**Submitted to *Transportation Science***
**manuscript TS-2015-0155**

# A special VRP arising in the optimization of waste disposal: a real case

Roberto Aringhieri
Dipartimento di Informatica, Università degli Studi di Torino, Italy roberto.aringhieri@unito.it,

Maurizio Bruglieri
Dipartimento di Design, Politecnico di Milano, Milano, Italy, maurizio.bruglieri@polimi.it,

Federico Malucelli
DEIB, Politecnico di Milano, Milano, Italy, federico.malucelli@polimi.it,

Maddalena Nonato
EnDiF, Università degli Studi di Ferrara, Ferrara, Italy, maddalena.nonato@unife.it,

We address a particular pick-up and delivery vehicle routing problem arising in the collection and disposal of bulky recyclable waste. Containers of different types, used to collect different waste materials, once full, must be picked-up by dedicated vehicles to be emptied at suitable disposal plants (according to their contents) and replaced by empty containers alike. All requests must be served and routes are subject to a maximum duration constraint. Minimizing the number of vehicles is the main objective, while minimizing the total route duration is a secondary objective. We formalize the problem as a special VRP on a bipartite graph, we analyze its structure and we compare it to similar problems emphasizing for the first time the impact of limited spare containers available. Moreover we propose a neighborhood based metaheuristic which alternatively switches from one objective to the other along the search path, and periodically destroys and rebuilds parts of the solution. The main algorithm components are experimentally evaluated on real and realistic instances, the largest of which fail to be solved by a MILP solver. We are increasingly competitive with the solver as the instance size increases, especially regarding fleet size. In addition the algorithm is applied to some benchmark instances for the Rollon-Rolloff VRP.

*Key words*: waste management; Rollon-Rolloff; distance constrained VRP; hierarchical neighborhood search; spare containers

## 1. Introduction

In recent years, bulky waste recycling started to be extensively promoted on a large scale by local authorities. While the collection of small recyclable household waste is carried out according to a door-to-door collection pattern or by way of dedicated bins which get periodically emptied on site, the bulky waste collection service is deployed according to a different logistic. Indeed, items have to be conveyed by the owners to dedicated conveyance centers rather than being periodically collected on site by company vehicles. Both in Europe and in North America, several conveyance centers have been established, so called recycling and disposal stations or multi-purpose recycling sites (RDSs). RDSs are usually located in the proximity of industrial settings, in the outskirts of large urban areas, or incorporated within existing solid waste transfer stations. Residents bring their bulky waste material to the closest RDS and dispose it into dedicated containers. Several types of materials can be disposed at an RDS. The list includes batteries, garden trim-yard, cardboard, furniture, electric appliances, metals, wood, and rubble, plus many others, except for industrial and construction debris. Each RDS hosts several containers, each one devoted to a single type of waste material. Containers are classified depending on access side (left, right or rear access skips) and compacting equipment; however, the container type dedicated to a given material may vary from RDS to RDS, as it may depend on the layout of the site.

Once a container is full, its content must be carried to a Material Reclaimer Facility (MRF) equipped to process that particular material. This is usually done by loading the container on a vehicle at the RDS and transporting it to an MRF where it is emptied. Due to the size, vehicles can transport one container at a time. The full container must be replaced by an empty one of the same type, at the original RDS, possibly by the original container itself once it has been emptied at a MRF. In the following we call this operation a *service request*. These operations take place when RDSs are not open to the public, typically one day a week, which allows a certain degree of freedom regarding the scheduling of picking up the full container and delivering an empty one. The multiplicity of container types, RDSs, and MRFs and potential spare containers make the problem of optimizing the container disposal operations rather complex and call for efficient optimization methods, especially in light of the fact that the quantity of recyclable waste constantly increases (almost doubling every two years in Italy according to Aringhieri et al. (2004a)).

In this paper we consider the problem faced by Gesenu (www.gesenu.it), a waste management company in central Italy, where the concern is both on the minimization of the number of vehicles needed to carry out the service and on the minimization of the total travel time. At the time this study was started, the service used to be manually managed as it involved few RDSs and few vehicles, however the imminent extension of the covered area and of the fleet size suggested the managers to adopt an optimization based solution approach.

## 1.1. Problem Statement

Let us consider: a set of locations, i.e., RDSs, MRFs, and the depot; a set of container types; a set of materials. Every RDS issues a set of service requests. Every request involves the RDS, a container type, and a material type, and it is made of two *elementary requests*: the first one concerns a full container to be carried to a proper MRF to be emptied (afterward referred to as `full`); the second one concerns its replacement by any empty container of the same type, with no synchronization constraints and regardless of the material type previously hosted in the empty container (afterward referred to as `empty`).

The vehicle is stationed at the MRF during emptying operations and must carry away the container right afterward. Several requests can be present at the same RDS. There is no one-to-one correspondence between a container type and a material type, so that the matching can vary from RDS to RDS. Each MRF can process only a given set of materials. The depot hosts vehicles and a limited set of spare containers for each type. Vehicles are identical and carry at most one container at a time, either empty or full. Vehicle routes start and end at the depot. Travel times between each pair of locations are known, and may differ depending on the loading state of the vehicle. All requests must be served and routes are subject to a maximum duration constraint. Minimizing the number of vehicles is the highest priority, while minimizing total duration is a secondary objective. We call this problem the *Ecological 1-Load Container Routing Problem* (E1LCRP).

**Figure 1** **An E1LCRP solution on the physical network for an instance with five requests located at three RDSs and two MRFs, the first handling paper (P) and metals (M), and the second wood (W), glass (G) and paper. Black and white circles represent the elementary requests `full` and `empty`, respectively. Two vehicles are used to dispose the waste: their route is depicted in solid line (first vehicle), dotted line (first tour second vehicle), and dashed line (second tour second vehicle).**

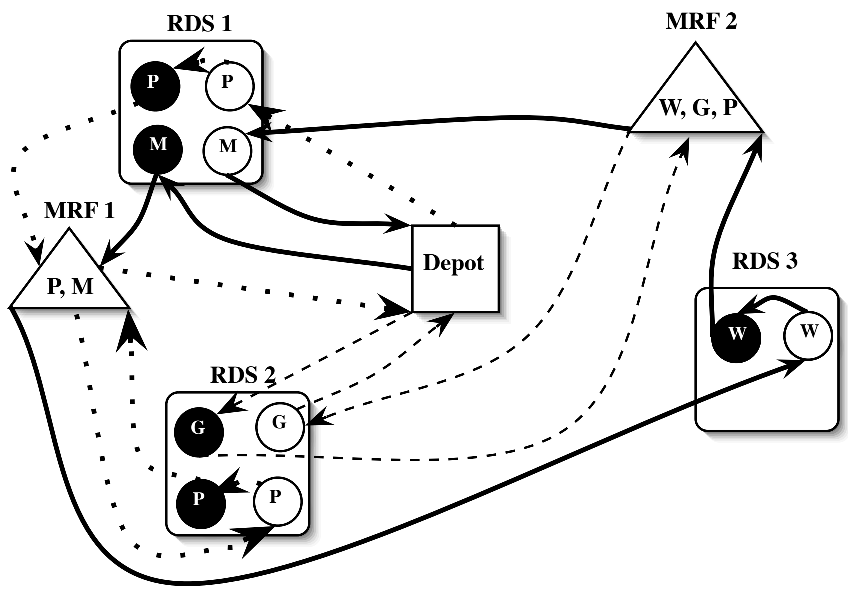


Figure 1 depicts a problem instance and a feasible solution on the physical network. The first vehicle (solid line) leaves the depot with no container, arrives at RDS 1, loads the full metal container

(**M**) and carries it to MRF 1 to empty it; then, the vehicle brings the empty container to RDS 3 and switches it with the one full of wood (**W**); the wood is later disposed of at MRF 2; the empty container is carried back to RDS 1 and placed in the metal slot. Finally, the vehicle returns to the depot. In this case, wood at RDS 3 and metal at RDS 1 are stored into containers of the same kind. The route of the second vehicle is made of two tours. In the first tour (dotted line) the vehicle loads an empty spare container available at the depot which is of the same type as the one full of paper (**P**) located at RDS 1. This container is carried to RDS 1 and swapped with the full one, which is loaded on the vehicle, carried to MRF 1, and emptied. After paper disposal, the empty container is carried to RDS 2 and switched with the one full of paper (**P**) present there; this is carried to MRF 1 where the paper is disposed. Finally, the empty container is returned to the depot. Along this tour, the vehicle travels loaded by containers of the same type. In its second tour (dashed line), the second vehicle leaves the depot unloaded heading to RDS 2 where the full glass container is loaded and carried to MRF 2. The glass is disposed and the vehicle takes the empty container back to its slot at RDS 2 before heading back to the depot.

This example shows that the service can be carried out according to three different patterns: i) bring an empty container to an RDS and swap it with a compatible full one: paper requests (**P**) at RDS 1 and 2 are both served according to this pattern in the first tour of the second vehicle; ii) take a full container to an MRF, empty it there, and bring it back to its original location: the glass request (**G**) at RDS 2 is served according to this pattern in the second tour of the second vehicle; iii) serve independently the two elementary requests `full` and `empty` a service request is made of, as it is the case of the metal request (**M**) at RDS 1 in the route of the first vehicle, where `full` is served at the beginning of the route and `empty` at the end; in general, `full` and `empty` are potentially served by two different vehicles, with no precedence relation nor synchronization between them. Pattern i) is termed a *swap*, pattern ii) is termed a *petal*. Finally, note that a tour can start with a *swap* only provided that spare containers are available at the depot.

### 1.2. Contributions and Paper Organization

The paper is organized as follows. We analyze previous studies on similar problems concerning waste container routing, enlightening the main differences with E1LCRP and summarizing the proposed solution approaches (Section 2). We show also that E1LCRP is a general framework able to formalize other Rollon-Rolloff VRPs. E1LCRP is then formalized and modeled as a duration constrained vehicle routing problem on a bipartite graph in Section 3. The solution approach is described in Section 4 where first the complexity is addressed, a constructive procedure is sketched and its correctness is proved, and the building blocks of the neighborhood search meta-heuristic are illustrated. One method to compute lower bounds for both total route duration and fleet size is described in Section 5, based on a mixed integer linear programming model. An experimental analysis of the impact of each neighborhood search component is reported in Section 6. In the same section, computational results are discussed, obtained on real instances as well as on more challenging and larger instances derived from random graphs. Furthermore, the approach is tested on benchmark instances from the literature with unbounded additional containers to compare with different algorithms developed for similar problems.

This paper builds on a previous work, Aringhieri et al. (2004b), where E1LCRP was originally introduced as a particular case in single-load waste container routing and the issue of limited spare containers was mentioned for the first time. The main contributions of this paper are: the detailed description of a new graph model for the problem; a systematic review of the literature; a neighborhood search meta-heuristic that exploits route decomposition into tours to pursue minimal fleet size and devises moves tailored on the E1LCRP graph structure to minimize tour duration; a constructive procedure able to handle a limited number of spare containers; an experimental analysis of the different components of the algorithm on realistic instances, with a special emphasis on the impact of the number of additional spare containers.

## 2. Literature Review

Several papers in the literature deal with the many different variants of the Vehicle Routing Problem (VRP) arising in waste collection; see Beliën et al. (2013) for an overview on this topic. Despite of the hardness of VRP, relatively simple heuristics seem to provide good quality results in the field of industrial waste collection, when the problem is defined on special graphs modeling the routing of containers which are carried on tractors as a single load; these problems are often referred to as *Rollon-Rolloff* vehicle routing problems. In this paper we discuss another case that supports this claim, and analyze which problem features may explain such behavior.

We start by reviewing the six studies of De Muelemeester et al. (1997), Bodin et al. (2000), Archetti and Speranza (2004), Baldacci et al. (2006), Wy and Kim (2013), Derigs et al. (2013) addressing Rollon-Rolloff problems closely related to E1LCRP and underline which issues have a direct impact on the solution approach. These problems and E1LCRP share the following features:

- a fleet of identical vehicles (homogeneous fleet) is located at a single depot;
- routes start and end at the depot and their duration cannot exceed a given threshold;
- several types of containers may be given and each vehicle can carry containers a) of any type, b) one at a time, either empty or full;
- requests concern: collection of full containers to be removed from a given site in order to be either emptied or relocated somewhere else; deliveries of empty containers at a given site;
- each request concerns a specific container type; when it involves a full container, it also concerns a specific type of waste material;

- waste material type unequivocally determines a subset of MRFs;
- company containers are interchangeable within each type, and can be passed from one client to another, whereas privately owned containers, when present, must be returned to their original site;
- empty spare containers are available;
- there is a one to one correspondence between a route, a driver's shift, and a vehicle.

Actually, Archetti and Speranza (2004) considers an heterogeneous fleet but the following restrictions hold: i) all requests are swaps; ii) there are two types of vehicles and two types of containers, each vehicle type is bound to a container type; iii) the vehicle always leaves the depot loaded by an empty container and performs a sequence of swaps before returning to the depot; iv) the depot is the only place where a container type can be changed. Therefore, when the driver returns to the depot and changes vehicle, it changes container type as well. It follows that the problem can be reformulated as if the fleet were homogeneous so that Archetti and Speranza (2004) shares the above mentioned features with De Muelemeester et al. (1997), Bodin et al. (2000), Baldacci et al. (2006), Wy and Kim (2013), Derigs et al. (2013) and E1LCRP. Nevertheless, the work of Archetti and Speranza (2004) structurally differs from the other studies since it considers a few constraints coming from the specific real case under study, in particular: (i) demand satisfaction and route duration are treated as soft constraints (overtime work and unsatisfied requests are allowed and yield penalties); (ii) time windows are client dependent; (iii) the number of routes (drivers) is bounded from above. These features require ad hoc solution approaches, therefore we drop it from our analysis.

