixed integer linear programming (MILP) models in which connectivity is explicitly imposed by linear constraints.

As previously mentioned, the SARP addressed in this paper is a problem which involves simultaneously tactical and operational decisions namely, sectors design and the planning of vehicle trips. The idea is to avoid successive sub-optimization by embedding operational activity measures, or their estimates, in strategic or tactical decisions (see e.g. Salhi & Rand, 1989; Simchi-Levi, 1992; Cattrysse, Van Oudheusden, & Lotan, 1997; Ghiani & Laporte, 2001). A survey on waste management systems mainly focused on strategic and tactical issues is due to Ghiani, Laganà, Manni, Musmanno, and Vigo (2014).

A few studies combine the design of service areas with the definition of trips, as is the case of Teixeira et al. (2004), Kim et al. (2006), or Ramos and Oliveira (2011) for node routing applications, and Male and Liebman (1978), Mourão et al. (2009), or Constantino et al. (2015) for the arc routing case.

The heuristic solution approach of Teixeira et al. (2004) for a recyclable waste collection case study is a constructive three phase method which aims to minimize operation costs. Firstly, a zone per vehicle is defined, taking the area and the population into account, in order to balance their collection effort. Then, for each sector, the last two phases determine the type of waste to be collected, and a single trip for each day of the month.

Kim et al. (2006) address a case study for a commercial waste collection vehicle routing problem with time windows. The aim is to group stops (collection points) into clusters, and to then build a single vehicle trip for each cluster, whilst minimizing total travel time. The authors propose an extended insertion algorithm and a clustering-based algorithm, both of which are enhanced with a simulated annealing improvement method. The clustering-based algorithm promotes the workload balance of the routes and also improves the proximity among the stops of the same route.

In their recyclable waste collection case study, Ramos and Oliveira (2011) present a constructive heuristic which simultaneously obtains the service area for each depot and the vehicle trips. Based on an estimated collecting time, the main goals are the minimization of the distances and the balancing of workload at the depots.

Male and Liebman (1978) propose a constructive heuristic based on a decomposition of the network into cycles. This heuristic is designed to simultaneously create sectors and build vehicle trips for garbage collection. The service area is represented by a connected and undirected graph, and a single vehicle trip per sector must be performed by a capacitated vehicle. The objective is to minimize the total deadheading time whilst creating contiguous and balanced sectors.

Mourão et al. (2009) developed three heuristics for a sectoring arc routing problem. Sectors are limited in workload time and the aim is to minimize total routing time. The heuristics promote connectivity, as well as workload time balance among sectors. A best insertion method builds sectors and vehicle trips simultaneously. In the two-phase heuristics, the sectors are built in phase 1 by two alternative procedures, both of which consider workload time estimates. Phase 2 executes a MCARP heuristic to obtain the vehicle trips within each sector.

Focused on real life applications, Constantino et al. (2015) developed solution methods for the MCARP with limited overlapping of the vehicle service routes. The proposed exact and heuristic methods are based on MILP models, in which an upper bound is imposed on the number of nodes shared by different routes. This upper bound is previously obtained via MILP models that minimize the number of nodes shared by different vehicle services.

Although the design of vehicle trips might not be included, the design of service areas for routing approaches may benefit from considering routing estimates (trip cost or duration). To illustrate this, we refer to Haugland et al. (2007), Jarrah and Bard (2012), and Lei et al. (2012), all of whom study sector design methods for stochastic vehicle routing problems, or Hanafi et al. (1999), for an arc routing problem.

Relation with related research

We propose a solution methodology for partitioning a network into balanced and connected subregions and simultaneously building the vehicle trips within each service area such that the total routing time is minimized. As well as Male and Liebman (1978), Hanafi et al. (1999), Mourão et al. (2009), and Constantino et al. (2015), this research aims to construct sectors for household waste collection. Thus, arc routing problems are taken into account.

Male and Liebman (1978) and Constantino et al. (2015) consider a single vehicle trip per sector. Moreover, in Male and Liebman (1978) the undirected case is considered. Here, the network is mixed and several trips per sector are allowed, as vehicles have a limited capacity and the service assigned to each sector is time limited.

Table 1
Comparing existing literature.

References	# vehicles per sector	Network type	Vehicles capacity	Trips design	Evaluation function
Male and Liebman (1978)	1	Undirected	Yes	Yes	No
Hanafi et al. (1999)	–	Mixed	No	No	Yes
Mourão et al. (2009)	Several	Mixed	Yes	Yes	No
Constantino et al. (2015)	1	Mixed	Yes	Yes	No
This reference	Several	Mixed	Yes	Yes	Yes

In Hanafi et al. (1999) the vehicle trips are not obtained. The authors estimate the workload time of a single non capacitated vehicle trip per sector. To evaluate the workload time balance, both the differences between this estimate to a target value and to some given bounds are computed. Two methods are proposed, a tabu search and a simulated annealing. To evaluate solutions, two weighted functions are used, one of which incorporating solely the workload time balance, whereas the other also includes the number of connected components. Here, the total routing time is also included. Moreover, no target or reference values for the workload time imbalance are previously imposed or known, which in turn makes the problem harder to solve.

Lastly, Mourão et al. (2009) suggest constructive heuristics that are mainly focused in creating balanced sectors, thus leading to solutions with a high number of connected components. Instead, here we propose methods that give the same level of importance to the aforementioned criteria. While Mourão et al. (2009) only suggest constructive heuristics, we also propose local search methods, which significantly improve the quality of the solutions.

Table 1 summarizes the main differences among the aforementioned researches.

3. Problem description

The household waste collection problem under study can be described as follows. A network represents the street segments (*links*), some of which requiring service (waste collection), whilst others do not (no waste to collect). Each *required street* segment, which is also named as a *task*, is symbolized: (i) by one edge, if it is a narrow two-way street that allows simultaneous collection for each of its two sides (zigzag or parallel collection); (ii) by two opposite arcs, if it is a large two-way street, with each side collected separately; (iii) by one arc, if it is a one-way street, or; (iv) by two parallel arcs, if it refers to a large one-way street that must be serviced twice, once for each side. Each task has a *service time*, which measures the time for collecting its waste. As some required street segments need to be represented by edges, each edge $u = (i, j)$ is replaced by two inverse linked arcs, u and $inv(u) = (j, i)$ (Lacomme, Prins, & Ramdane-Chérif, 2001). Henceforward, we therefore refer to required edges as arcs.

The *non-required street* segments, or *deadheading links*, are always represented by one arc or two arcs, depending on whether they relate to a one-way or a two-way street, respectively. Furthermore, deadheading is allowed, which means that even required streets can be traversed without being serviced. Therefore, each arc has a *deadheading time*.

A homogeneous fleet of K vehicles is available, each with a limited capacity of W. The fleet is based at a single *depot*, which also represents the disposal facility (dumpsite) where the vehicles are emptied. The time required to empty a vehicle is referred to as *dump time*.

Let us define a *feasible vehicle trip* as a tour to and from the depot, including the service of some tasks within the vehicle's

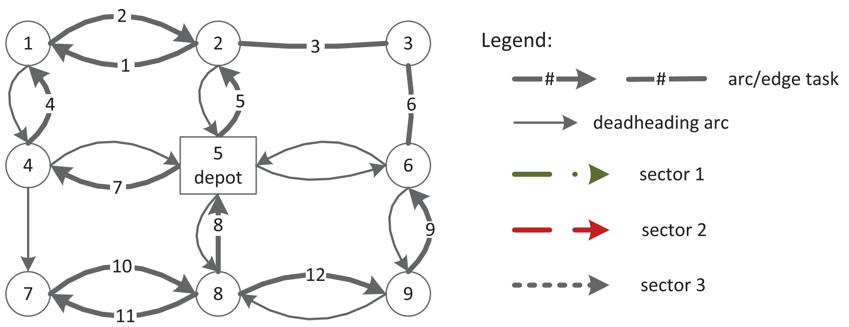


Fig. 1. Network.

capacity. Thus, a *vehicle service* is a set of feasible vehicle trips, and its *workload time* is the time required to complete it, which includes service, deadheading, and dump times. As a crew is assigned to each vehicle, and bearing in mind that labor regulations impose a limit on their working period, a fixed workload time limit of L per vehicle is considered.