For sake of completeness, beside Archetti and Speranza (2004) we mention a few recent papers tackling real life vehicle routing problems which can be seen as a generalization of the pure Rollon-Rolloff problem, such as Wy et al. (2013), Hauge et al. (2014), le Blanc et al. (2006), Elbek and Wøhlk (2016). Likewise Archetti and Speranza (2004), each one addresses a few application-specific requirements and priorities that call for tailored solution approaches. In particular: Elbek and Wøhlk (2016) tackles a multi-period problem; in le Blanc et al. (2006) and Hauge et al. (2014) the single load capacity constraint is relaxed, as vehicles can carry from two ( le Blanc et al. (2006)) to eight ( Hauge et al. (2014)) containers at a time; Wy et al. (2013) considers several scheduling restrictions: time windows at clients sites, at yards and at the disposal plants, drivers lunch breaks, and early service preference for special trips that complete a service that was started the day before. Finally, other related studies such as Imai et al. (2007), Jula et al. (2005) concern full container loads with time constraints: in those cases, though, the container is never emptied so it does not circulate on the network except for its shipping from origin to destination. Note that Wy et al. (2013) is the only study to mention a limited number of containers that are stored at each individual yard but apparently there is a sufficient number of containers, as a whole, to satisfy all the requests. The authors tackle this issue algorithmically, during the solution construction phase, whereas in this paper we formalize this issue directly into the network model that supports the solution approach.

From here on, we disregard Rollon-Rolloff generalizations and concentrate on comparing E1LCRP with De Muelemeester et al. (1997), Bodin et al. (2000), Baldacci et al. (2006), Wy and Kim (2013), Derigs et al. (2013). In these problems, overtime is forbidden, time windows at requests are identical since those are induced by maximum route duration, the number of routes is unbounded since vehicles and drivers are such, therefore no request is ever rejected. In particular, Bodin et al. (2000), Wy and Kim (2013), Derigs et al. (2013) all tackle the same problem first introduced by Bodin et al. (2000) and only differ regarding the solution approach.

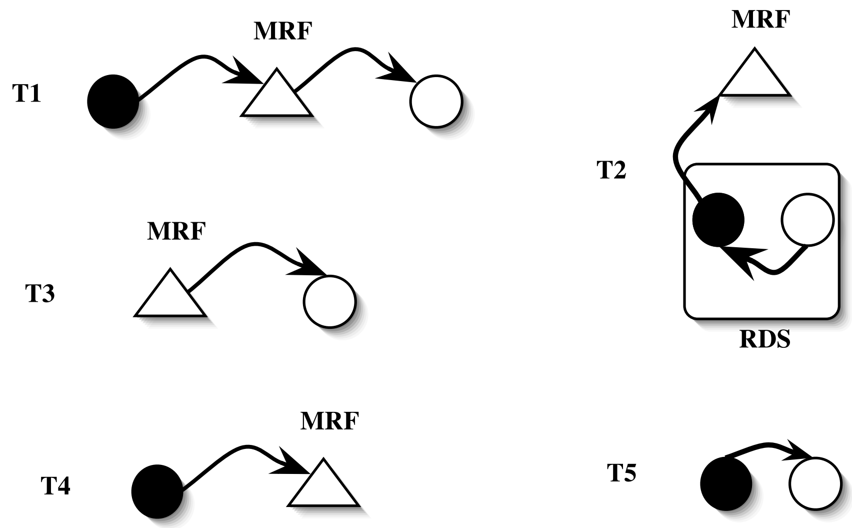
Hereafter, we focus on the three issues that, in our opinion, most characterize this class of problems, i.e., request types, spare containers, and objective function.

**Composite requests**

Given the two elementary requests introduced earlier, i.e., full container disposal (`full`) and empty container delivery (`empty`), additional request types can be built by sequencing `full` and `empty`; Bodin et al. (2000) introduce the following four request types and the associated *trips*: trip T1 corresponds to serving a composite request of the kind `full-empty`; trip T2 corresponds to `empty-full` in the same RDS, i.e., a swap; trip T3 corresponds to the elementary request `empty`; trip T4 corresponds to the elementary request `full`. When the two RDSs involved in the two elementary requests part of a T1 trip coincide, we get a petal pattern. In all but De Muelemeester et al. (1997) most T1s are petal patterns.

An additional trip type T5 is introduced by Baldacci et al. (2006) which consists in the relocation of a partially filled container from a site to another one without detouring to the dumping site: T5 can be modeled as a composite request `full-empty` concerning a new type of container. The five trips are depicted in Figure 2 while petal patterns and swap patterns are depicted in Figure 3. Note that, if the number of `empty` and `full` requests for each container type is not balanced, then there must be at least one container repository where spare containers are available or can be stored after usage. Regarding trip types, it can be observed that if all requests were T1 trips, then the problem would reduce to a distance constrained VRP. If all requests were swaps and unlimited spare containers were available at the MRFs, the problem would reduce to a Bin Packing Problem (BPP). Furthermore, we argue that any trip type proposed so far can be reformulated in terms of a composite request made of elementary `empty` and `full` requests. In our opinion, even the two additional trips T6 and T7 introduced in Wy et al. (2013), namely *BTY* and *FFD*, can be cast

**Figure 2** **T1, T2, T3, T4 and T5 in the scenario with no spare containers. Black and white circles represent the request of full container disposal and the request of delivering an empty container, respectively.**



in our framework, provided the following adjustments. FFD concerns picking up a full container from the depot and emptying it at a disposal site. In our framework it can be modeled by an ad hoc RDS at zero distance form the depot where a `full` request is the only request located there. BTY requires to bring an empty container from a client site to a yard; it can be modeled as an `empty` request at each yard and a `full` request at the client site, involving a new container type not considered by any other request. In particular, each such `empty` request models the potential return of an additional container, which in our framework does not need to be necessarily served, as explained in the next section. The vehicle reaches `full` unloaded (as for any other `full` request) and proceeds to the most convenient `empty` request along its route.

## Spare containers

E1LCRP is the only study, among the six discussed, dealing with a limited number of spare containers. This bound acts as a global constraint and ties decisions related to different routes: for example, modifying the service sequence of the requests in a route may require using a spare container which may or may not be available depending on whether it has been used by another route. We believe this feature makes our problem more challenging, as discussed hereafter.

Spare containers increase the degree of freedom in the routing. Without spare containers, sequences of requests are subject to compatibility constraints: for example, prior to serving a given `empty` request, an unloaded vehicle must first serve a `full` request concerning a container of the same type. On the contrary, if spare containers are available, an unloaded vehicle can load one container of the desired type at the repository and proceed to serve the selected `empty` request, whatever the previous served request was. Conversely, a vehicle loaded by an empty container may unload it at a repository and proceed to serve any `full` request. Without repositories of spare containers, a compatible `empty` request should be served first in order to unload the vehicle. In other words, a detour by a container repository changes the status of the vehicle as wanted (from loaded to unloaded and vice versa, or changing the type of the container the vehicle is loaded with), thus allowing any sequence of requests to become feasible.

Three aspects related to spare containers impact on the problem difficulty: the number of container repositories, their locations, the number of spare containers stationed at each site, which can be either finite or unbounded for each container type.

*Spare container repositories' cardinality.* The more repositories are, the higher chances are that the status of a vehicle can be modified by a short detour to a repository, thus increasing the number of feasible routes. Moreover, any set of disjoint sequences of trips, where all sequences start and end at the same repository, can be combined into routes by solving a (modified) BPP regardless of the order in which they will be executed within the route.

*Repositories location.* Concerning repository locations, the previous effect is amplified when repositories coincide with network locations that must be visited along a route. In particular, when repositories are located at the MRFs, the opportunity for a change of status comes after each `full` request at no extra travel time.

*Bounded spare containers.* Spare containers are a shared resource by the vehicles, even if each one can be used at most once. It follows that bounded spare containers yield a global constraint on the solution, so that the feasibility of individual routes is not sufficient for the feasibility of the whole solution. This issue was first raised in Aringhieri et al. (2004b) of which this paper is a follow up. In Wy et al. (2013), where spare containers are stocked in limited quantities at different yards, this feature is handled algorithmically, during the solution construction phase, when selecting the yard at which an empty container will be picked up given the current solution. On the contrary, in our case, the additional containers are part of the network model, as explained in the next Section.

## Objective functions

A solution can be evaluated according to either fleet size, which models capital cost, or routing cost, modeling operational costs; the latter is the most common objective, either expressed in terms of distance, duration, or monetary cost of the routes; this criterion is usually correlated to the maximum duration constraint holding on individual routes. Optimizing capital cost is challenging since the most efficient routing does not imply minimum fleet size. Vice versa, the minimum fleet solution may be operationally expensive, although within each route the routing must be optimized with respect to this criterion in order to fit maximum duration. Often a hierarchical objective function is used, which incorporates both aspects, but the solution is challenging since the two objectives can be conflicting, as shown by Li et al. (1992).

**Table 1 A summary of comparison over the literature.**

| feature | De Muelemeester et al. | Bodin et al. | Archetti and Speranza | Baldacci et al. | E1LCRP |
|---|---|---|---|---|---|
| 1: MRF | M | S | M | MM | MM |
| 2: MRF location | *d*, any | any | any | any | *d*, any |
| 3: spare containers | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\leq$ |
| 4: repositories | S | S | S | M | S |
| 5: repository location | *d* | MRF | *d* | MRF | *d* |
| 6: requests | T3,T4 | T1-T4 | T2 | T1-T5 | T1-T7 |
| 7: balance E F | $\neq$ | $\neq$ | $=$ | $\neq$ | $=$ |
| 8: constraint | distance | time | time | time | time |
| 9: constraint vs OC | $=$ | $=$ | $\neq$ | $\neq$ | $=$ |
| 10: I o.f. | OC | OC | MC | OC+V | V |
| II o.f. | – | V | – | – | OC |
| 11: fleet size | $\infty$ | $\infty$ | $\leq$ | $\infty$ | $\infty$ |

Table 1 provides a concise picture of how each study is positioned with respect to the previous issues. Since Wy and Kim (2013), Derigs et al. (2013) address the same problem as Bodin et al. (2000) we report for Bodin et al. (2000) only. Each row addresses the following issue: 1 reports the number of MRFs: M stands for multiple, S for single. If some kind of waste can be assigned to more than one MRF we use the symbol MM; 2 reports the location of the MRFs: at the depot (*d*) or anywhere else in the region (any); 3 reports the number of spare containers, either unbounded ($\infty$) or bounded ($\leq$); 4 reports the number of container repositories and their location is given in 5; 6 reports which types of trips are handled, and 7 tells whether empty and full requests are balanced for each container type; 8 concerns the criterion used in the route feasibility constraint, i.e., distance or duration (time); 9 highlights whether the above criterion is the same ($=$) or it is different from ($\neq$) the one involved in the operational costs (OC); 10 concerns the objective function: OC stands for operational costs, MC for monetary costs which include OC and penalties, V for the fleet size; a two-levels hierarchical objective function is denoted by I o.f. and II o.f.; finally, 11 tells whether the fleet size is bounded or unbounded.

The information provided in Table 1 suggests the following insights. Waste type determines the (set of) MRFs at which a full container can be emptied. In case of a single MRF (as in the work of Bodin et al. (2000), Wy and Kim (2013), Derigs et al. (2013)), such node has to be visited once for each trip of type T4 and its successor in the route can be any T3 trip without additional travel time. Moreover, when such unique MRF coincides with the container repository, the possibility for the vehicle of changing status comes at no extra cost after T4. Therefore, T4 trips can be the predecessors of any other trip type, while T3 trips may follow any other trip. This allows to consider all the sequences of trips that start and end at the MRF as building blocks of the solution, which can be assembled in any order into routes, regardless of the solution cost.

Finally, we comment on the solution approaches developed in each work, that are designed to address the specific problem features listed above.

In the work of Archetti and Speranza (2004), where only T2 trips are considered and the depot hosts the spare containers, the vehicle is tied to the same container type in between any two successive passages by the depot. In this approach, the main difficulty is the trip selection. Indeed, a big effort is put on setting up a penalty system able to reflect the consequence on solution quality of postponing a request vs. using an extra vehicle or paying for overwork. Computational results are provided on real data with up to 35 requests per day, but no lower bound is provided.

In the problem solved in Bodin et al. (2000), Wy and Kim (2013), Derigs et al. (2013) a single container repository is located at the unique MRF. In Bodin et al. (2000) the authors exploit the predominance of T2 trips which shifts the problem structure towards a BPP. They compare several approaches, and a simple heuristic performs almost as well as a sophisticated exact approach, exploiting this problem structure. Wy and Kim (2013) and Derigs et al. (2013) were published almost at the same time and report results on the same data set as Bodin et al. (2000): both propose several variants of a Large Neighbourhood Search combined with local improvements and Local Search. In both cases results marginally improve Bodin et al. (2000) (more on total duration than on fleet size) with a comparable number of iterations.

Baldacci et al. (2006) involves multiple disposal facilities at which an unlimited number of empty containers are available. They further generalize the problem considering arc cost and arc weight as different and independent functions, so that, given a set of nodes, the cheapest route may not be feasible whilst a more expensive one is. A column generation framework is developed which is also suitable for the Vehicle Routing Problem with Time Windows (TW-VRP). The authors use an additive bounding procedure to obtain a tight lower bound and generate a promising set of routes which are passed to a commercial software as the columns of a set partitioning problem.

De Muelemeester et al. (1997) solves a version with unlimited spare containers minimizing total duration. The depot acts as the only repository and hosts the dump of so called domestic requests, thus providing several possibilities of starting and ending a route at no extra cost. Trips are sequenced with the only aim of minimizing total duration, and the prevalence of domestic customers makes the maximum duration constraint easier to be satisfied. The solution approach exploits a tight lower bound coming from the solution of a cycle cover on the compatibility graph whose nodes are the elementary requests in T3 and T4. It consists in building a heuristic initial feasible solution to be improved by classical VRP moves.

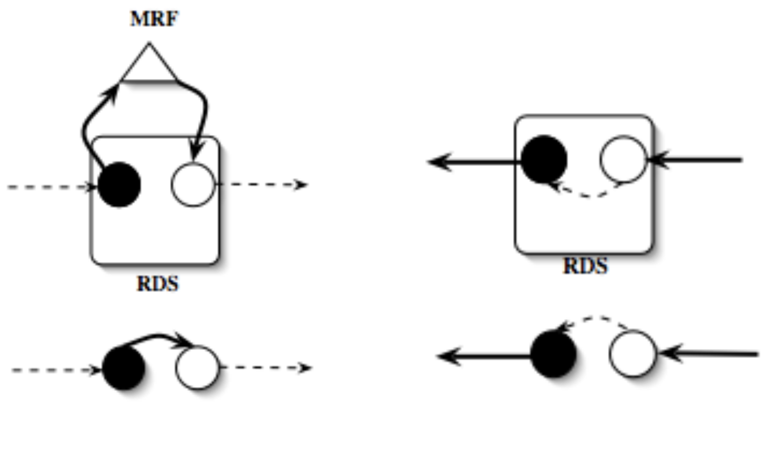
In E1LCRP trips are all T3s and as many T4s, but the approach allows to handle all the kinds of trips described so far. E1LCRP generalizes the problem of De Muelemeester et al. (1997) since it considers a bounded number of spare containers and therefore it cannot be solved by any of the six reported approaches. Any demand pattern can be modeled in terms of elementary requests suitably sequenced, so that any trip type can be modeled in E1LCRP by a partial fixing of the routing decisions. Moreover, a proper number of spare containers for each type must be added to balance the demand or to model the unbounded containers situation (one for each request). However, E1LCRP differs from the problems of Bodin et al. (2000), Wy and Kim (2013), Derigs et al. (2013), Baldacci et al. (2006) concerning the number and location of the container repositories. In fact, in E1LCRP the unique container repository is located at the depot. This feature allows for an intermediate structure in between nodes and routes, that we term a *tour*. Tours can be sequenced disregarding compatibility of the vehicle status, and can be assembled into routes regardless of the order. Note that the total duration component of the objective function depends on the solution tours, while the fleet size component depends on how tours are partitioned into routes; this last step reduces to a Bin Packing Problem. This two-level structure could explain why very difficult problems such as those aforementioned could be handled so well by relatively simple solution approaches.

## 3. A Network Model for the Ecological Single Load Container Routing Problem

Let $R = \{1, \ldots, n\}$ be the set of service requests. Each request $i$ is characterized by a material type $\mu_i$, an RDS $\gamma_i$, and a container type $\beta_i$ among the $K$ available types $\{1, \ldots, K\}$. Requests $i$ and $j$ are said to be *compatible* if they involve the same type of container, i.e., $\beta_i = \beta_j$. E1LCRP can be modeled as a D-VRP on a digraph $G = (N, A)$ representing actions rather than the physical network. Let $N$ include $E \cup F \cup \{d\}$ where $d$ is the depot and sets $E, F$ model service requests as follows: $\forall i \in R$, $f_i \in F$ represents the action of bringing the full container to an eligible MRF (equipped to process material $\mu_i$) and emptying it there, while $e_i \in E$ represents the actions of taking an empty container of type $\beta_i$ to $\gamma_i$ in order to replace the full one, respectively. Clearly, $|E| = |F|$ and both sets $E,\ F$ can be partitioned according to container type into $E = \bigcup_{k \in \{1,\ldots,K\}} E_k$ and $F = \bigcup_{k \in \{1,\ldots,K\}} F_k$, where $E_k = \{e_i \in E : \beta_i = k\}$ and $F_k = \{f_i \in F : \beta_i = k\}$, $\forall k \in \{1, \ldots, K\}$.

The arc set $A$ is made of two classes of arcs: *Loaded arcs* ($A^L$) model vehicles activities when loaded with a container, either full or empty, while *unloaded arcs* ($A^U$) model any vehicle trip without container. A loaded arc $(f_i, e_j)$ models the following sequence of actions: transporting the full container of $i$ from $\gamma_i$ to an MRF, emptying the container, moving it to $\gamma_j$, and unloading it there. Notice that the selected MRF is the eligible one for material $\mu_i$ on the cheapest detour from $\gamma_i$ to $\gamma_j$. Disposal plants are thus embedded into loaded arcs, whose cost includes travel times, dumping and unloading times. Loaded arcs connect $f_i$ to $e_j$ for any pair of compatible requests $(i, j)$. However, we restrict this set to those arcs such that the duration of the operations associated with the path made of the nodes $d, f_i, e_j, d$ does not exceed the maximum duration, for feasibility purpose. Such closed path is termed a *basic tour*. Remind that, if the two requests are co-sited and compatible, i.e., $\gamma_i = \gamma_j$ and $\beta_i = \beta_j$, then arc $(f_i, e_j)$ models the so called *petal pattern*, that is, the action of bringing back a container to its original RDS as soon as it has been emptied (see Figure 3a). Note that the cost of a petal pattern depends on the distance from the RDS to the chosen MRF, which in turn depends on $\mu_i$, the material of $f_i$. Therefore, in case of multiple requests at one RDS concerning the same container type, say two requests $i$ and $j$, then both loaded arcs $(f_i, e_j)$ and $(f_i, e_i)$ model a petal pattern with the same cost, and for each solution involving arc $(f_i, e_j)$ an equivalent solution exists using arc $(f_i, e_i)$.

**Figure 3 A petal pattern (a) and a swap pattern (b) are depicted as they are modeled in the physical network (on top) and in the abstract graph $G$ (at the bottom). Dashed arcs are unloaded while solid arcs are loaded.**



(a) petal pattern (b) swap pattern

Unloaded arcs connect $e_i$ to $f_j$, $\forall i, j \in R$, regardless of container types. Arc $(e_i, f_j)$ models a vehicle that travels unloaded from $\gamma_i$ up to $\gamma_j$ where it picks-up the full container of request $j$. Its cost includes the travel time from $\gamma_i$ to $\gamma_j$ plus container loading time. Note that the arc is defined only given that its duration plus the travel time from $d$ to $\gamma_i$ and from $\gamma_j$ to $d$ passing by the closest MRF, does not exceed the maximum duration. If $\gamma_i = \gamma_j$ and $\beta_i = \beta_j$, an arc is said *co-sited*: co-sited unloaded arcs have a null trip and model the *swap pattern*, i.e., bringing an empty container to a site and swap it with a compatible full one (see Figure 3b).

Note that a vehicle changes its status from loaded to unloaded, and vice versa, according to the last of the actions associated with each arc. Indeed, it is unloaded when outcoming from an $e_i$ node, and loaded when outcoming from an $f_j$ node.

If no spare container is available, a vehicle must leave and return to the depot unloaded. However, the availability of spare containers at the depot allows to deliver an empty container as a first action, or to return to the depot just after a dumping action at a MRF with the vehicle being still loaded by an empty container. Spare containers are modeled by the two sets $\Phi = \cup_k \Phi_k$ and $\Upsilon = \cup_k \Upsilon_k$, where each subset $\Phi_k$ or $\Upsilon_k$, $k \in \{1, \dots, K\}$, refers to containers of type $k$, and $m_k = |\Phi_k| = |\Upsilon_k|$ is the number of additional spare containers of type $k$ available at the depot. Let $R_k$ be the subset of requests involving a container of type $k$, then an instance with unbounded number of spare containers is modeled by a graph with $m_k = |R_k|$, while an instance with bounded spare containers corresponds to a graph such that $\exists k \in \{1, \dots, K\}$ s.t. $m_k < |R_k|$.

Each node $\phi \in \Phi$ models the pick-up of an empty container at the depot, while each node $\upsilon \in \Upsilon$ models its return. Therefore, since vehicles may also leave and return the depot loaded, $A^U$ also includes arcs: $(d, f_i), (e_i, d), (d, \phi), (\upsilon, d) \forall i \in R$, $\forall \phi \in \Phi$, $\forall \upsilon \in \Upsilon$, whereas $A^L$ also includes arcs $(\phi, e_i)$ and $(f_i, \upsilon)$ for each compatible pair, i.e., $\forall i \in R$, $\forall \phi \in \Phi_k$, $\forall \upsilon \in \Upsilon_k$ s.t. $\beta_i = k$.

In summary, $G = (N, A)$ where $N = E \cup F \cup \{d\} \cup \Phi \cup \Upsilon$ and $A = A^U \cup A^L$. Graph $G$ becomes a weighted network by introducing a cost function on $A$ as summarized in Table 2.

Let us extend the notation as follows: $P$ is the set of MRF ($P_{\mu_i}$ are those processing material $\mu_i$); $t_{ij}$ is the travel time from the physical location $i$ to the physical location $j$, where by physical location we intend either an RDS, the depot or a MRF; $\sigma_p$ is the time required for emptying a container at MRF $p$, $\forall p \in P$; $\tau^L$ and $\tau^U$ are the time for loading and unloading, respectively, a container from the vehicle.

The cost of an arc is given by the overall time required to carry out the sequence of actions modeled by the arc. In particular, arc $(f_i, e_j)$ is unequivocally associated with the MRF $p \in P_{\mu_i}$ along the shortest detour from $\beta_i$ to $\beta_j$.

**Table 2 Cost definition of the arcs in graph $G$.**

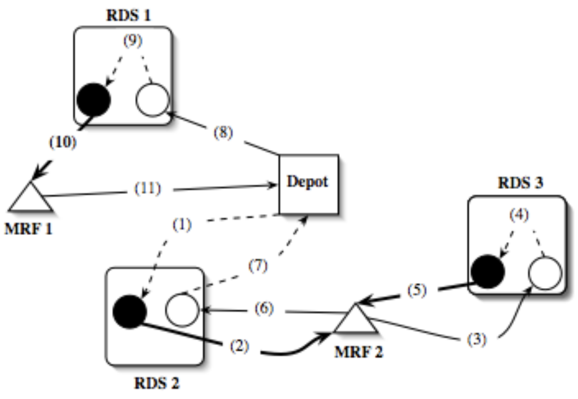
| $(u, v)$ | $c_{uv}$ | nodes |
|---|---|---|
| $(d, f_i)$ | $t_{d\gamma_i} + \tau^L$ | $\forall f_i \in F$ |
| $(d, \phi)$ | $\tau^L$ | $\forall \phi \in \Phi$ |
| $(\phi, e_i)$ | $t_{d\gamma_i} + \tau^U$ | $\forall e_i \in E$, $\forall \phi \in \Phi_k$: $k = \beta_i$ |
| $(e_i, d)$ | $t_{\gamma_i d}$ | $\forall e_i \in E$ |
| $(\upsilon, d)$ | $0$ | $\forall \upsilon \in \Upsilon$ |
| $(f_i, \upsilon)$ | $min_{p \in P_{\mu_i}} \{t_{\gamma_i p} + \sigma_p + t_{pd} + \tau^U\}$ | $\forall f_i \in F$, $\forall \upsilon \in \Upsilon_k$:$k = \beta_i$ |
| $(e_i, f_j)$ | $t_{\gamma_i \gamma_j} + \tau^L$ | $\forall e_i \in E$, $\forall f_j \in F$ |
| $(f_i, e_j)$ | $min_{p \in P_{\mu_i}} \{t_{\gamma_i p} + \sigma_p + t_{p\gamma_j} + \tau^U\}$ | $\forall f_i \in F$, $\forall e_j \in E$:$\beta_i = \beta_j$ |

On the graph $G$ so far defined, any feasible activity of a vehicle can be represented, including switching the type of the container of a loaded vehicle when spare containers are available. Indeed, the pair of arcs $(\upsilon, d)$ and $(d, \phi)$, with $\upsilon \in \Upsilon_k$ and $\phi \in \Phi_{k'}$, models the substitution of a container of type $k$ with a container of type $k'$ at the depot. Notice that the vehicle starts and finishes its daily itinerary at the depot, but it may pass by it several times in order to load, unload or change container type, since spare containers are stored at the depot. Therefore, the vehicle itinerary can be partitioned into smaller portions of activities between successive passages by the depot. This suggests two scheduling levels, tours and routes which are formally defined below.