Given the aforementioned conditions, the problem consists of determining a set of K sectors and a feasible vehicle service per sector, such that each task is serviced by one trip only, whilst minimizing the: (i) total routing time; (ii) workload time imbalance, and; (iii) the number of connected components in the subgraph induced by the tasks per sector.

Total routing time (*TT*) measures the time required to perform all the trips in all the sectors, which equates to the sum of the workload times over all the sectors. It should thus be minimized, in order to reduce operational costs. *Workload time imbalance* (*WIB*), on the other hand, evaluates the difference between the workload time of the longest and the shortest sectors. This measure is related to fairness among the services provided by the vehicle crews, as smaller values point to similar workload times assigned to different vehicle crews. Lastly, the *number of connected components* (*CC*) counts the number of connected components over all sectors. Its minimization increases connectivity, and thus it is used to pursue better designed solutions.

4. Local search heuristics

Local Search (LS) is a commonly used technique in combinatorial optimization that has proven to be effective for generating good solutions for a countless number of problems. Starting from a candidate solution, a LS algorithm iteratively moves to a neighbor solution in pursuit of a better one.

In this paper we propose two local search heuristics: a Hill Climbing (HC), and a Tabu Search (TS). Their main difference rely on the acceptance criteria of a new solution. HC only allows improving moves, and consequently a new solution is always a better solution. Additionally, TS also accepts some non-improving moves, depending on a short-term memory list that is updated during the search. Both HC and TS are tailored to improve the attractiveness of the solutions, whilst maintaining the good features of the initial solution.

The methods here proposed will be illustrated with the mixed network depicted in Fig. 1, where thinner links represent deadheading streets, and thicker ones stand for tasks. Numbers on links are used to identify tasks. We assume deadheading and service times of 1 and 2, respectively. Moreover, each task has an unitary demand and 3 vehicles are available each one with a capacity of 4 units. For the sake of simplicity, no dump time and no workload time limit are considered.

In this section, we first present the function to evaluate solutions. We then explain the constructive heuristic to generate the initial solution, needed for the LS algorithms. After, the moves as well as the neighborhood structures that are common to both LS heuristics are described. The detail of the two LS heuristics ends the section.

4.1. Solution evaluation

To pursue with the objective of creating connected and balanced sectors whilst minimizing the routing time, during the search process each incumbent solution S is evaluated by the following normalized weighted sum:

$$Eval(S) = \beta_{TT} * \frac{TT(S) - LB}{UB_{TT} - LB} + \beta_{CC} * \frac{CC(S) - K}{UB_{CC} - K} + \beta_{WIB} * \frac{WIB(S)}{UB_{WIB}} \tag{1}$$

The $Eval(.)$ function combines three measures: (i) total routing time, $TT(.)$; (ii) number of connected components, $CC(.)$; and (iii) workload time imbalance $WIB(.)$.

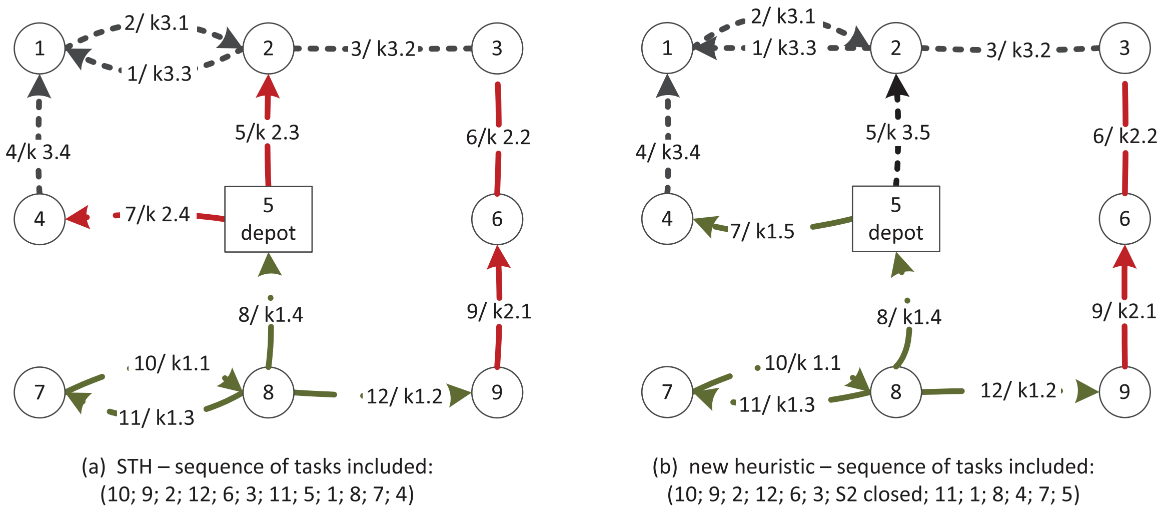
Better solutions thus present smaller values of $Eval(.)$. To ensure that all the measures have the same level of importance, the original values are scaled down to similar ranges, via lower bounds (LB and K), and upper bounds (UB_{TT}, UB_{CC} and UB_{WIB}). LB represents the best-known lower bound for the MCARP (Belenguer, Benavent, Lacomme, & Prins, 2006; Gouveia, Mourão, & Pinto, 2010), K is the given number of sectors, whereas UB_{TT}, UB_{CC}, and UB_{WIB} respectively measure the total routing time, the number of connected components, and the workload time imbalance of the initial solution.

Furthermore, each criterion is weighted by a 0/1 integer parameter, namely β_{TT}, β_{CC}, and β_{WIB}. We highlight that these parameters are meant to allow the user to opt for considering, or not, the corresponding criterion for evaluating solutions. For instance, if the workload time imbalance is a feature that is not required anymore it is enough to set β_{WIB} to 0, and the search will be focused on sectors that minimize the routing time and the number of connected components.

4.2. Construction of an initial solution

The initial solution $S0$ is built using a constructive heuristic based on a cluster-first/route-second approach. Firstly, the tasks are partitioned into clusters, and then the trips are built within each cluster, taking the capacity of the vehicles into account.

For the purpose of clarification, some definitions are firstly introduced. For each pair of arcs (x,y), $D_{x,y}$ is the shortest deadheading time (or distance) from x to y, excluding the deadheading times of x and y. As the network is mixed, and given that each required edge is represented by two linked inverse arcs, u and $inv(u)$, the time between two tasks u and v is measured by the U-distance U_{uv}, as proposed in Mourão et al. (2009, Section 2.3). The U-distance computes the minimum among at most eight shortest



Legend for arcs: (task number)/(sector).(order of inclusion in sector)

Fig. 2. Constructive methods – sectors.

deadheading times, as follows: $U_{uv} = \min\{D_{u,v};\ D_{u,inv(v)};\ D_{inv(u),v};\ D_{inv(u),inv(v)};\ D_{v,u};\ D_{v,inv(u)};\ D_{inv(v),u};\ D_{inv(v),inv(u)}\}$.

Given a sector k and a task u, the best insertion position of u in k is the insertion position of task u in sector k that least increases the workload time in the sector. This insertion position contemplates both existing trips, at any position within the vehicle capacity, and new trips in sector k.

A task u, in a sector $k1$, is *adjacent* to a task v, in a different sector $k2$, if u and v share at least one node. If such a pair of tasks (u, v) exists then sectors $k1$ and $k2$ are also said to be adjacent and task u (v) is adjacent to sector $k2$ ($k1$). Let also define $\Gamma(u)$ as the set of tasks adjacent to u.

Concerning a sector, it is said to be *opened* if tasks can still be added to it, i.e. if unassigned tasks adjacent to the sector still exist. Otherwise, the sector is *closed*.

The construction of the initial solution starts by selecting K distinct tasks to be the seed-tasks of each sector. The first seed-task is randomly selected. All the remaining are selected, one at a time, according to a *minmax* criterion, which minimizes the maximum U-distance to the previously chosen seed-tasks. This rule aims to spread the seed-tasks all over the entire network. Moreover, the status of each sector is considered opened, and the workload time estimate of each sector k, $\widetilde{W}(k)$, is computed.