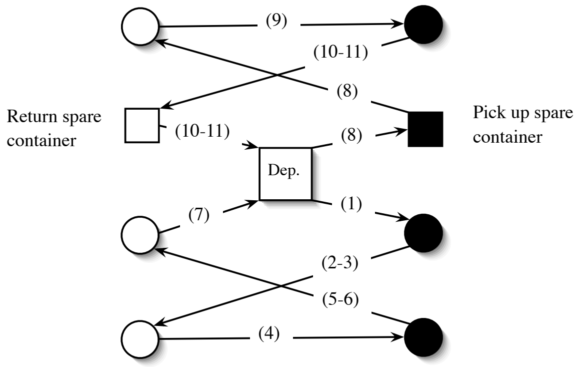
A *tour* is an elementary cycle through $d$ in $G$ whose nodes are an alternating sequence in the sets $E \cup \Upsilon$ and $F \cup \Phi$. A *route* is a collection of tours disjoint on $N \setminus \{d\}$ with total duration less than or equal to a given $D$. A feasible solution is a collection of routes passing exactly once by each $f_i \in F$ and $e_j \in E$ and at most once by each $\upsilon \in \Upsilon$ and $\phi \in \Phi$.

Any solution approach has to handle the special features of the graph: the graph $G$ is bipartite (no triangular inequalities) other than at the depot; the reverse of a loaded arc is unloaded, and vice versa, however, while an unloaded arc is always defined, the loaded one depends on compatibility; the cost matrix is highly asymmetric since loaded arcs hide detour by MRF. Figure 4 compares the representation of the solution for a 3 requests instance on the physical graph and on the bipartite graph. Our approach works on $G$ and exploits its structure, while handling explicitly the two scheduling levels, tours and routes.

**Figure 4** **A solution to E1LCRP with 3 requests is depicted on the physical network (△ represent recycling plants, the vehicle is unloaded on dashed arcs, loaded by an empty container on regular arcs and by a full one on bold arcs), as well as on the bipartite graph (□ model additional containers). Arc ids $(\cdot)$ represent the visit sequence. Arcs $(d, \phi_1)$ and $(\upsilon_1, d)$ are unlabeled since the operation of loading and unloading a spare container at the depot is implicit in the physical graph.**



(a) physical graph



(b) bipartite graph

## 4. Solution Algorithm

In this section, we discuss the computational complexity of the problem (Subsection 4.1) and the basic components of the solution approach, namely: the construction of the starting solution (Subsection 4.2), the neighborhood search, the efficient merging of tours into routes, the use of a variable objective function to handle the two objectives, the intensification and diversification strategies (Subsection 4.3).

## 4.1. NP-Completeness

The version of E1LCRP that minimizes the fleet size is strongly NP-hard by reduction from the D-VRP (Li et al. 1992, Nagarajan and Ravi 2012). Consider an instance of D-VRP given by an undirected graph $\tilde{G} = (\tilde{N} \cup \{0\}, \tilde{A})$, where each node of $\tilde{N}$ is a customer, 0 is the depot, the arc set $\tilde{A}$ represents all the pairs of customers that can be visited by a vehicle, $\tilde{t}_{ij}$ is the shortest distance from node $i$ to node $j$ for each $(i,j) \in \tilde{A}$ and $\tilde{D}$ is the maximum distance that can be traveled by each vehicle. We can build an E1LCRP instance where the depot coincides with 0, no spare containers are available, a service request corresponds to each customer of $N$ and for each request a different container type and a different material is involved, the MRF location and the RDS location coincide and the time required for loading, unloading and emptying a container are negligible (i.e., $\tau^L = \tau^U = \sigma_p = 0$ for each RDS $p$). Moreover concerning the time spent to go from the RDS of a service request $i$ to the RDS of a service request $j$ let be $t_{ij} = \tilde{t}_{ij}$ if $(i,j) \in \tilde{A}$ and $t_{ij} = \infty$, otherwise. Let $\tilde{D}$ be the maximum duration of each vehicle route. It is easy to see that each feasible solution of the E1LCRP on such an instance, if any, corresponds to a feasible solution of the D-VRP on the original instance with the same number of vehicles and overall distance.

The version of the E1LCRP where the number of vehicles is given and the total duration of the vehicle routes is minimized is strongly NP-hard as well, since it contains the TSP as special case (when the number of vehicles is one). Note that the problem of approximating the version of E1LCRP where the number of vehicles must be minimized within a relative factor of 2 is NP-hard. The proof follows immediately from the fact that it holds for the D-VRP (Theorem 2 of Li et al. (1992)) and the reduction from D-VRP to E1LCRP used in the proof of Theorem 1 is *gap-preserving*, since the E1LCRP optimal value for the transformed instance coincides with the D-VRP optimal value for the original instance.

## 4.2. Starting Solution

The tours of the starting solution are computed by a modified version of the well-known Clarke and Wright's algorithm (M-CW) for the VRP. In fact, the parallel version of Clarke and Wright's heuristic proved successful in tackling the problem addressed by De Muelemeester et al. (1997) and it provided the starting solution for a neighborhood based improving algorithm in Bodin et al. (2000). However, the algorithm has to be modified in order to achieve feasibility regarding the limited availability of spare containers.

M-CW works on the auxiliary graph $\widetilde{G} = (\widetilde{N}, \widetilde{A})$, where $\widetilde{N} = E \cup F \cup \{d\}$ and $\widetilde{A} = (E \times F) \cup (\{d\} \times E) \cup (F \times \{d\}) \cup \{(f_i, e_j) \in A^L\} \cup (\{d\} \times F) \cup (E \times \{d\})$, as in De Muelemeester et al. (1997). In $\widetilde{G}$ there are no spare container nodes, as they are implicitly modeled by the arcs in $(\{d\} \times E) \cup (F \times \{d\})$, i.e., $(d, e_i)$ models the pick up of one spare container of type $\beta_i$, while $(f_j, d)$

models the drop off of one of type $\beta_j$. M-CW starts from a set of *elementary tours*, one for each node in $E \cup F$ either of the kind $(d, e_j, d)$ or $(d, f_i, d)$. Since each elementary tour either picks up or returns a spare container, the initial solution is *container-wise* infeasible whenever spare containers are bounded, that is, when the spare containers available are less than the requests for at least one container type. M-CW is composed of two distinct phases.

- **Phase 1:** seeking spare containers feasibility while maintaining duration feasibility. Starting from the elementary tours, the only merging operations allowed are those that save one spare container by inserting a loaded arc $(f_i, e_j) \in A^L$ and deleting arcs $(f_i, d)$ and $(d, e_j)$: it merges two elementary tours into one so called *basic tour*. Among the merging operations allowed, at each iteration the one with highest saving is chosen. Note that if $(f_i, e_j)$ exists in $\widetilde{A}$ then the basic tour yielded by the merging is time-feasible, due to the definition of $A^L$.
- **Phase 2:** Clarke and Wright saving procedure. If container-wise feasibility is reached in phase 1, the second phase of M-CW proceeds as the standard Clarke and Wright algorithm, checking the maximum tour duration when merging. Two types of merging operations are now allowed: either a spare container is saved, as before, by merging $(f_j, d)$ and $(d, e_i)$ into $(f_j, e_i)$, or a detour by the depot is skipped by deleting $(e_i, d)$ and $(d, f_j)$ to insert the unloaded arc $(e_i, f_j)$.

Basically, for each container type $k$, of which $m_k$ is the number of spare containers, phase 1 procedure tries to operate enough mergings so to leave at most $2m_k$ elementary tours involving containers of type $k$. The intuition is that the arcs in the basic tours induce a matching in the bipartite graph $G^k$ with nodes $F_k \cup E_k$, whose edges are those induced by the arcs in $A^L \cap F_k \times E_k$; the procedure looks for a matching leaving at most $2m_k$ free nodes. It pursues this target in a greedy way, guided by highest savings, and when it fails it backtracks according to different strategies depending on which case holds among the followings.

*HP A:* All compatible requests $f_i$ and $e_j$ are linked by an arc, i.e., the basic tour $(d, f_i, e_j, d)$ is not longer than $D$.

*HP B1:* For each request $i \in R$ the length of the basic tour made of the petal pattern does not exceed $D$. Such tours have the form $(d, f_i, e_j, d) : \gamma_i = \gamma_j$.

*HP B2:* All non-cosited compatible requests $f_i$ and $e_j$ are linked by an arc, i.e. each basic tour $(d, f_i, e_j, d) : \gamma_i \neq \gamma_j \wedge \beta_i = \beta_j$ is not longer than $D$).

*HP C:* The duration of any elementary tour is not longer than $D$.

The phase 1 procedure goes as follows.

- Case 1: $(A)$: Savings are computed and ranked as usual. Starting from the ones with highest saving, elementary tours are merged into basic tours. As soon as container-wise feasibility is reached for a container type $k$, no further merging involving type $k$ takes place. Since full and empty requests are balanced for each container type, i.e., $|E_k| = |F_k| \forall k$, container-wise feasibility is always reached.

- Case 2: $(B1 \wedge \neg B2)$: Apply the procedure provided for case $(A)$, merging elementary tours into basic tours. Either container wise feasibility is reached or, for at least a container type $k$, no more mergings are possible but there are more elementary tours left than twice the available containers $m_k$. The backtracking procedure proceeds as follows. Since $(B1)$ holds, some of the computed basic tours are not petal patterns. Recall that highest saving mergings are operated first, and these do not necessarily yield petal patterns. In such a case, some of these tours are broken and their nodes are reconnected to form petal patterns, as follows. Let us focus on container type $k$. Note that, at each RDS, all the elementary request nodes of type $k$ still part of an elementary tour (unmerged nodes) must belong either to $F_k$ or to $E_k$, since otherwise an additional merging yielding a petal pattern could be operated. Moreover, considering all RDSs as a whole, such unmerged nodes are equally divided between $F_k$ and $E_k$, since each merging allowed in the first phase involves one node for each set. Suppose that there are $2n_k$ unmerged nodes and $m_k < n_k$ spare containers available. For $n_k - m_k$ times, do the following: select at random an unmerged node $f_i \in F_k$ and a merged node $e_j \in E_k$ that are cosited. There must be at least one such $e_j$, since elementary requests at each RDS are balanced (each request $i \in R$ yields $f_i \in F$ and $e_i \in E$). Brake the basic tour $e_j$ belongs to, and merge $e_j$ with $f_i$ producing a petal pattern. Let $f_{i'} \in F_k$ be the node preceding $e_j$ in the broken tour: if $f_{i'}$ has a cosited unmerged node in $E_k$, then build a basic tour connecting $f_{i'}$ to that node, according to the petal pattern. Otherwise iterate the procedure on $f_{i'}$. The procedure converges since $E_k$ and $F_k$ are balanced at each RDS, and each merging involves one node for each set. Once the procedure has been applied $n_k - m_k$ times, the remaining nodes in the elementary tours can be served by the available spare containers. Container-wise feasibility is thus achieved since all petal patterns are feasible according to $(B1)$. Note that the sequence of nodes involved into each backtracking operation corresponds to an augmenting alternating path in $G^k$.
- Case 3: $(\neg B1 \wedge B2)$: Consider separately each container type $k$. First, all basic tours made of petal patterns are built by merging, if any. Then, all other mergings are carried out, always according to the highest saving priority rule. The procedure fails to return a feasible solution if the unmerged nodes in $F_k \cup E_k$ outnumber $2m_k$. Note that such nodes must all be cosited and located at the same RDS, say $\gamma$, and are present in equal number in the two sets $F_k$ and $E_k$. The procedure iteratively backtracks, breaking one at a time one of the basic tours produced by the previous mergings involving only RDSs different from $\gamma$. Each such broken tour releases two nodes $f_{i'}, e_{j'} : \gamma_{i'}, \gamma_{j'} \neq \gamma$ each of which is merged with one unmerged node located in $\gamma$, say $e_j$ and $f_i$ respectively, thus yielding two new basic tours both having one node located at RDS $\gamma$. If container-wise feasibility is not reached, then the instance is container wise infeasible. The proof is given below, see 1.

- Case 4: $(\neg B1 \wedge \neg B2 \wedge C)$: Apply the procedure provided for case $(A)$. If it fails to reach container wise feasibility for at least one $k$ and nor $(B1)$ neither $(B2)$ holds, there is no guarantee that a feasible solution exists. For each such $k$ build the bipartite graph $G^k$ and compute a maximum cardinality matching starting from the matching induced by the arcs in $A^L$ selected in the current tours. If, for each type of container $k$, the number of free nodes is no more than twice the number of spare containers of type $k$, then a feasible solution exists; if not, the problem is container-wise infeasible. For a formal proof see 1.

PROPOSITION 1 **(feasibility)**. M-CW *returns a feasible solution, if any.*

*Proof of Proposition 1:* It suffices to prove it for case 3 and 4. Consider case 3:
We have to prove that if container-wise feasibility is not reached, then the instance is not container wise feasible. Suppose that the backtracking procedure has terminated with still more unmerged nodes in $F_k \cup E_k$ at some RDS $\gamma$ than $2m_k$ but no more basic tours to brake (not involving RDS $\gamma$). Consider the bipartite graph $G^k$. It suffices to prove that the matching associated to the current solution is a maximum cardinality matching. Ad absurdum, suppose an augmenting alternating path exists in the graph: it should start and end at unmerged nodes in RDS $\gamma$ (otherwise an additional basic tour could be obtained), and it should contain at least one matched edge none of whose vertices belongs to RDS $\gamma$, thus contradicting the hypothesis that the backtracking procedure had terminated.

Consider case 4: compute a maximum cardinality matching in $G^k$: two cases are possible.
i) *For each $k$, the number of free nodes is no greater than twice the number of spare containers of type $k$.* A feasible solution can be computed by building one basic tour for each edge in the matching, and an elementary tour for each free node. Since requests are balanced, for each $k$ there are as many free nodes in $E_k$ as in $F_k$. Therefore, the associated elementary tours can be feasibly operated by way of the available spare containers.
ii) *For at least one container type, $k$, the number of free nodes in the matching is greater than twice the number of spare containers of type $k$.* Hereafter, we show by contradiction that a feasible solution can not exist.
Denote by $P_k$ be the set of edges in the matching on the bipartite graph $G^k$, involving container type $k$, with $|P_k| = p_k$. Let $r_k$ be the number of requests involving containers of type $k$, and $m_k$ the number of spare containers of this type. Let $free(k)$ be the number of nodes in $E_k \cup F_k$ left unmatched in the maximum cardinality matching solution $P_k$: as mentioned, such nodes are equally divided between $E_k$ and $F_k$. According to ii), $free(k) = 2(r_k - p_k) > 2m_k$ holds. Suppose that a feasible solution, say $Sol$, exists, i.e., $Sol$ is a collection of elementary and basic tours in $\widetilde{G}$ such

that each node in $E_k \cup F_k$ belongs to either one of the former or one of the latter. We show that this leads to a contradiction.
Let $t_k$ be the number of loaded arcs of type $k$ in $Sol$. Recall that, in a feasible solution, from a node $f_i$ there is either an outgoing loaded arc or the arc $(f_i, d)$; likewise, into a node $e_j$ is either entering a loaded arc $(f_i, e_j)$ or the arc $(d, e_j)$. Therefore, the number of arcs $(d, e_j)$ and $(f_i, d)$ related to type $k$ is $2(r_k - t_k)$. Since $Sol$ is feasible, $(r_k - t_k) \leq m_k$. Therefore, loaded arcs in $Sol$ provide a feasible matching with $t_k \geq r_k - m_k$ while in $P_k$ we have $r_k - m_k > p_k$ leading to $t_k > p_k$, thus contradicting the maximality of matching $P_k$. $\square$

Finally, notice that $A \equiv (B1 \wedge B2)$, both $A$ and $B1$ are sufficient conditions for feasibility, while $B2$ is not. In fact, if $A$ then there is a feasible solution made of basic tours, which uses no spare containers; if $B1$, then there is a solution made of basic tours each made of one petal pattern. If $B2$, thought, a counter example can be easily built that does not admit a solution with no spare containers. (see appendix) Finally, each of $A$, $B1$, and $B2$ implies $C$, which in turn is a necessary condition for a feasible solution to exist, due to triangular inequalities on distances on the physical graph. M-CW returns a set of tours, among which some can be elementary and basic. These tours are iteratively improved and merged into routes as described in Section 4.3.

## 4.3. Neighborhood Search

Neighborhood search based metaheuristics have been successfully applied to solve several versions of VRPs as shown by Laporte et al. (2000), Toth and Vigo (2002), due to their flexibility in handling most types of constraints.

Since we deal with both minimum total duration and minimum fleet size our algorithm operates at two different levels of granularity, namely tours and routes as defined in Section 3. Tours are used when minimizing the total duration whilst the algorithm works with routes to minimize the fleet size. This idea will be motivated in the remaining of the Section.

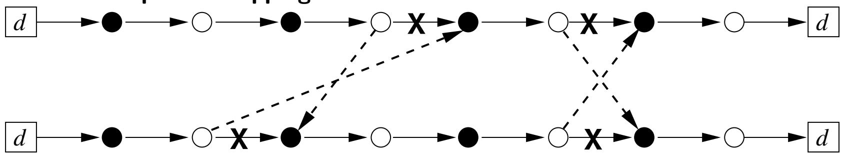
With respect to a given a feasible solution, let us define $z$ as the number of routes, $\omega_v$ the duration of route $v$, for $v = 1, \ldots, z$ , $\overline{\omega} = \max\{\omega_v : v = 1, \ldots, z\}$ the duration of the longest route, and $\omega = \sum_{v=1}^{V} \omega_v$ the total duration.

Any tour can be seen as an alternating sequence of $e_i$ and $f_j$ nodes, that is an alternating sequence of loaded and unloaded arcs. Tours start and end with unloaded arcs. Loaded arcs connect compatible nodes while unloaded arcs may connect nodes related to containers of any kind. Such particular graph topology suggests several neighborhood structures that can be exploited within a local search framework.

All the neighborhoods aim at reducing the total route lengths with inter-tour moves (neighborhoods $N_1$ and $N_2$) and intra-tour moves (neighborhood $N_3$): some neighborhoods can be seen as the specialization of classical VRP moves to $G$ (see, e.g., Aarts and Lenstra (1997), Toth and Vigo (2002)), while others have been suggested by the particular structure of $G$.

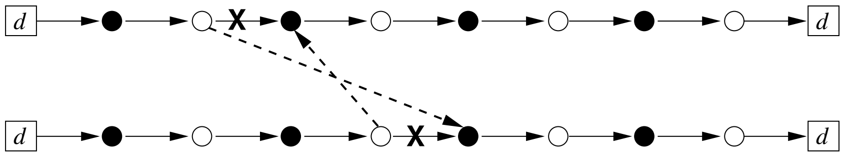
**Neighborhood $N_1$: String Exchange and Relocation of Unloaded Arcs** Consider any arc sequences $s_1$ and $s_2$ belonging to disjoint tours, that start from a node in $F \cup \Phi$ and end in a node of $E \cup \Upsilon$, that is sequences whose first and last arcs are loaded; let $\sigma_1 \geq \sigma_2 \geq 0$ be their sizes (i.e., their number of arcs). Note that in the degenerate case $\sigma_2 = 0$ and $s_2$ is the *null* sequence, denoted as $s_2 = \emptyset$, where the preceding unloaded arc coincides to the following unloaded arc. The two sequences are swapped as usual: $s_1$ and $s_2$ are removed from their tour by deleting the four unloaded arcs that precede or follow each sequence, and are inserted into the other tour, thus reconnecting each of them in the only possible way without reversing, by inserting four new unloaded arcs. An example is depicted in Figure 5.

**Figure 5** $N_1$ **move: an example of swapping**



We take into account some special inter-tour cases: 1) if both $s_1$ and $s_2$ contain all loaded arcs in their own tour, that is, the four unloaded arcs preceding and following the sequences belong to $\delta(d)$, then the move returns the original tours. 2) If $s_2 = \emptyset$, the move removes $s_1$ from its tour and *relocates* it into the tour of $s_2$. In particular, if $s_1$ contains all loaded arcs of its tour, this move merges one tour into another. 3) If both sequences are either at the end or at the beginning of their own tour, the move performs a *cross* (see the example depicted in Figure 6).

**Figure 6** $N_1$ **move: the case of a cross**



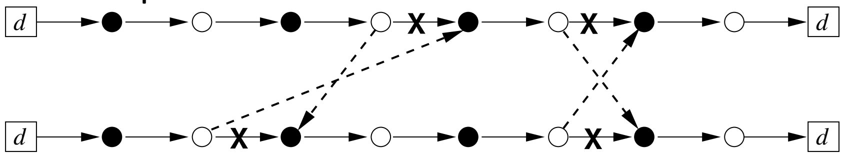
Finally, consider the intra-tour case, when $s_1$ and $s_2$ are disjoint sequences in the same tour. When $s_2 = \emptyset$, the move yields a *relocation* of $s_1$ before or after its current position. According to the classic moves description, such a move can be described as a 3-opt move without arc reversing, performed on a single tour and restricted to triplets in $A^U$. Relocation is the only intra-tour move allowed in $N_1$.

Move $N_1$ can be used to enforce an efficient pattern in the current solution related to swaps. Recall that an arc is said to be co-sited when its two vertices model requests that share the same RDS. In particular, loaded co-sited arcs model a petal pattern while unloaded co-sited arcs model a swap. Swaps can provide a substantial saving since have null travel time. If two co-sited nodes $e_i$

and $f_j$ belong to different tours, they can be sequenced by applying an $N_1$ move. Otherwise, when they belong to the same tour, but are not adjacent, they can be sequenced by a relocation where $s_1$ is made of node $f_j$ and its outgoing arc while $s_2 = \emptyset$ involves the unloaded arc outgoing from $e_i$. Note that the swap idea can be generalized to a pair of co-sited nodes of different container type. Due to the large dimension of neighborhood $N_1$, its exploration could be controlled by bounding $\sigma_1$.

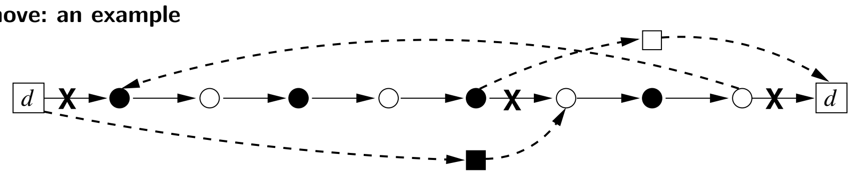
**Neighborhood $N_2$: String Cross of Loaded Arcs** The exploration of $N_2$ considers a pair of compatible loaded arcs in different tours. These arcs are then deleted, and the tours recomposed to yield a cross.

**Figure 7** $N_2$ **move: an example**



**Neighborhood $N_3$: Intra-tour Additional Empty Container** Moves generated by neighborhood $N_3$ exploits the availability of an additional container of a given type. This is suitably inserted into a tour, thus involving the reversing of some loaded arcs as depicted in Figure 8. The opposite move, which removes an additional empty container from a tour, is also considered in $N_3$.

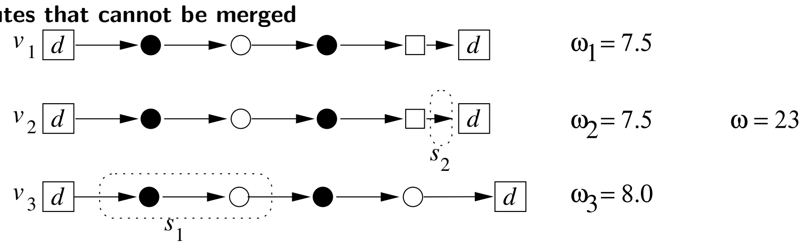
**Figure 8** $N_3$ **move: an example**



**4.3.1. From Tours to Routes** Given a set of tours satisfying our set of requests $R$, they have to be partitioned and merged to yield feasible routes. The neighborhoods illustrated before aim at reducing the total duration $\omega$, although case 2) of move $N_1$ decreases $z$ if the route of $N_1$ is made of a single tour. Indeed, a move that reduces total duration does not necessarily lead towards a set of tours that allows a better partitioning into routes. On the other hand, working directly on routes would require the use of complex moves, made of non monotonic steps with respect of total duration, aiming at spreading one route on to the others in a feasible way. The following example illustrates this situation.

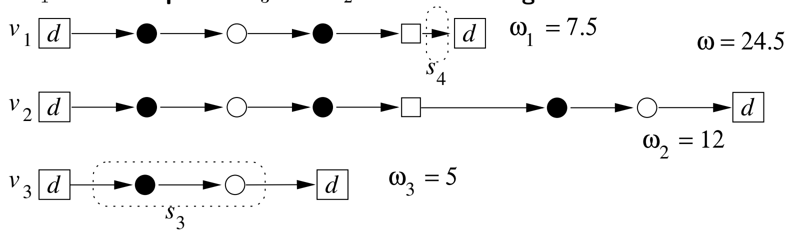
Figure 9 depicts 3 routes $v_1$, $v_2$ and $v_3$ whose durations are respectively 7.5, 7.5 and 8. The corresponding total duration $\omega$ is equal to 23. Note that $z$ =3 and no two routes can be merged

**Figure 9** **3 routes that cannot be merged**



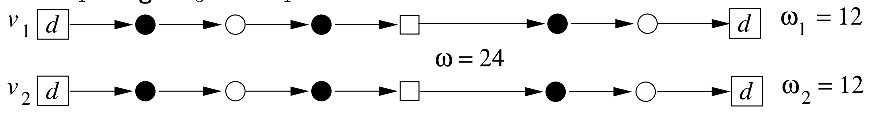
into a feasible one since $D = 13$. However, $z$ can be decreased by two non improving moves as follows.

**Figure 10** **A move $N_1$ relocates part of $v_3$ into $v_2$ while increasing $\omega$**



In Figure 9 are highlighted a pair of sequences $s_1$ and $s_2$ with $\sigma_2 = 0$, that are used in a $N_1$ type move yielding the routes in Figure 10. Note that $\omega$ increases.

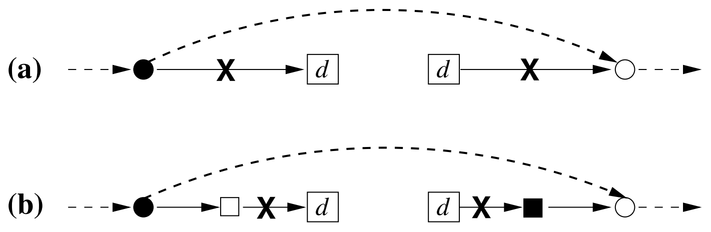
**Figure 11** **A move $N_1$ merges $v_3$ with $v_1$**



Now, a move of type $N_1$ on $s_3$ and $s_4$ merges $v_3$ into $v_1$ so that $z = 2$ as depicted in Figure 11.