At this point, $\widetilde{W}(k)$ simply measures the workload time of a vehicle trip servicing a single task (the seed-task) which thus includes the shortest deadheading time from the depot to and back the seed-task, the service time of the seed-task, and the dump time.

An iterative process is then applied for the assignment of the remaining tasks. At each iteration, the opened sector k with the smallest workload time estimate is selected for expansion. Then, the unassigned task adjacent to sector k, if any, that minimizes the U-distance to the seed-task is assigned to k. Once assigned, the task is inserted into its best insertion position and $\widetilde{W}(k)$ is updated. Otherwise, the status of sector k is set to closed, as no more unassigned tasks could ensure its connectivity. This procedure is repeated until all the tasks are assigned or all the sectors are closed.

If some tasks remain unassigned, which may happen if, e.g. the demand graph is disconnected, all the sectors are reopened. From this point on, and as before, the sector with minimum workload time estimate, say k, is selected and the unassigned task closest to its seed-task, u, is added and positioned in its best insertion position. In this case, a new connected component is considered in k. The unassigned tasks adjacent to u, if any, are next considered to be added to this new component in k.

It worth to remark that this heuristic does not guarantee neither feasibility (the workload time limit of some vehicles can be exceeded) nor balanced sectors (see Fig. 2(b)).

The new constructive heuristic here suggested, although similar to the STH of Mourão et al. (2009), may be considered as an improved method, as it better deals with the connectivity of the sectors. In fact, the main differences concern the task selection rule and the criterion for closing a sector. Regarding STH, in each iteration the unassigned task closest to the seed-task of sector k (adjacent or not to it) is selected and $\widetilde{W}(k)$ is recomputed. If $\widetilde{W}(k)$ does not exceed the time range L the task is assigned to sector k, otherwise sector k is considered closed, and the task remains unassigned. Therefore, STH is mainly focused on building balanced sectors whilst minimizing the routing time. Instead, the new heuristic guarantees connected sectors, if possible, whilst minimizing the routing time. Despite both features, balance and connectivity, are equally important we here prioritize the connectivity since it is a harder issue to deal with by means of a search process.

The three sectors for the previous example are depicted in Fig. 2, which illustrates the differences between STH (Fig. 2(a)) and the new constructive method here presented (Fig. 2(b)). Starting from the same set of seed-tasks (the first link included in each sector: $10/k1.1$ – task 10 in sector $k1$; $9/k2.1$ – task 9 in sector $k2$; and $2/k3.1$ – task 2 in sector $k3$), the resulting sectors are indeed different. As expected, the new heuristic provides connected sectors (see Fig. 2(b)) while STH yields better balanced but not connected sectors (see $k2$ in Fig. 2(a)).

Once the tasks are partitioned into sectors, the Improvement Merge (IM) heuristic is applied to built the vehicle trips within each sector. The IM heuristic was suggested by Belenguer et al. (2006) as an improvement of the extended augment merge (EAM) of Lacomme, Prins, and Ramdane-Chérif (2004), which, in turn, extends to the MCARP the classical augment-merge of Golden and Wong (1981).

Applying the IM to the sectors ($k1$, $k2$ and $k3$) identified by STH (Fig. 2(a)) originates a solution with one trip per sector. The workload times for each sector are respectively 10 (trip (11,10,12,8)), 12 (trip (5,6,9,7)), and 12 (trip (4,2,1,3)). From the sectors provided by the new heuristic (Fig. 2(b)) IM gets a feasible solution with a trip servicing two tasks (in $k2$), with a workload time of 8 (trip (9,6)), and two sectors ($k1$ and $k3$) with two trips each and workload times of 13 (trip 1 (11,10,12,8); trip 2 (7)), and 15 (trip 1 (4,2,1,3); trip 2 (5)), respectively. This example illustrates that STH tends to produce more balanced sectors than the new heuristic. However, it can build sectors which are not connected as in this case (in sector $k2$ tasks 5 and 7 are not linked to tasks 6 and 9).

4.3. Moves and neighborhood structures

Neighbor solutions are obtained by moving tasks between sectors. Each time a task is moved to a different sector it is positioned at its best insertion position. Note also that moving task u from sector $k1$ to an adjacent sector $k2$ will not increase the number of connected components in $k2$, but it may increase the number of connected components in $k1$. Thus, neighbor solutions have to be reevaluated by function *Eval* regarding all its terms, i.e. the routing and the imbalance time, as well as the number of connected components.

Two different moves are considered: swap and compound shift. Distinct moves demand for different neighborhood structures as next detailed. Henceforward, it is assumed a given solution S, with sectors identified by $k\#$ and tasks by u, v, x, y.

4.3.1. Swap moves

A *swap move*, $Swap(u, v)$, interchanges the sector to which tasks u and v belong to. Thus, if u is initially assigned to $k1$ and v to $k2$, u is moved to $k2$ whilst v is moved to $k1$. The neighborhood structure defined for these moves is named *CLSwap*. In order to avoid increasing the number of connected components, the pairs of tasks, $u \in k1$ and $v \in k2$, to be included in *CLSwap* must verify at least one of the following conditions:

(i) $u \in \Gamma(v)$;
(ii) $\exists(x \in k1 \wedge y \in k2) : x \in \Gamma(v) \wedge y \in \Gamma(u)$.

To illustrate these conditions let us consider Fig. 2(b). Some examples of tasks which may be swapped are next detailed.

A. $u = 5 \in k3 \wedge v = 7 \in k1$; as $5 \in \Gamma(7)$, thus tasks 5 and 7 satisfy (i);
B. picking the same tasks $u = 5 \wedge v = 7$, $\exists(x = 4 \in k3 \wedge y = 8 \in k1) : 4 \in \Gamma(7) \wedge 8 \in \Gamma(5)$, and thus tasks 5 and 7 also satisfy (ii);
C. for tasks $u = 4 \in k3 \wedge v = 8 \in k1$, $\exists(x = 5 \in k3 \wedge y = 7 \in k1) : 5 \in \Gamma(8) \wedge 7 \in \Gamma(4)$, and thus tasks 4 and 8 verify (ii).

Swap moves applied to adjacent tasks under the above conditions (i) and (ii) are used to try to identify neighbor solutions that keep, as much as possible, the number of connected components (cases A and B). However, swap moves may also increase the number of connected components in one sector (case C), or in both sectors as illustrated in Fig. 3. Although tasks u, $x \in k2$ and $v, y \in k1$ verify (ii), $Swap(u, v)$ increases the number of connected components in both sectors.

4.3.2. Compound shift moves

A *compound shift move*, $CShift(u, v)$, consists of a sequence of at most three shift moves. Again, the neighbor solutions are selected in order to try to maintain the connectivity. Thus, a sequence is built taking a pair of adjacent tasks ($u \in k1$ and $v \in k2$) into account. Let T be the trip that serves task v in $k2$. Firstly, the sequence order is randomly selected: forward or backward. Suppose

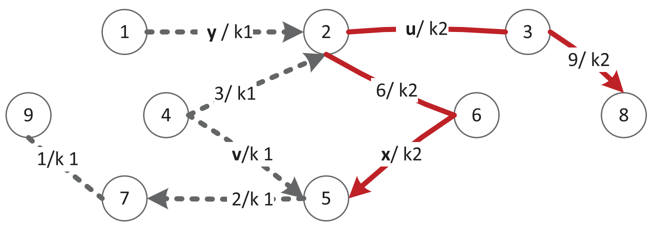


Fig. 3. Example of a swap move.

that the backward (forward) sequence is selected. Then, consider a sequence of tasks ($v2, v1, v$) where $v1$ is the immediate predecessor (successor) task of v and $v2$ is the immediate predecessor (successor) task of $v1$ in T. Note that a trip may include deadheading links between two consecutive tasks even if only tasks are being considered. The compound shift move $CShift(u, v)$ comprises the three following moves:

(i) *single shift*, $CSS(u, v)$: v is moved to $k1$;
(ii) *double shift*, $CDS(u, v, v1)$: $v1$ is moved to $k1$ after the single shift $CSS(u, v)$ has been performed;
(iii) *triple shift*, $CTS(u, v, v1, v2)$: $v2$ is moved to $k1$ after the double shift $CDS(u, v, v1)$ has been performed.