We tackle this issue by working on two levels of granularity, i.e., tours and routes. The composition of tours into routes is directly dealt with by a Bin Packing approach. Considering the standard BPP (Martello and Toth 1990), objects and bins model tours and routes respectively, and each bin has a maximum capacity equal to $D$. Notice that a route (bin) is composed by a set of tours starting and ending at the depot $d$. In BPP the insertion order of the items within the same bin does not influence the residual capacity. On the contrary, according to how tours are sequenced in the route, some savings can be achieved. In particular there are two cases as depicted in Figure 12. First, when the last node of a tour is any $e_i$ and the first node of the following tour is any $f_j$, a shortcut that skips the depot is possible as in case (a). Second, when a tour ends with a node $v \in \Upsilon_k$ and the first node of the next tour is $\phi \in \Phi_k$, a shortcut skipping $v$, $\phi$ and $d$ is possible, as shown in case (b). Note that, the tours are not actually merged but the potential saving is considered when computing the duration of the route.

**Figure 12** **A shortcut skipping the depot (a) and a shortcut skipping the depot and saving one spare container (b)**



We generalize the well-known *Best Fit Decreasing* algorithm into the *modified Best Fit Decreasing* (M-BFD) in order to deal with the sequencing of tours taking into account the potential savings. A further version of M-BFD is also provided in order to deal with a fixed number of routes, and called M-BFD-F: the algorithm proceeds as M-BFD until it is possible to insert tours into routes maintaining the feasibility; as soon as the insertion of a tour makes infeasible the whole solution, M-BFD-F inserts a tour into a route $r$ in such a way to minimize the infeasibility of the routes, that is $\omega_r - D$. In the following, we refer to the two algorithms as M-BPH, i.e., Modified Bin Packing Heuristics.

**4.3.2. Infeasible Solutions and a Variable Objective Function** The exploration of infeasible solutions during the neighborhood search has been proved useful in determining better solutions for Vehicle Routing Problems as described in Gendreau et al. (1994), where overcapacity and overtime are allowed along the search although penalized by two self-adjusting parameters. Accordingly, we admit intermediate solutions which violate the duration constraint $\overline{\omega}$, i.e., some routes can last longer than $D$. In order to minimize both fleet size and total duration, the objective function leading the search must depend on both $z$ and $\omega$. At the same time, overtime is penalized as defined in formula (1): $\varphi(sol)$ is a variable objective function adding an extra cost to the cost of a solution $sol$ when $sol$ is infeasible; this extra cost is set to the maximum constraint violation $(\overline{\omega} - D)$ times the number of routes $z$.

$$\varphi(sol) = \begin{cases} z\,D + \omega\,, & \text{if } \overline{\omega} \leq D \\ z\,\overline{\omega} + \omega\,, & \text{if } \overline{\omega} > D \end{cases} \tag{1}$$

When $D < \overline{\omega}$, the inequality $z\,D + \omega < z\,\overline{\omega} + \omega$ holds, so that no feasible solution can be discarded in favor of an infeasible one with the same number of vehicles.

**4.3.3. Intensification and Diversification Strategies** In order to better explore the solution space, we introduce two strategies to intensify and diversify the search when particular conditions hold.

The intensification strategy consists in confining the search for a given number of iterations $k_{int}$ into a subregion characterized by a fixed number of routes, one less than the number of routes in the solution just before the intensification. Clearly, this can lead far out of the feasible region, but the variable objective function $\varphi(sol)$ is designed so that it leads the search back towards feasibility, within the forced subregion. The idea of forcing the reduction of (quite) flat objective functions adopting also a variable objective function has been proved effective in Aringhieri and Dell'Amico (2005).

Whenever M-BPH decreases $z$ to $z' < z$ is the condition needed to start the intensification strategy. In order to obtain the reduced solution (in terms of number of routes), the tours belonging to the shortest route are spread on to the other routes obtaining a solution with $z' - 1$. In the following $k_{int}$ iterations, M-BPH is forced to use $z' - 1$ bins in order to allow neighborhood search to reach again the feasibility. If feasibility is not regained, the search resumes from a solution with at least $z'$ routes at the next application of M-BPH without restrictions.

We observed that the neighborhood search and M-BPH tend to generate routes with bigger, at least one, tour inside. The diversification strategy occurs after $k_{div}$ consecutive iterations monotonically non decreasing, as follows: all routes composed by one tour are recomputed by splitting the single tour in two separate tours, and the broken arcs are made tabu in order to avoid successively merging. The basic idea is to give more degree of freedom both to neighborhood search and to the M-BPH.

**4.3.4. HNS Algorithm** Finally, we briefly describe the whole algorithm putting together all the building blocks discussed before. The pseudo code is described in Algorithm 1. Our algorithm for E1LCRP is a Hierarchical Neighborhood Search (HNS) that starts its search from the solution $sol_0$ computed by M-CW. The initial solution is then improved by a neighborhood search (lines 3– 18) and by the exploitation of some intensification (lines 19– 29) and diversification (line 31) strategies. At the end a new iteration starts until the maximum number of iterations $k_{\max}$ has been reached. The solution obtained at the $k - th$ iteration is indicated with $sol_k$.

According to the computational experience on the Capacitated Asymmetric VRP reported in Vigo (1996), better results can be achieved by using a larger neighborhood obtained by the union of the moves provided by the single neighborhoods, namely *relocate*, *exchange* and *cross*, rather than cycling iteratively from one to the other. We adopt these advice concerning $N_1$ and $N_2$, while we resort to $N_3$ as a sort of mild diversification move.

Therefore, our neighborhood search generates new solutions, exploring $N = N_1 \cup N_2 \cup N_3$, which are evaluated using the variable objective function $\varphi(sol)$ described by formula (1): the best move, taking into account also the solutions satisfying the usual aspiration criteria, are then applied. If this move decreases $\omega$, we apply the M-BPH trying to reduce the total number of routes $z$ . Two tabu lists of fixed length are used: $\ell_1$ avoids moves involving nodes used in the previous $|\ell_1|$ moves whilst $\ell_2$ avoids a node to return to the tour from which it was removed for $|\ell_2|$ moves. Clearly, $|\ell_1| < |\ell_2|$. After each move, the conditions which control the intensification and the diversification strategies are verified and, if satisfied, the corresponding strategy is applied.

**Algorithm 1** Pseudo code for HNS

```
1: $sol_0 = $ M-CW; $sol^* = sol_0$; $k = 1$; intPhase = divPhase = false; $k_d = 0$;
2: while $(k \le k_{\max})$ do
3:     if $(\neg(\text{intPhase} \vee \text{divPhase}))$ then                ▷ Neighborhood Search
4:         $sol_k = \arg\min\{\varphi(sol) : sol \in N_1 \cup N_2 \cup N_3\}$;
5:         if $(\omega(sol_k) < \omega(sol_{k-1}))$ then
6:             $sol_k = $ M-BPH$(sol_k)$;
7:             if $(\varphi(sol_k) < \varphi(sol^*))$ then $sol^* = sol_k$ end if
8:             if $(z\ (sol_k) < z\ (sol_{k-1}))$ then
9:                 $z' = z\ (sol_k) - 1$;
10:                $sol_k = $ spreadShortestRouteTours$(sol_k)$;
11:                intPhase = true; $k_i = 0$;
12:            end if
13:            $k_d = 0$;
14:        else
15:            $k_d = k_d + 1$;
16:            if $(k_d \ge k_{div})$ then divPhase = true end if
17:        end if
18:        $k = k + 1$;
19:    else if ( intPhase ) then                ▷ Intensification Phase
20:        $sol_k = \arg\min\{\varphi(sol) : sol \in N_1 \cup N_2 \cup N_3\}$;
21:        $sol_k = $ M-BPH$(sol_k, z')$;                ▷ Force M-BPH to use $z'$ bins
22:        if (isFeasible$(sol_k)$) then
23:            intPhase = false;
24:            if $(\varphi(sol_k) < \varphi(sol^*))$ then $sol^* = sol_k$ end if
25:        else
26:            $k_i = k_i + 1$;
27:            if $(k_i > k_{int})$ then intPhase = false end if                ▷ Temporary infeasible solution
28:        end if
29:        $k = k + 1$;
30:    else                ▷ Diversification Phase
31:        $sol_k = $ splitSingleTourRoutes$(sol_{k-1})$; divPhase = false; $k_d = 0$; $k = k + 1$;
32:    end if
33: end while
```

## 5. Lower Bounds

E1LCRP is a special case of Distance constrained uncapacitated VRP (D-VRP) D-VRP arises whenever in a VRP the sum of the (positive) arc weights in each route is bounded from above. D-VRP is not as much studied as its two well known generalization, namely the Distance constrained Capacitated VRP (DCVRP)and the TW-VRP, and despite its many applications, tailored approaches for the pure D-VRP are still few (see Laporte et al. (1984), Li et al. (1992) for the exact approaches and Nagarajan and Ravi (2012) for approximation of the min fleet size D-VRP under different metrics). Actually, D-VRP is a relaxation of the formers since it lacks the additional structure provided by the capacity constraint, which allows for Bin Packing Problem relaxations in DC-VRP, as well as the partial order relation on the requests that time windows induce when they differ from one another, as in TW-VRP. This lack of structure makes tight lower bounds hard to compute.

The objective function of E1LCRP is twofold, considering both total duration $\omega$ and fleet size $z$. Since $z$ is the leading component, the total traveling time of any solution to E1LCRP is bounded from below by the shortest duration $\omega^*$ of the single-objective D-VRP minimizing $\omega$, so we are interested in lower bounds for each of both objectives in a single-objective D-VRP formulation. Moreover, although $\omega$ and $z$ are potentially conflicting, any lower bound on either $z$ or $\omega$ can be exploited to provide a valid lower bound on the other component.

Deriving a tight lower bound on $z^*$ can be quite challenging if neither $\omega^*$ nor a lower bound $\omega^{LB}$ are available, whereas a straightforward one can be computed as the ceiling of the ratio of $\omega^*$ over $D$, i.e., $z^{LB} = \lceil \omega^*/D \rceil \leq z^*$, or $z^{LB} = \lceil \omega^{LB}/D \rceil \leq \lceil \omega^*/D \rceil \leq z^*$.

Given either $z^*$ or $z^{LB}$, a straightforward, though quite loose, lower bound on the total duration is given by $\omega^{LB} = (D/2+1)\,(\lceil z^{LB} \rceil - 1) + D/2$. Indeed, if at least two vehicles were traveling no longer than $D/2$, their routes could be merged, yielding a time-feasible solution with one less $z^{LB} - 1$ vehicles.

More sophisticated lower bounds can be obtained by mathematical programming relaxation techniques of different MILP models, as for example in Baldacci et al. (2006) where the set partitioning formulation is exploited, providing tight but computationally expensive bounds. On the contrary, hereafter we consider a MILP model for E1LCRP whose relaxation can be easily computed by a commercial solver for real life size instances.

### 5.1. A Mixed Integer Linear Programming Model

E1LCRP can be formulated as a Mixed-Integer Linear Programming (MILP) problem by an arc based flow model on the following extended graph $G' = (N, A')$. In order to model fleet size as an objective function, each arc outgoing from the depot node $d$ is associated with the use of an additional vehicle. Therefore, the arc set $A$ needs to be extended to $A' = A \cup \Lambda$, where $\Lambda = \{(v, \phi) :$

$\upsilon \in \Upsilon_k$, $\phi \in \Phi_{k'}$, $\forall k, k'$, $k \neq k'\}$, is a new set of arcs that explicitly model the change of container type from $k$ to $k'$ which takes place at the depot along one route. These arcs have cost $\tau^L$, that is the time required to load a container. Note that in $G'$ a route is an elementary cycle containing $d$ while a tour in general is not. Solving the corresponding D-VRP on $G'$ would solve E1LCRP, and this provides a mathematical model for the problem, as follows.

Consider a binary variable $x_{uv}$ for each arc $(u,v)$ in $A'$ and a continuous variable $t_u \geq 0$ for each node $u \in N$, $t_d = 0$. Let $\delta^+(u) := \{v \in N : (u,v) \in A'\}$ and $\delta^-(u) := \{v \in N : (v,u) \in A'\}$ for all $u \in N$, as usual. Recall that $D$ is the maximum route duration. A MILP model for E1LCRP follows.

$$\min \quad (M\, z + \omega) \tag{2}$$

$$\sum_{v \in \delta^+(d)} x_{dv} = z \tag{3}$$

$$\sum_{(u,v) \in A'} c_{uv} x_{uv} = \omega \tag{4}$$

$$\sum_{v \in \delta^+(u)} x_{uv} = 1 \qquad \forall u \in F \cup E \tag{5}$$

$$\sum_{v \in \delta^+(u)} x_{uv} \leq 1 \qquad \forall u \in \Phi \cup \Upsilon \tag{6}$$

$$\sum_{v \in \delta^+(u)} x_{uv} = \sum_{v \in \delta^-(u)} x_{vu} \qquad \forall u \in N \tag{7}$$

$$\sum_{(u,v) \in A'} c_{uv} x_{uv} \leq D \sum_{v \in \delta^+(d)} x_{dv} \tag{8}$$

$$t_u + c_{uv} x_{uv} \leq t_v + \tilde{D}\,(1 - x_{uv}) \qquad \forall (u,v) \in A' \setminus \{(u,d) : u \in \delta^-(d)\} \tag{9}$$

$$t_u + c_{ud} x_{ud} \leq D \qquad \forall u \in \delta^-(d) \tag{10}$$

$$x_{uv} \in \{0,1\} \qquad \forall (u,v) \in A' \tag{11}$$

$$t_u \geq 0 \qquad \forall u \in N \tag{12}$$

The objective function (2) is the weighted sum of the fleet size component modeled by $z$, defined by constraints (3) (one vehicle for each route), and the total duration component, modeled by $\omega$, defined by constraints (4). By setting $M$ to a suitably big constant, fleet size minimization becomes the main objective so that a solution with a minimum total duration is selected among those having the minimum fleet size. Constraints (5) impose that requests are covered exactly once; constraints (6) ensure that each additional container be used at most once; constraints (7) are flow conservation constraints; constraint (8) relates fleet size to total duration, since it imposes that the number of routes must be not lower than the total duration divided by the maximum route duration; constraints (10) impose maximum route duration. Constraints (9) are the classical Miller-Tucker-Zemlin like subtour breaking constraints, strengthened by setting $\tilde{D} = D - \tilde{t}_{ud} - \tilde{t}_{dv}$, where $\tilde{t}_{ud}$ denotes the duration of the shortest path from node $u$ to the depot ($\tilde{t}_{ud} = t_{ud}$ for $u \in E \cup \Upsilon$), and $\tilde{t}_{dv}$ denotes the duration of the shortest path from the depot to node $v$ ($\tilde{t}_{dv} = t_{dv}$ for $v \in F \cup \Phi$).

In order to compute a lower bound on total duration $\omega^{LB}$ in a reasonable time we solve (2)–(12) by setting $M = 0$, relaxing (11) into $x_{uv} \leq 1$, and imposing integrality on $z = \sum_{u \in \delta^+(d)} x_{du}$ that is the outflow of the depot. A trivial lower bound on $z$ is obtained as mentioned at the beginning of the section that is $z^{LB} = \lceil \omega^{LB}/D \rceil$.

## 6. Experiments and Results

The aim of this section is to provide some computational insights regarding the capability of the proposed algorithm HNS. To this end, we report an extensive computational analysis solving three sets of instances based on real and realistic data. Afterward, we test our approach on benchmark instances from the Rollon-Rolloff literature (the 20 test instances introduced in Bodin et al. (2000)) to verify its capability in handling problems with similar structure and compare with algorithms specialized for the original Rollon-Rolloff problem.

Section 6.1 concerns real instances and Section 6.2 the realistic ones, built to have more challenging tests, providing all the technical specifications about the data generation process. In Section 6.3, we evaluate the impact on the solution quality and the running time of each component of HNS. Section 6.1 and Section 6.5 report a comparison of HNS with the lower bound obtained by solving the MILP model in Section 5 with a general purpose MILP solver.

The proposed algorithm is coded in standard C and compiled by gcc 4.6.0 and the bound is computed using Gurobi 4.5.1. Parameters $k_{\max}$ and $k_{div}$ have been calibrated on each instance set, according to the problem size, while the values of $\sigma_1$ and $\sigma_2$ in neighborhood $N_1$ are not relevant since the route duration limit keeps the average number of nodes in a route sufficiently small. All the computational tests are performed on a 2.40 GHz Intel Xeon E5620 double processor, with 18 GB of memory, under Linux operating system.

### 6.1. Results on real instances

The first set of instances $B_1$ was provided by Gesenu and describes real life cases in representative days that were perceived as challenging by the company, given that, at present, the scheduling is solved by hand. Each instance is characterized by the number of service requests and the number of available containers at the depot. Set $B_1$ is made of problems with $|R| \leq 11$ and 6 containers of different types. This yields graphs having at most 34 nodes, since the number of nodes $|N|$ depends on $|R|$ according to $2|R|+1 \le |N| \le 4|R|+1$.

Computational results are reported in Table 3: instance name is in the first column. We compare four solutions: the actual solution adopted in Gesenu (column *Company*), the exact solution obtained by solving the MILP model presented in Section 5 (column MILP) by Gurobi, the solution computed after the constructive step (column M-CW), and the solution computed by our algorithm (column HNS). For each solution, the number of vehicles ($z$) and the total duration ($\omega$) of the routes are reported. Parameter calibration on the $B_1$ instances yielded $k_{\max} = 500$ and $k_{div} = 50$.

**Table 3 Comparing results for real instances with $|N| \le 34$**

| | Company | | MILP solver | | M-CW | | HNS | |
|---|---|---|---|---|---|---|---|---|
| Instances | $z$ | $\omega$ | $z$ | $\omega$ | $z$ | $\omega$ | $z$ | $\omega$ |
| B1.0 | 3 | 789 | 2 | 668 | 2 | 668 | 2 | 668 |
| B1.1 | 3 | 813 | 2 | 701 | 2 | 709 | 2 | 709 |
| B1.2 | 3 | 686 | 2 | 665 | 2 | 665 | 2 | 665 |
| B1.3 | 3 | 1001 | 3 | 843 | 3 | 966 | 3 | 843 |
| B1.4 | 3 | 975 | 3 | 839 | 3 | 839 | 3 | 839 |
| B1.5 | 3 | 1075 | 3 | 878 | 3 | 878 | 3 | 878 |

The solutions provided by the company are mainly based on the simplest procedure from an organizational point of view, that is the petal pattern. This implies that the company does not exploit the availability of empty containers which are only used as a safety stock. On the contrary, our approach fully exploits this fact, modeling the two actions as distinct nodes in our network model. This allows to obtain better solutions than those provided by the company even on instances small enough to be within the reach of a skilled operator. The quality of our solutions is also certified by the comparison with the optimal ones computed by Gurobi. It can be observed that in all but one case the constructive step already computes the (sub) optimal solution returned by HNS, thus confirming the findings reported in De Muelemeester et al. (1997) regarding the performance of the Clarke and Wright procedure, of which our M-CW is a variant.

## 6.2. Setting up the computational experiments: realistic instances

In order to test our approach on more challenging instances, we build realistic instances exploiting the real data on the Gesenu service network made of the actual depot, RDSs, and MRFs, using real travel and service times. In fact, based on the existing facilities of the company, if we suppose that all of the 10 different waste materials can be stored at each of the 10 RDS, we can generate instances with up to 100 different service requests. In case of unbounded spare containers the corresponding graph may reach 401 nodes. The set of realistic instances $B_2$ is generated by randomly selecting the required number $|R|$ over the 100 possible service requests. In detail, $B_2$ consists of 5 subsets of

10 instances each, differing for the number of requests, i.e., cardinality of $|R| = 20, 30, 40, 60, 80$. Each instance is replicated by changing the *container setup*, i.e., the number of empty containers available at depot: type "0", "1", "$\mu$", and "$\infty$" denotes 0, 1, an *average number* and a *large number* of containers available for each container type whilst "def" denotes the actual company container stock, i.e. six containers of different types. The $\infty$ setup consists in fixing the number of empty containers equal to the number of requests requiring this type of container; the $\mu$ setup is given by the ratio of the number of requests requiring a given container over the average number of requests.

In order to generate larger instances, we added 16 new RDSs in such a way to cover all the main areas of Umbria region, anticipating the effects of a potential aggregation of this service management activity which is currently operated independently by the different local providers. A picture describing the complete RDS network is reported in Aringhieri et al. (2008). The new setting allows the generation of the benchmark set $B_3$, composed of more challenging instances up to $|R| = 260$: 10 with $|R| = 100$, 10 with $|R| = 150$ and 10 with $|R| = 200$. Each instance has two different versions: $D_0$ has the $|R|$ requests clustered on the minimum number of RDSs (e.g., $|R| = 100$ requests are clustered on 10 RDSs randomly selected) while $D_1$ equally distributes them among all the RDSs (e.g., each RDS has about $\frac{|R|}{26}$ requests). Finally, each instance has been replicated changing the *container setup* as described before. We do not report the results for the $\mu$ setting since they coincide with the $\infty$ due to the fact that even in the larger instances ($n = 200$) the use of spare containers in any solution is much below the number of available ones.

In terms of number of instances, we have $|B_2| = 250$ and $|B_3| = 120$. Table 4 summarizes the features of the benchmarks in $B_2$ and $B_3$. For all instances, the maximum route duration is $D = 375$. All the three benchmark sets are available at http://www.di.unito.it/˜aringhie/benchmarks.html.

**Table 4 Description of benchmarks $B_2$ and $B_3$.**

| | $\|R\|$ | 0 | | 1 | | $\infty$ | | $\mu$ | | def | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\|N\|$ | # inst. | $\|N\|$ | # inst. | $\|N\|$ | # inst. | $\|N\|$ | # inst. | $\|N\|$ | # inst. |
| | 20 | 41 | 10 | 51 | 10 | 81 | 10 | 47 | 10 | 53 | 10 |
| | 30 | 61 | 10 | 71 | 10 | 121 | 10 | 71 | 10 | 73 | 10 |
| $B_2$ | 40 | 81 | 10 | 91 | 10 | 161 | 10 | 99 | 10 | 93 | 10 |
| | 60 | 121 | 10 | 133 | 10 | 241 | 10 | 147 | 10 | 133 | 10 |
| | 80 | 161 | 10 | 173 | 10 | 321 | 10 | 195 | 10 | 173 | 10 |
| | | | 50 | | 50 | | 50 | | 50 | | 50 |
| | 100 | 201 | 10 | 213 | 10 | 401 | 10 | – | – | 241 | 10 |
| $B_3$ | 150 | 301 | 10 | 313 | 10 | 601 | 10 | – | – | 341 | 10 |
| | 200 | 401 | 10 | 413 | 10 | 801 | 10 | – | – | 441 | 10 |
| | | | 30 | | 30 | | 30 | | | | 30 |

The HNS parameters have been calibrated by a series of preliminary computational tests and have been set to the following values for both $B_2$ and $B_3$: $k_{\max} = 60000$, $k_{div} = 300$, $k_{int} = 20$, $\sigma_1 = \sigma_2 = 25$, $\ell_1 = 5$ and $\ell_2 = 20$.

## 6.3. Evaluation of the algorithm components

In this section we compare the performance of several variants of HNS obtained by enabling or disabling the *ad hoc* components of the algorithm that is: the infeasibility handled by the objective function, the set of restarts (intensification and diversification strategies) and the use of neighborhood $N_3$ as a sort of mild diversification. The experiments have been performed on the 50 instances in $B_2$ having $|R| = 40$, which are sufficiently difficult not to be solved to optimality by the solver, but for which good lower bounds are attained.

Table 5 reports the results of the computational test: the first four columns summarize the mix of tested components; the fifth column reports the total duration average gap with respect to the available exact solutions; the sixth and seventh columns report for how many instances the HNS solution reached the lower bound on the number of vehicles and the lower bound plus 1, respectively. For all variants running times are comparable. Notice that the intensification component requires the management of infeasibility.

**Table 5** **evaluating HNS components ("n.a." means not applicable).**

| Mix of | HNS with: | | | avg. gap | # instances | |
|---|---|---|---|---|---|---|
| components | infeasibility | intensification | $N_3$ | $(\omega - \omega_b)/\omega_b$ | $z_b$ | $z_b + 1$ |
| (A) | × | × | × | 0.44% | 32 | 18 |
| (B) | × | × | | 0.82% | 30 | 20 |
| (C) | × | | × | 0.72% | 27 | 23 |
| (D) | × | | | 1.41% | 26 | 24 |
| (E) | | n.a. | × | 0.79% | 29 | 21 |
| (F) | | n.a. | | 0.83% | 28 | 22 |

The analysis proves the effectiveness of *ad hoc* components for HNS when they are all enabled (mix (A)). In particular the addition of neighborhood $N_3$ has always a positive impact. Moreover also infeasibility seems a crucial issue: without handling infeasibility (mix (E) and (F)) in HNS, the gap on $\omega$ doubles, more vehicles are used and computational times are longer. However, infeasibility deteriorates the search when used alone (mix (D)).

## 6.4. Results on large realistic instances

Benchmarks $B_2$ and $B_3$ are introduced to evaluate the algorithm behavior on a larger but lifelike settings.

Table 6 reports the comparison between HNS and M-CW as far as the gap with the lower bound on the number of vehicles $z$ is concerned: gaps from 0 to 6 and $\geq 7$ are considered separately. The

**Table 6 (HNS,M-CW) vs. lower bound: number of vehicles.**

| | $\|R\|$ | # inst. | absolute gaps w.r.t. lower bound | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | $\geq 7$ | |
| | 20 | 50 | 42 | 7 | 8 | 32 | 0 | 10 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 30 | 50 | 35 | 0 | 14 | 0 | 1 | 26 | 0 | 20 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 |
| $B_2$ | 40 | 50 | 32 | 1 | 18 | 15 | 0 | 29 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 60 | 50 | 19 | 0 | 31 | 1 | 0 | 15 | 0 | 22 | 0 | 12 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 80 | 50 | 17 | 0 | 30 | 0 | 3 | 2 | 0 | 14 | 0 | 26 | 0 | 8 | 0 | 0 | 0 | 0 |
| | 100 | 40 | 0 | 0 | 2 | 0 | 5 | 0 | 12 | 1 | 13 | 2 | 2 | 4 | 0 | 2 | 6 | 31 |
| $B_3$ | 150 | 40 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 8 | 0 | 4 | 0 | 7 | 1 | 20 | 39 |
| | 200 | 40 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 6 | 0 | 3 | 0 | 30 | 40 |

value of the gap is reported at the heading of each column. Below, for each such column and for each set of instances in $B2$ and in $B3$, the number of instances solved by HNS (on the left) and by M-CW (on the right) with such gap is reported. The gap is computed with respect to the lower bound introduced at the end of Section 5.