It should be remarked that $v1$ and/or $v2$ may not exist. Therefore, each compound shift move yields at most three neighbor solutions. Then, $CShift(u, v)$ provides the best of these neighbor solutions, which means the best one evaluated by *Eval*.

For a given solution S there will exist as many compound shift moves as the number of sequences of tasks in the aforementioned conditions. The list of all these sequences is named as *CLCShift*, and it represents the neighborhood structure for these moves.

Unlike swap moves, compound shift moves only oblige to recompute the number of connected components in one sector, more precisely in the sector from which tasks are removed.

Let us return to the example in Fig. 2(b), and suppose that (11,10,12,8) is a trip in $k1$. By picking $u = 9 \in k2$ adjacent to $v = 12 \in k1$ all the three backward shift moves can be performed, and the neighbor solution is the best among those provided by CSS(9,12), CDS(9,12,10), and CTS(9,12,10,11).

In this later example, none of the three shift moves increases the number of connected components. Instead, if we consider the adjacent tasks $u = 6 \in k2$ and $v = 3 \in k3$, and the sequence (3,1,2) from trip (4,2,1,3), the number of connected components will remain with moves CSS(6,3) and CDS(6,3,1) (see Fig. 4(a) and (b)), but it will increase with CTS(6,3,1,2) (see $k3$ in Fig. 4(c)).

4.4. Hill Climbing (HC) algorithm

HC is a hill climbing heuristic, which means that a neighbor solution is accepted only if it is better than the current one. The iterative search process is repeated until a maximum number of iterations (*MaxIt*) is reached, or until all the neighbor solutions of the current one are non-improving solutions. The output solution is the best one found during the search process. Algorithm 1 summarizes the general framework of HC.

4.5. Tabu Search (TS) algorithm

The major drawback of the HC here proposed is that it can easily be trapped in a local optimum. This problem can be diminished through more sophisticated methodologies, such as tabu search (Glover & Laguna, 1997). Tabu search escapes from local optima through an adaptive memory called *tabu list*, allowing non-improving moves.

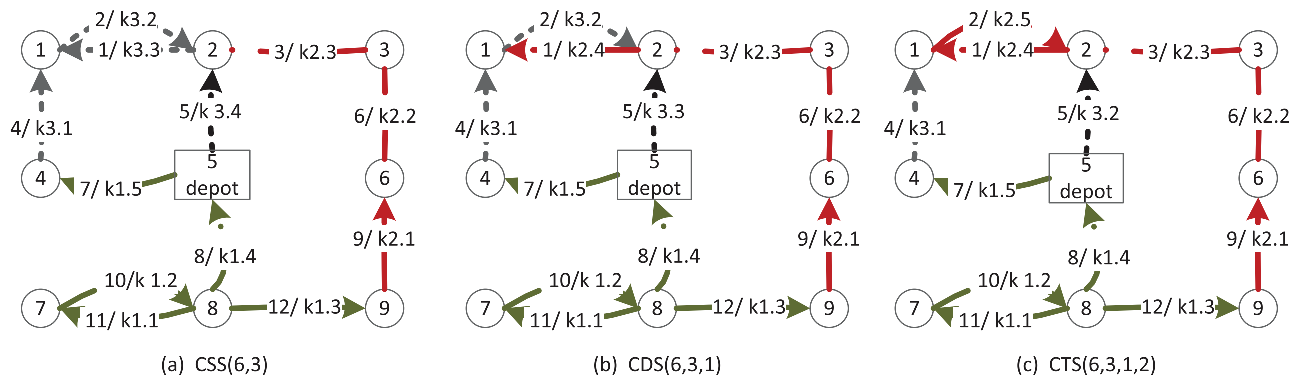


Fig. 4. Compound shift moves in example of Fig. 2(b).

Algorithm 1 HC algorithm.

```
Input: $S0$                                          ▷ Initial solution
Input: $MaxIt$                                       ▷ Maximum number of iterations
1: $S \leftarrow S0$
2: $Cost_S \leftarrow Eval(S0)$
3: $It \leftarrow 1$                                 ▷ Counter for iterations
4: repeat
5:     Create $CLSwap(S)$                            ▷ neighborhood of Swap moves
6:     Create $CLCShift(S)$                          ▷ neighborhood of CShift moves
7:     $Cost_{BestN} \leftarrow +\infty$             ▷ Initialize cost of best neighbor solution
8:     repeat
9:         Remove the first pair of tasks $(u, v)$ from $CLSwap(S)$
10:        $Neigh(S) \leftarrow Swap(u, v)$          ▷ Apply Swap move
11:        if $(Eval(Neigh(S)) < Cost_{BestN})$ then
12:            $Cost_{BestN} \leftarrow Eval(Neigh(S)), BestN \leftarrow Neigh(S)$
13:        end if
14:    until $(CLSwap(S)$ is empty$)$
15:    repeat
16:        Remove tasks $(u, v, v1, v2)$ from $CLCShift(S)$
17:        $Neigh(S) \leftarrow CShift(u, v)$        ▷ Apply CShift move
18:        if $(Eval(Neigh(S)) < Cost_{BestN})$ then
19:            $Cost_{BestN} \leftarrow Eval(Neigh(S)), BestN \leftarrow Neigh(S)$
20:        end if
21:    until $(CLCShift(S)$ is empty$)$
22:    $FlagMove \leftarrow true$
23:    if $Cost_{BestN} < Cost_S$ then
24:        $S \leftarrow BestN, Cost_S \leftarrow Cost_{BestN}$
25:        $FlagMove \leftarrow false$
26:    end if
27:    $It \leftarrow It + 1$
28: until $(FlagMove$ or $It > MaxIt)$
Output: $S$                                          ▷ Best solution found
```

In the proposed TS, the tabu status is applied to tasks, and a *tabu tenure* (*TabTen*) is set to define how long these tasks will remain with the tabu status. For instance, if task u is moved from $k1$ to $k2$ then it cannot be assigned to $k1$ once again for the next *TabTen* iterations. As a result, a move is tabu if it involves moving a tabu task. Nevertheless, tabu moves can be accepted if they lead to solutions that are better than the best one already found (aspiration criterion). Once the best neighbor is selected, the *TabuList* is updated accordingly. The stopping criteria are the maximum number of iterations (*MaxIt*), and the maximum number of consecutive iterations without improvement (*MaxImpIt*), and the output solution is the best one found during the search process. Algorithm 2 summarizes the general framework of the developed TS algorithm.

Table 2
Characteristics of the *slpr* instances.

Instance	Number of Nodes	Links	Required edges	Required arcs	Sectors
slpr-a-01	28	94	0	52	2
slpr-a-02	53	169	5	99	2
slpr-a-03	146	469	33	271	4
slpr-a-04	195	651	34	469	7
slpr-a-05	321	1056	58	748	12
slpr-b-01	28	63	5	45	2
slpr-b-02	53	117	9	92	2
slpr-b-03	163	361	26	279	5
slpr-b-04	248	582	8	493	8
slpr-b-05	401	876	37	764	13
slpr-c-01	28	52	39	11	2
slpr-c-02	53	101	77	23	2
slpr-c-03	163	316	241	61	6
slpr-c-04	277	604	362	142	9
slpr-c-05	369	841	387	416	14

5. Computational experiments

The proposed heuristics were coded in Delphi 7, and run on a 2.9 Gigahertz Intel CORE i7-3520M CPU with 8 Gigabyte RAM.

Computational experiments were performed on two sets of instances. The first set, hereafter named as *slpr*, is based on the *lpr* instances introduced by Belenguer et al. (2006) as random MCARP instances that mimic street networks. These *lpr* instances were then transformed into SARP ones by adding two parameters: the number of sectors (K); and the maximum workload within each sector (L), which is considered fixed and equal to 21,600 seconds. Moreover, it is assumed that a fleet of homogeneous vehicles, with capacity (W) equal to 10,000 kilograms, is available at the depot node.

Table 2 lists the characteristics of the 15 *slpr* instances, namely the number of nodes, links, required edges, required arcs and sectors. A more detailed description of these instances can be found in Mourão et al. (2009).

The second set, named as *seix*, includes nine instances (*seix*1 to *seix*9) generated from a refuse collection system in Seixal, a municipality near Lisboa, Portugal. These instances are all derived from the same street network, with data provided by the municipality,

Algorithm 2 TS algorithm.

```
Input: $S0$, $TabTen$, $MaxIt$, $MaxImpIt$   ▷ Initial solution, tabu tenure,
    and stopping criteria
1:  $TabuList = \{\}$
2:  $S, Best \leftarrow S0$
3:  $Cost_S, Cost_{Best} \leftarrow Eval(S)$
4:  $It, IterImp \leftarrow 1$   ▷ Counters for iterations
5:  repeat
6:      Create $CLSwap(S)$   ▷ neighborhood of Swap moves
7:      Create $CLCShift(S)$   ▷ neighborhood of CShift moves
8:      $Cost_{BestN} \leftarrow +\infty$   ▷ Initialize cost of best neighbor solution
9:      repeat
10:         Remove the first pair of tasks $(u, v)$ from $CLSwap(S)$
11:         $Neigh(S) \leftarrow Swap(u, v)$   ▷ Apply Swap move
12:         if ($Swap(u, v)$ not tabu) and
              ($Eval(Neigh(S)) < Cost_{BestN}$) then
13:             $Cost_{BestN} \leftarrow Eval(Neigh(S))$,$BestN \leftarrow Neigh(S)$
14:         else if ($Swap(u, v)$ is tabu) and
                  ($Eval(Neigh(S)) < Cost_{Best}$) then
15:             $Cost_{BestN} \leftarrow Eval(Neigh(S))$,$BestN \leftarrow Neigh(S)$
16:         end if
17:     until ($CLSwap(S)$ is empty)
18:     repeat
19:         Remove tasks $(u, v, v1, v2)$ from $CLCShift(S)$
20:         $Neigh(S) \leftarrow CShift(u, v)$   ▷ Apply CShift move
21:         if ($CShift(u, v)$ not tabu) and
              ($Eval(Neigh(S)) < Cost_{BestN}$) then
22:             $Cost_{BestN} \leftarrow Eval(Neigh(S))$,$BestN \leftarrow Neigh(S)$
23:         else if ($CShift(u, v)$ tabu) and
                  ($Eval(Neigh(S)) < Cost_{Best}$) then
24:             $Cost_{BestN} \leftarrow Eval(Neigh(S))$,$BestN \leftarrow Neigh(S)$
25:         end if
26:     until ($CLCShift$ is empty)
27:     $S \leftarrow BestN$,$Cost_S \leftarrow Cost_{BestN}$
28:     Update $TabuList$
29:     if $Cost_{BestN} < Cost_{Best}$ then
30:         $Best \leftarrow BestN$,$Cost_{Best} \leftarrow Cost_{BestN}$
31:         $IterImp \leftarrow 1$
32:     else
33:         $IterImp \leftarrow IterImp + 1$
34:     end if
35:     $It \leftarrow It + 1$
36: until ($It > MaxIt$ or $IterImp > MaxImpIt$)
Output: $Best$   ▷ Best solution found
```

and differ in the number of sectors, vehicle capacity, and workload time limit, as depicted in Table 3.

To evaluate the performance of the proposed heuristics, total routing time (*TT*) in seconds, number of connected components (*CC*), workload imbalance (*WIB*) in seconds, and CPU running time (*tcpu*) in seconds, are all presented. We also provide, whenever possible, some percentage gaps for the total routing time, namely LB_{gap}, and UB_{gap}. The lower bound gap is defined as $LB_{gap} = \frac{TT(H)-LB}{LB} \times 100$ percent, where *LB* accounts for the best-known MCARP lower bound (Belenguer et al., 2006; Gouveia et al., 2010), and $TT(H)$ for the total routing time of the solution provided by the respective heuristic *H*. The upper bound gap is given by $UB_{gap} = \frac{TT(S0)-TT(H)}{TT(S0)} \times 100$ percent, with $TT(S0)$ being the total routing time of the initial solution.

5.1. Parameters tuning

The TS algorithm demands for some input parameters, namely the maximum number of iterations (*MaxIt*), the maximum number of iterations without improvement (*MaxImpIt*), and the tabu tenure (*TabTen*). All these parameters were tuned via the tuning software IRACE (López-Ibánez, Dubois-Lacoste, Stützle, & Birattari, 2011), using the default settings of the package (one thousand iterations), and the *slpr* instances as the tuning set. As a result, three elite candidate list of parameters were obtained. All of them indicate a *TabTen* of 8. Concerning *MaxIt* and *MaxImpIt*, different values were obtained. Nevertheless, they all point to a similar range of values: more than 800 but less than 900 iterations, and more than 90 but less than 100 iterations without improvement. Based on these results, we opted for setting $TabTen = 8$, $MaxIt = 900$ and $MaxImpIt = 100$.

The HS algorithm, on the other hand, only requires the *MaxIt* parameter. This parameter is a stopping criterion, and thus to verify if the chosen value influences the performance of the algorithm it is enough to check in which iteration the best solution was found. Some preliminary experiences made over both sets of instances, *slpr* and *seix*, indicate that 900 iterations is more than enough, since the algorithm have never run more than 400 iterations.

5.2. Effect of the β parameters

The main objective of this research is to present algorithms that are capable of finding good quality solutions, which means solutions that minimize simultaneously three criteria, namely the number of connected components, the workload time imbalance and the total routing time. Nonetheless, the suggested function to evaluate solutions, $Eval(S)$, by means of the 0/1 β parameters (β_{TT}, β_{CC}, and β_{WIB}), allows to opt for focusing the objective on only one or two of the aforementioned criteria, as remarked in Section 4.1.

In order to analyze the effect of the β parameters on the algorithms' performance, some computational experiments were made with the *slpr* instances. The results are reported in Table 4. The first two columns respectively identify the algorithms (HC and TS) and the statistics (Min, Avg, and Max respectively for the minimum, the average, and the maximum value). Then, there are three groups

Table 3
Characteristics of the *seix* instances.

	Number of					Time	Vehicle
Instance	Nodes	Links	Required edges	Required arcs	Sectors	limit	capacity
seix1	106	214	84	52	2	18,000	3000
seix2	106	214	84	52	2	18,000	5000
seix3	106	214	84	52	2	18,000	10,000
seix4	106	214	84	52	3	14,400	2000
seix5	106	214	84	52	3	14,400	3000
seix6	106	214	84	52	3	14,400	5000
seix7	106	214	84	52	4	10,800	1000
seix8	106	214	84	52	4	10,800	2000
seix9	106	214	84	52	4	10,800	3000

Table 4
Algorithms' performance for different β parameter settings – *slpr* instances.

		LB_{gap} (percent)				Connected components				Workload imbalance			
		$\frac{TT(H)-LB}{LB}$				CC				WIB(seconds)			
		(100)	(010)	(001)	(111)	(100)	(010)	(001)	(111)	(100)	(010)	(001)	(111)
	Min	**0.24**	1.53	2.48	1.00	**2**	**2**	**2**	**2**	14	39	**0**	**1**
HC	Avg	**2.96**	4.82	5.73	4.26	10	**6**	12	**6**	6997	3354	**11**	325
	Max	**5.79**	9.21	12.64	8.05	35	**14**	39	**14**	14,203	10,297	**40**	2197
	Min	**0.00**	1.53	1.83	0.30	**2**	**2**	**2**	**2**	425	39	**0**	**2**
TS	Avg	**2.63**	4.82	5.97	3.66	12	**6**	13	**6**	8917	3354	**3**	**6**
	Max	**5.32**	9.21	13.52	7.10	36	**14**	36	**14**	16,996	10,297	**13**	**14**

Table 5
Computational results – *slpr* instances.

	Total routing time																
	TT(seconds)			LB_{gap}(percent)			UB_{gap}(percent)		$CC-K$			WIB(seconds)			CPU time (seconds)		
				$\frac{TT(H)-LB}{LB}$			$\frac{TT(S0)-TT(H)}{TT(S0)}$										
Instance	$S0$	HC	TS	$S0$	HC	TS	HC	TS	$S0$	HC	TS	$S0$	HC	TS	$S0$	HC	TS
slpr-a-01	13,795	13,830	13,761	2.31	2.57	2.05	−0.25	0.25	0	0	0	39	8	3	0.03	0.02	3.20
slpr-a-02	29,423	29,303	28,971	4.89	4.46	3.28	0.41	1.54	0	0	0	413	1	11	0.09	0.05	42.72
slpr-a-03	78,789	78,416	78,467	3.51	3.02	3.09	0.47	0.41	3	1	1	360	22	9	0.22	1.07	13.27
slpr-a-04	134,790	133,733	132,515	6.18	5.35	4.39	0.78	1.69	0	0	0	6441	22	63	0.26	13.63	442.14
slpr-a-05	217,571	215,808	215,712	7.32	6.45	6.40	0.81	0.85	0	0	0	8402	2197	569	0.43	41.07	138.22
slpr-b-01	15,327	15,132	14,880	3.32	2.00	0.30	1.27	2.92	0	0	0	1165	20	0	0.02	0.05	4.83
slpr-b-02	29,752	29,467	29,295	3.83	2.84	2.24	0.96	1.54	0	0	0	216	3	1	0.04	0.17	13.74
slpr-b-03	82,790	82,420	81,079	6.33	5.86	4.14	0.45	2.07	0	0	0	2917	39	91	0.14	0.65	38.90
slpr-b-04	137,775	137,155	135,761	8.54	8.05	6.96	0.45	1.46	0	0	0	4163	40	41	0.25	4.16	54.73
slpr-b-05	229,114	226,028	224,696	9.21	7.74	7.10	1.35	1.93	0	0	0	10,297	2168	7185	0.43	34.86	201.05
slpr-c-01	18,925	18,826	18,812	1.53	1.00	0.93	0.52	0.60	0	0	0	343	10	12	0.02	0.10	12.00
slpr-c-02	37,382	37,382	37,292	2.87	2.87	2.62	0.00	0.24	0	0	0	376	14	0	0.05	0.12	26.77
slpr-c-03	116,075	116,167	116,169	4.46	4.54	4.55	−0.08	−0.08	0	0	0	4310	145	32	0.17	4.23	113.03
slpr-c-04	174,432	173,460	173,264	3.56	2.98	2.86	0.56	0.67	0	0	0	4588	116	68	0.26	27.77	89.61
slpr-c-05	270,028	268,551	268,001	4.71	4.13	3.92	0.55	0.75	0	0	0	4386	73	112	0.42	83.89	234.41

of four columns, with each group representing one criterion and each column a scenario for the β parameters. The columns are labeled as a triple, representing the values assigned to β_{TT}, β_{CC}, and β_{WIB} respectively, which, in turn, indicate whether the corresponding criterion is being considered (1), or not (0).

The results listed in Table 4 show that the best LB_{gap} gaps and the best WIB values are obtained when the corresponding β parameter is set to one, and the remaining ones are set to zero, no matter which algorithm is being considered. Regarding the number of connected components, the best results are provided by the scenarios in which $\beta_{CC} = 1$. Thus, it seems crucial to take the number of connected components into account, no matter which values are being considered for the other β parameters. To sum up, scenario (111) seems to be a good commitment, as by considering all the criteria simultaneously, the obtained solutions do not appear to deteriorate the quality of each of the individual criterion too much. In the next two sections, we provide a more detailed analysis of the performance of the algorithms on each one of the instances sets, and using all the criteria, i.e. the set (111) for the β parameters.

5.3. *Analysis of the* slpr *instances*

Table 5 reports the results, per instance, obtained with the initial constructive method ($S0$) and with both local search algorithms (HC and TS). The first column identifies the instance, and the remaining seventeen columns are divided into six groups. The groups, in turn, are subdivided into columns, headed as $S0$, HC, and TS, to identify the heuristic they refer to. The first three groups provide results about total routing time. The first group, named as TT(seconds), lists the total routing time, which is expressed in seconds, and the second and third groups list the LB_{gap}, and the UB_{gap}. It is worth noting that positive UB_{gap} percentage gaps highlight the fact that the local search algorithms were able to decrease the total routing time of solution $S0$. The fourth group, $CC-K$, compares the number of connected components (CC) with the number of sectors (K), while the fifth group lists workload imbalance WIB(seconds), expressed in seconds. The last group reports CPU time, in seconds. We remark that the time for both HC and TS also includes the time for determining the initial solution $S0$.

Analyzing Table 5, it can be stated that:

- Despite being computed via a MCARP lower bound, which can be a weak lower bound for the problem at hand, the LB_{gap} percentage gaps are quite small, which in turn point to good feasible solutions. Only in 2 out of 15 instances (a-01, c-03) the UB_{gap} is negative, which means that both LS heuristics are able to decrease the initial routing times. Even for those two aforementioned instances, the negative UB_{gap} is quite small (−0.25% and −0.08%, respectively). Moreover, comparing the UB_{gap} provided by HC and TS, one can observe that TS outperforms HC, exception made for the a-03 instance;
- In all but one instance (a-03), the number of connected components matches the number of sectors, no matter which algorithm is applied. For the a-03 instance, LS methods outperform the initial one. It is worth remarking that this instance is the only one that has a demand subgraph with two connected components, one of which with only a few number of tasks;
- With regards to the WIB criterion, the results show that both HC and TS are able to improve $S0$. Moreover, LS algorithms perform similarly, being TS better for 8 instances, and HC for the remaining 7;

Table 6
Comparing TS with previous approaches – *slpr* instances.

	TT (seconds)			*CC*			*WIB* (seconds)		
Instance	TS–CTH	TS–STH	TS–BIH	TS–CTH	TS–STH	TS–BIH	TS–CTH	TS–STH	TS–BIH
slpr-a-01	80	0	16	−1	0	0	−30	−176	−142
slpr-a-02	−302	−278	−772	−2	−2	−2	−130	−274	−86
slpr-a-03	−157	−44	−1611	−5	−4	−4	−456	−702	−387
slpr-a-04	−1795	−3120	−4816	−20	−18	−18	−616	−821	−455
slpr-a-05	−4749	−6686	−9004	−35	−26	−26	−373	−184	61
slpr-b-01	−307	−440	−327	−1	−2	−2	−153	−194	−107
slpr-b-02	−341	−449	−946	0	0	0	−791	−31	−140
slpr-b-03	−1466	−2370	−3018	−10	−5	−5	−623	−88	−144
slpr-b-04	−93	−3473	−4980	−22	−15	−15	−890	−820	−371
slpr-b-05	−5634	−13,996	−17,351	−65	−83	−83	4207	6344	6681
slpr-c-01	−67	−213	−202	0	0	0	−227	−201	−96
slpr-c-02	13	161	−166	0	0	0	−265	−213	−262
slpr-c-03	−504	−391	−1852	−6	−10	−10	−613	−318	−464
slpr-c-04	−2543	−2679	−4961	−10	−21	−21	−645	−945	−662
slpr-c-05	−4306	−4529	−8826	−20	−31	−31	−789	−545	−206

- The CPU time tends to increase with the dimension of the problem, no matter which method is being considered. The constructive heuristic runs in a few seconds, whereas the local search based algorithms are more time consuming, as expected. The differences between HC and TS CPU times are certainly due to the number of iterations required until one of the stopping criteria is reached, and also to the fact that each iteration obliges re-computing the number of connected components. Nonetheless, the CPU time is always less than 8 minutes, even for TS, that is the most time consuming method.

To conclude, the constructive method provides good feasible solutions with respect to total routing time (*TT*) and to the number of connected components (*CC*), but with high workload time imbalances (*WIB*). The local search algorithms are able not only to significantly decrease the workload time imbalance, but also to frequently improve the total routing time, with small increases in just a few instances. The number of connected components is also maintained or even improved. We also remark that the decrease in the total routing time is not accompanied by an increase in the workload time imbalance as it could be expected, since these objectives can have an interfering effect on each other.

Leaving aside the CPU running time, which is meaningless for the problem being analyzed, TS is the local search algorithm that almost always provides the best results.

To assess the efficiency of TS over the alternative approaches already reported in the literature, namely CTH, STH and BIH (see Mourão et al., 2009), Table 6 lists the difference between the results provided by TS and by the alternative approaches for each one of the criteria being evaluated. Thus, negative values mean that TS performs better than the alternative approach. The results are divided into three groups, one for each criterion. As the number of connected components is not reported in Mourão et al. (2009), we had to compute them by re-running the three aforementioned approaches: CTH, STH and BIH.

In Table 6 it can be seen that TS yields not only the smallest workload imbalances in all but 2 instances (a-05 and b-05), but also the lowest number of connected components in all the cases. The decrease in the number of connected components is quite significant for the large-sized instances. Furthermore, these improvements are almost always accompanied by a decrease in the total routing time. The only exceptions are instances a-01 and c-02, for which worse routing times are deeply compensated by an improvement on the *WIB* criterion, indicating solutions with better characteristics from a practical point of view. Although slower than the aforementioned constructive heuristics, TS CPU time increases no more than 8 minutes, which is meaningless.

From a more managerial perspective, we can conclude that TS provides better quality solutions than the previous approaches: sectors are better balanced, which reveals an improvement on fairness amongst crews; the number of connected components is smaller pointing to sectors that are geographically located in more delimited areas, which is also a benefit for waste collection management.

5.4. *Analysis of the* seix *instances*

Table 7 summarizes the results, per *seix* instance, yielded by the initial constructive method ($S0$), HC, and TS.

As can be observed, $S0$ outperforms both local search methods only for instance *seix*7 (with a UB_{gap} of -2.59 and -2.66, respectively for HC and TS). Moreover, when compared to HC or TS, $S0$ provides the best total routing time only for 2 out of 9 instances (*seix*5 and *seix*7 for HC; and *seix*7 and *seix*8 for TS). The differences between HC and TS are meaningless: TS is usually better than HC (in 6 out of 9 instances), being the largest difference 189 seconds; and it is worse in 3 cases, with the largest difference equal to 78 seconds. All the methods reach the minimum number of connected components, which is the number of sectors. Both local search methods are able to improve the workload time imbalance of the initial solution ($S0$). These differences are particularly significant for the instances with 4 sectors ($seix7 - seix9$), in which the *WIB* decreases from about one hour to just a few minutes (less than 2 minutes with TS). Taking all the three criteria into account (*TT*, *CC* and *WIB*) TS performs slightly better than HC. With regards to CPU times, the initial method is the quickest, whereas TS is the slowest one, although running in just a few seconds.

To conclude this analysis, and as before, Table 8 compares TS with the previous alternative approaches, namely STH, BIH and CTH.

As has already been stated for the *slpr* instances, TS usually gives better *TT* values than the previous approaches. Moreover, even when *TT* increases the imbalance and the number of connected components decrease. Thus, once again, it can be stated that TS outperforms the previous approaches, as it produces solutions that are better fitted for real life applications, with even small routing time values.

6. Conclusions

In this paper we propose a constructive heuristic and two local search heuristics for a sectoring-arc routing problem. Using a real life application, namely the collection of urban waste, the proposed methods were designed to provide solutions not only with

Table 7
Computational results – *seix* instances.

	Total routing time																
	TT (seconds)			LB_{gap} (percent)			UB_{gap} (percent)		$CC-K$			WIB (seconds)			CPU time (seconds)		
				$\frac{TT(H)-LB}{LB}$			$\frac{TT(S0)-TT(H)}{TT(S0)}$										
Instance	$S0$	HC	TS	$S0$	HC	TS	HC	TS	$S0$	HC	TS	$S0$	HC	TS	$S0$	HC	TS
seix1	32,423	32,334	32,290	4.94	4.65	4.51	0.27	0.41	0	0	0	1691	6	8	0.02	0.03	7.00
seix2	31,549	31,480	31,435	3.58	3.35	3.20	0.22	0.36	0	0	0	1179	2	1	0.00	0.05	8.55
seix3	30,717	30,586	30,627	2.18	1.74	1.88	0.43	0.29	0	0	0	1367	26	21	0.00	0.14	10.12
seix4	33,579	33,510	33,490	5.41	5.19	5.13	0.21	0.27	0	0	0	1460	34	10	0.00	0.17	13.01
seix5	32,818	32,846	32,657	6.22	6.31	5.70	−0.09	0.49	0	0	0	1047	19	1	0.00	0.47	49.82
seix6	31,472	31,466	31,385	3.32	3.30	3.04	0.02	0.28	0	0	0	1239	37	6	0.00	0.36	15.18
seix7	37,513	38,486	38,511	6.87	9.64	9.71	−2.59	−2.66	0	0	0	3775	52	4	0.00	0.34	8.99
seix8	33,809	33,801	33,879	6.13	6.11	6.35	0.02	−0.21	0	0	0	3003	508	28	0.00	0.47	5.54
seix9	33,003	32,867	32,837	6.82	6.38	6.28	0.41	0.50	0	0	0	3510	1054	87	0.00	0.3	8.91

Table 8
Comparing TS with previous approaches – *seix* instances

	TT (seconds)			CC			WIB (seconds)		
Instance	TS–CTH	TS–STH	TS–BIH	TS–CTH	TS–STH	TS–BIH	TS–CTH	TS–STH	TS–BIH
seix1	311	690	124	−2	−4	−4	−59	−198	−137
seix2	−154	−135	−433	−2	−4	−4	−278	−175	−96
seix3	−110	−111	−222	−2	−4	−4	−134	9	−375
seix4	314	123	216	−4	−2	−2	−444	−702	−508
seix5	−184	−187	−615	−3	−2	−2	−273	−307	−507
seix6	−39	−126	70	−3	−4	−4	−152	−253	−101
seix7	1284	1472	1772	−2	−2	−2	−752	−947	−137
seix8	−226	−152	−559	−3	−3	−3	−308	−370	−207
seix9	781	702	715	−4	−4	−4	−275	−153	−112

small total routing times but also that incorporate some required practical features, such as sectors that are both balanced and connected.

Each of the proposed methods was evaluated on two sets of instances: benchmark instances that were randomly generated according to real life applications, and instances derived from real data. The constructive heuristic proved to be very fast and generally able to provide initial solutions with the optimal number of connected components. Nevertheless, they usually present high imbalance values, mainly for large-sized instances.

The local search heuristics, one being a hill climbing approach (HC) and the other a tabu search approach (TS), were devised to simultaneously handle the three optimization criteria. The computational results highlight the importance of considering all the criteria for evaluating solutions during the search process.

From a more managerial perspective, TS yields significant practical improvements. Sectors are better balanced and have a smaller number of connected components. Consequently, they are geographically located in more delimited areas, which thus simplifies waste collection management.

The design of sectors plays a crucial role when devising solutions for the problem being handled. Therefore, one future research direction is to develop more tailored heuristics or metaheuristics that iteratively alternate between the design of sectors and the design of vehicle trips.

Acknowledgment

This work is supported by National Funding from FCT - Fundação para a Ciência e a Tecnologia, under the projects: UID/MAT/04561/2013 and PTDC/MAT-NAN/2196/2014.

References

Assis, L. S., Franca, P. M., & Usberti, F. L. (2014). A redistricting problem applied to meter reading in power distribution networks. *Computers and Operations Research, 41*, 65–75. doi:10.1016/j.cor.2013.08.002.

Bação, F., Lobo, V., & Painho, M. (2005). Applying genetic algorithms to zone design. *Soft Computing, 9*(5), 341–348. doi:10.1007/s00500-004-0413-4.

Belenguer, J. M., Benavent, E., Lacomme, P., & Prins, C. (2006). Lower and upper bounds for the mixed capacitated arc routing problem. *Computers and Operations Research, 33*(12), 3363–3383. doi:10.1016/j.cor.2005.02.009.

Bozkaya, B., Erkut, E., & Laporte, G. (2003). A tabu search heuristic and adaptive memory procedure for political districting. *European Journal of Operational Research, 144*(1), 12–26. doi:10.1016/S0377-2217(01)00380-0.

Cattrysse, D., Van Oudheusden, D., & Lotan, T. (1997). The problem of efficient districting. *OR Insight, 10*(4), 9–13. doi:10.1057/ori.1997.17.

Constantino, M., Gouveia, L., Mourão, M. C., & Nunes, A. C. (2015). The mixed capacitated arc routing problem with non-overlapping routes. *European Journal of Operational Research, 244*(2), 445–456. doi:10.1016/j.ejor.2015.01.042.

Corberán, A., & Laporte, G. (Eds.) (2014). Arc routing: Problems, methods, and applications. MOS-SIAM series on optimization. Philadelphia: Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611973679.

Corberán, A., & Prins, C. (2010). Recent results on arc routing problems: An annotated bibliography. *Networks, 56*(1), 50–69. doi:10.1002/net.20347.

Dror, M. (Ed.). (2000). *Arc routing: Theory, solutions and applications*. Dordrecht: Kluwer Academic Publishers. doi:10.1080/07408170208936917.

Ghiani, G., Laganà, D., Manni, E., Musmanno, R., & Vigo, D. (2014). Operations research in solid waste management: A survey of strategic and technical issues. *Computers and Operations Research, 44*, 22–32. doi:10.1016/j.cor.2013.10.006.

Ghiani, G., & Laporte, G. (2001). Location-arc routing problems. *OPSEARCH, 38*(2), 151–159.

Glover, F., & Laguna, M. (1997). *Tabu search* (1st). Netherlands: Kluwer Academic Publishers: Dordrecht.

Golden, B., Raghavan, S., & Wasil, E. (Eds.) (2008). The vehicle routing problem: Latest advances and new challenges. Operations research/computer science interfaces series. Springer.

Golden, B. L., & Wong, R. T. (1981). Capacitated arc routing problems. *Networks, 11*(3), 305–315. doi:10.1002/net.3230110308.

Gonzalez-Ramírez, R. G., Smith, N. R., Askin, R. G., Miranda, P. A., & Sánchez, J. M. (2011). A hybrid metaheuristic approach to optimize the districting design of a parcel company. *Journal of Applied Research and Technology, 9*(1), 19–35.

Gouveia, L., Mourão, M. C., & Pinto, L. S. (2010). Lower bounds for the mixed capacitated arc routing problem. *Computers and Operations Research, 37*(4), 692–699. doi:10.1016/j.cor.2009.06.018.

Hanafi, S., Freville, A., & Vaca, P. (1999). Municipal solid waste collection: An effective data structure for solving the sectorization problem with local search methods. *INFOR, 37*(3), 236–254.

Haugland, D., Ho, S. C., & Laporte, G. (2007). Designing delivery districts for the vehicle routing problem with stochastic demands. *European Journal of Operational Research, 180*(3), 997–1010. doi:10.1016/j.ejor.2005.11.070.

Jarrah, A. I., & Bard, J. F. (2012). Large-scale pickup and delivery work area design. *Computers and Operations Research, 39*(12), 3102–3118. doi:10.1016/j.cor.2012.03.014.

Jozefowiez, N., Semet, F., & Talbi, E.-G. (2009). An evolutionary algorithm for the vehicle routing problem with route balancing. *European Journal of Operational Research, 195*(3), 761–769. doi:10.1016/j.ejor.2007.06.065.

Kim, B. I., Kim, S., & Sahoo, S. (2006). Waste collection vehicle routing problem with time windows. *Computers and Operations Research, 33*(12), 3624–3642. doi:10.1016/j.cor.2005.02.045.

Lacomme, P., Prins, C., & Ramdane-Chérif, W. (2001). A genetic algorithm for the capacitated arc routing problem and its extensions. In E. Boers (Ed.), *Applications of evolutionary computing*. In Lecture notes in computer science*: Vol. 2037* (pp. 473–483). Berlin: Springer. doi:10.1007/3-540-45365-2_49.

Lacomme, P., Prins, C., & Ramdane-Chérif, W. (2004). Competitive memetic algorithms for arc routing problems. *Annals of Operations Research, 131*(1–4), 159–185. doi:10.1023/B:ANOR.0000039517.35989.6d.

Lei, H., Laporte, G., & Guo, B. (2012). Districting for routing with stochastic customers. *EURO Journal on Transportation and Logistics, 1*(1-2), 67–85. doi:10.1007/s13676-012-0005-x.

Lin, H. Y., & Kao, J. J. (2008). Subregion districting analysis for municipal solid waste collection privatization. *Journal of the Air and Waste Management Association, 58*, 104–111. doi:10.3155/1047-3289.58.1.104.

López-Ibánez, M., Dubois-Lacoste, J., Stützle, T., & Birattari, M. (2011). The IRACE package: Iterated racing for automatic algorithm configuration. *Technical Report, 2011-004*. Belgium: IRIDIA, Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle, Université Libre de Bruxelles.

Male, J. W., & Liebman, J. C. (1978). Districting and routing for solid-waste collection. *Journal of the Environmental Engineering Division-ASCE, 104*(1), 1–14.

Mourão, M. C., Nunes, A. C., & Prins, C. (2009). Heuristic methods for the sectoring arc routing problem. *European Journal of Operational Research, 196*(3), 856–868. doi:10.1016/j.ejor.2008.04.025.

Mourgaya, M., & Vanderbeck, F. (2007). Column generation based heuristic for tactical planning in multi-period vehicle routing. *European Journal of Operational Research, 183*(3), 1028–1041. doi:10.1016/j.ejor.2006.02.030.

Muyldermans, L., Cattrysse, D., & Van Oudheusden, D. (2003). District design for arc-routing applications. *Journal of the Operational Research Society, 54*(11), 1209–1221. doi:10.1057/palgrave.jors.2601626.

Oyola, J., & Løkketangen, A. (2014). GRASP-ASP: An algorithm for the CVRP with route balancing. *Journal of Heuristics, 20*(4), 361–382. doi:10.1007/s10732-014-9251-4.

Perrier, N., Langevin, A., & Campbell, J. F. (2006a). A survey of models and algorithms for winter road maintenance. Part I: System design for spreading and plowing. *Computers and Operations Research, 33*(1), 209–238. doi:10.1016/j.cor.2004.07.006.

Perrier, N., Langevin, A., & Campbell, J. F. (2006b). A survey of models and algorithms for winter road maintenance. Part II: System design for snow disposal. *Computers and Operations Research, 33*(1), 239–262. doi:10.1016/j.cor.2004.07.007.

Perrier, N., Langevin, A., & Campbell, J. F. (2008). The sector design and assignment problem for snow disposal operations. *European Journal of Operational Research, 189*(2), 508–525. doi:10.1016/j.ejor.2007.05.022.

Ramos, T. R. P., & Oliveira, R. C. (2011). Delimitation of service areas in reverse logistics networks with multiple depots. *Journal of the Operational Research Society, 62*(7), 1198–1210. doi:10.1057/jors.2010.83.

Ríos-Mercado, R. Z., & Fernández, E. (2009). A reactive GRASP for a commercial territory design problem with multiple balancing requirements. *Computers & Operations Research, 36*(3), 755–776. doi:10.1016/j.cor.2007.10.024.

Salazar-Aguilar, M. A., Ríos-Mercado, R. Z., González-Velarde, J. L., & Molina, J. (2012). Multiobjective scatter search for a commercial territory design problem. *Annals of Operations Research, 199*(1), 343–360. doi:10.1007/s10479-011-1045-6.

Salhi, S., & Rand, G. K. (1989). The effect of ignoring routes when locating depots. *European Journal of Operational Research, 39*(2), 150–156. doi:10.1016/0377-2217(89)90188-4.

Simchi-Levi, D. (1992). Hierarchical planning for probabilistic distribution-systems in Euclidean spaces. *Management Science, 38*(2), 198–211. doi:10.1287/mnsc.38.2.198.

Teixeira, J., Antunes, A. P., & Sousa, J. P. (2004). Recyclable waste collection planning – A case study. *European Journal of Operational Research, 158*(3), 543–554. doi:10.1016/S0377-2217(03)00379-5.

Toth, P., & Vigo, D. (Eds.) (2014). Vehicle Routing: Problems, Methods, and Applications. MOS-SIAM series on optimization (2nd). Philadelphia: Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611973594.

Wøhlk, S. (2008). A decade of capacitated arc routing. In B. Golden, S. Raghavan, & E. Wasil (Eds.), *The vehicle routing problem: Latest advances and new challenges*. In Operations research/computer science interfaces (pp. 29–48). Berlin: Springer. doi:10.1007/978-0-387-77778-8_2.