As expected, the gap provided by HNS is quite small for the instances in $B_2$ while the quality deteriorates for the instances in $B_3$. Concerning benchmark set $B3$, no instance can be solved with gap 0, and for 150 and for 200 requests only one instance can be solved with gap 3 and 4, respectively, while most instances are solved with gaps higher than 6. Probably this is also due to the bound deterioration.

Furthermore, comparing the results of HNS with those of M-CW, it is evident the capability of HNS to largely improve the quality of the initial solution provided by M-CW which increases with the size of the instances: actually, HNS on $B_2$ computes a solution having a gap equal to 2 only for 4 instances, otherwise the gap is less than or equal to 1; on the contrary, M-CW is able to compute a gap less than or equal to 1 only for 56 while the remaining 194 have a gap equal to or greater than 2.

## 6.5. Comparison with the MILP solver

In the following we present a sequence of Tables reporting a comparison with the MILP solver. Table 7 reports an *equal effort* comparison meaning that we use the MILP solver in a single-thread mode imposing a time limit equal to the average total running time of HNS when solving instances having the same number of requests $R$. We report the number of instances for which HNS computes a better solution than the MILP solver, and vice versa. We propose two types of comparison: the first one takes into account the total duration $\omega$ of the solution when the value of $z$ is the same; the second one takes into account only $z$. The last column reports also the number of instances in which the MILP solver “fails”, that is when it reaches the time limit without providing a feasible solution.

**Table 7 HNS vs. MILP solver: equal effort comparison.**

| | | | comparing $z$ and $\omega$ | | | comparing only $z$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\|R\|$ | # inst. | HNS | MILP | tie | HNS | MILP | tie | MILP fail |
| $B_2$ | 20 | 50 | 4 | 16 | 30 | 3 | 2 | 45 | 0 |
| | 30 | 50 | 3 | 18 | 29 | 3 | 2 | 45 | 0 |
| | 40 | 50 | 6 | 16 | 28 | 3 | 1 | 46 | 0 |
| | 60 | 50 | 42 | 7 | 1 | 34 | 0 | 16 | 0 |
| | 80 | 50 | 47 | 3 | 0 | 46 | 0 | 4 | 3 |
| $B_3$ | 100 | 40 | 32 | 8 | 0 | 31 | 5 | 4 | 0 |
| | 150 | 40 | 31 | 9 | 0 | 30 | 8 | 2 | 10 |
| | 200 | 40 | 36 | 4 | 0 | 35 | 4 | 1 | 9 |

The results showed the superiority of HNS as soon as the number of requests is greater than or equal to 60. Note that taking into account only the value of $z$, we remark that our algorithm provides the best solution more often than the MILP solver. As expected when the number of requests increases the MILP solver fails more frequently. Conversely, the MILP solver seems competitive only when solving instances up to $R = 40$.

The same tests reported have been carried out also using the MILP solver at its maximum potentiality exploiting multi thread computation (15 threads) and increasing the time limit (2400 seconds of wall time). Even if the solution quality of the MILP solver improves (especially for instances in $B_2$), such tests confirm again the effectiveness of HNS. Unexpectedly, the number of fails counted for instance in $B_3$ largely increases.

Table 8 reports the average running time in seconds of HNS and the MILP solver. It also reports the average running time of computing the first solution having the same value of $z$ and $\omega$ of the best solution yielded by HNS at the end of the computation. While the MILP solver is generally faster than HNS on smaller instances, we remark that HNS requires, on average, the 5.05% and the 21.83% of the whole running time to compute a solution having the best possible value of $z$ for instances in $B_2$ and $B_3$, respectively.

**Table 8 HNS vs. MILP solver: average running time in seconds. For HNS the iteration and time at which the best solution is reached are reported.**

| | | | total running time | | HNS | | | |
|---|---|---|---|---|---|---|---|---|
| | $\|R\|$ | # inst. | HNS | MILP | 1st iter $z$ | time | 1st iter $\omega$ | time |
| $B_2$ | 20 | 50 | 103.3 | 44.6 | 1084 | 1.8 | 16347 | 28.1 |
| | 30 | 50 | 251.7 | 126.0 | 2543 | 10.7 | 18238 | 76.5 |
| | 40 | 50 | 566.7 | 339.5 | 786 | 7.4 | 14095 | 133.1 |
| | 60 | 50 | 559.1 | 549.9 | 3688 | 34.4 | 28014 | 261.0 |
| | 80 | 50 | 1228.1 | 1228.7 | 7061 | 144.5 | 33253 | 680.7 |
| $B_3$ | 100 | 40 | 3643.9 | 3644.4 | 13246 | 804.5 | 27323 | 1659.4 |
| | 150 | 40 | 12699.6 | 12700.6 | 10541 | 2231.2 | 18797 | 3978.7 |
| | 200 | 40 | 24816.2 | 24817.5 | 15496 | 6409.4 | 19512 | 8070.3 |

### 6.6. The impact of spare containers on the hardness of the problem

As mentioned in the paper, the peculiarity of our approach is to explicitly deal with a limited number of spare containers. The main concern of this section is to evaluate, from a computational point of view, the impact of the amount of spare containers on the hardness of the problem. Our claim is that a limited number of spare containers makes more difficult the problem. To support our claim we investigate how the gap on $z$ changes with respect to the different spare containers settings. We present results on the $B3$ instances disregarding $B2$ since the majority of the instances in $B2$ are solved with a null gap on the fleet size.

**Table 9 Comparing the gap on $z$ of HNS and M-CW; instances with $|R| = 100, 150, 200$.**

| | | 0 | | 1 | | def | | $\infty$ | |
|---|---|---|---|---|---|---|---|---|---|
| Instances | | HNS | M-CW | HNS | M-CW | HNS | M-CW | HNS | M-CW |
| B3.0.R100 | $D_0$ | 4 | 12 | 8 | 11 | 4 | 7 | 3 | 5 |
| B3.1.R100 | $D_0$ | 4 | 11 | 4 | 12 | 3 | 8 | 3 | 3 |
| B3.2.R100 | $D_0$ | 4 | 9 | 4 | 10 | 3 | 6 | 2 | 4 |
| B3.3.R100 | $D_0$ | 4 | 10 | 4 | 11 | 5 | 5 | 3 | 6 |
| B3.4.R100 | $D_0$ | 7 | 9 | 10 | 10 | 4 | 7 | 2 | 5 |
| B3.0.R100 | $D_1$ | 8 | 10 | 4 | 12 | 5 | 6 | 3 | 6 |
| B3.1.R100 | $D_1$ | 8 | 10 | 4 | 10 | 3 | 8 | 1 | 5 |
| B3.2.R100 | $D_1$ | 3 | 11 | 3 | 10 | 3 | 5 | 2 | 5 |
| B3.3.R100 | $D_1$ | 4 | 8 | 2 | 7 | 2 | 5 | 1 | 4 |
| B3.4.R100 | $D_1$ | 13 | 14 | 4 | 16 | 3 | 11 | 3 | 7 |
| | | 59 | 104 | 47 | 109 | 35 | 68 | 23 | 50 |
| B3.0.R150 | $D_0$ | 15 | 17 | 15 | 17 | 6 | 18 | 4 | 8 |
| B3.1.R150 | $D_0$ | 12 | 19 | 7 | 20 | 7 | 17 | 4 | 7 |
| B3.2.R150 | $D_0$ | 6 | 14 | 5 | 15 | 5 | 15 | 4 | 6 |
| B3.3.R150 | $D_0$ | 17 | 19 | 13 | 19 | 7 | 20 | 4 | 7 |
| B3.4.R150 | $D_0$ | 11 | 18 | 14 | 18 | 6 | 20 | 4 | 7 |
| B3.0.R150 | $D_1$ | 13 | 19 | 7 | 19 | 19 | 21 | 4 | 8 |
| B3.1.R150 | $D_1$ | 10 | 14 | 7 | 14 | 5 | 14 | 4 | 8 |
| B3.2.R150 | $D_1$ | 7 | 17 | 13 | 18 | 6 | 18 | 4 | 8 |
| B3.3.R150 | $D_1$ | 15 | 18 | 6 | 18 | 6 | 19 | 3 | 7 |
| B3.4.R150 | $D_1$ | 7 | 15 | 7 | 16 | 6 | 17 | 5 | 9 |
| | | 113 | 170 | 94 | 174 | 73 | 179 | 40 | 75 |
| B3.0.R200 | $D_0$ | 13 | 21 | 13 | 22 | 9 | 24 | 6 | 10 |
| B3.1.R200 | $D_0$ | 17 | 23 | 10 | 24 | 9 | 23 | 5 | 11 |
| B3.2.R200 | $D_0$ | 14 | 22 | 9 | 23 | 8 | 23 | 6 | 8 |
| B3.3.R200 | $D_0$ | 10 | 20 | 10 | 21 | 9 | 21 | 5 | 10 |
| B3.4.R200 | $D_0$ | 18 | 25 | 20 | 26 | 9 | 26 | 5 | 8 |
| B3.0.R200 | $D_1$ | 9 | 21 | 9 | 21 | 10 | 21 | 4 | 10 |
| B3.1.R200 | $D_1$ | 9 | 19 | 8 | 20 | 9 | 19 | 5 | 10 |
| B3.2.R200 | $D_1$ | 21 | 22 | 12 | 24 | 8 | 26 | 6 | 10 |
| B3.3.R200 | $D_1$ | 16 | 21 | 20 | 23 | 21 | 23 | 5 | 9 |
| B3.4.R200 | $D_1$ | 20 | 21 | 9 | 21 | 18 | 22 | 5 | 8 |
| | | 147 | 215 | 120 | 225 | 110 | 228 | 52 | 94 |

A comparison of the gap between $z$ and the lower bound for HNS and M-CW on the instances in $B_3$ is reported in Table 9. For each instance, a pair of columns report the absolute gap for the four different spare container settings. The last row reports the sum of the gaps on the corresponding column. Note that instances are identified through the random seed used in the generator, the requests number and their version ($D_0$ or $D_1$).

The results for instance with $|R| = 100$ shows a worsening of the gap as soon as the number of the spare containers reduces: actually, the sum of the gaps for HNS and M-CW are respectively 23 and 50 with setting "$\infty$" while is 47 and 109 with setting "1" and 59 and 104 with setting "0". Note that the same behavior is confirmed for instances with $|R| = 150$ and $|R| = 200$ as reported in the remaining of Table 9. Let us remark that the bound obtained for the setting "$\infty$" are also a bound for the other settings, but not vice versa.

Furthermore, the average gap of HNS and M-CW is respectively 12.05% and 27.12% for instances with $|R| = 100$, 15.66% and 29.41% for instances with $|R| = 150$, and 15.55% and 27.70% for instances with $|R| = 200$. This means that the average gap is quite stable with respect to the size of the instance considered.

Finally, we would like to analyze the number of vehicles used to show the impact of a different spare container setting on the final solution. Table 10 reports the average number of vehicles computed by HNS and M-CW for the instances in $B_3$ for "0", "1", "def" and "$\infty$" spare container settings.

**Table 10 Average number of vehicles used by HNS and M-CW.**

| $\|R\|$ | 0 | | 1 | | def | | $\infty$ | |
|---|---|---|---|---|---|---|---|---|
| | HNS | M-CW | HNS | M-CW | HNS | M-CW | HNS | M-CW |
| 100 | 40.8 | 45.3 | 38.7 | 44.9 | 36.2 | 43.0 | 34.5 | 37.2 |
| 150 | 63.8 | 69.5 | 60.9 | 68.9 | 57.1 | 67.7 | 52.2 | 55.7 |
| 200 | 85.5 | 92.3 | 81.7 | 92.2 | 78.7 | 90.5 | 70.1 | 74.3 |

The results confirm the claim reported above. Moreover, it is worth noting that the number of vehicles used with setting "0" are about the 20% more than those used with setting "$\infty$". In addition, note that these results also confirm the capability of HNS to improve the solution computed by M-CW as already reported in Table 6.

## 6.7. Tests on Rollon-Rolloff benchmark instances

In section 2 we pointed out that the E1LCRP can be used to model other problems in the class of Rollon-Rolloff VRP by suitably adapting the input. For this reason, even if our algorithm is not specialized for that class of problems, we carried out some experiments on the benchmark instances proposed in Bodin et al. (2000) and used also in Derigs et al. (2013) and in Wy and Kim (2013).

The benchmark instances include T1, T2, T3 and T4 services and unbounded spare containers of a single type, located at MRF. In order to adapt the input to E1LCRP we actually reduced the compatibility to only one arc for T1 and T2 and we created dummy full or empty requests for T3 and T4. However we have not considered any shortcut in the code. Thus this type of input may lead to some inefficiencies in the search with respect to specialized algorithms. We carried out these tests fixing to 20000 the number of iterations. In Table 11 we report the original absolute values of Bodin et al. (2000) and compare them with the more recent results of Wy and Kim (2013) and Derigs et al. (2013) by reporting the gaps of the two objective function components, and finally in the last two columns we report the gaps of HNS. Note that, in accordance with Bodin et al. (2000), we report only the deadhead time (DH), i.e., the total duration minus the service time.

**Table 11 Comparisons on the Rollon-Rolloff benchmark instances**

| Instances | | Bodin | | Wy (gaps) | | Derigs (gaps) | | HNS (gaps) | |
|---|---|---|---|---|---|---|---|---|---|
| | | $z$ | DH | $z$ | DH | $z$ | DH | $z$ | DH |
| 50 | A | 10 | 337 | 0 | -1.19% | 0 | -1.78% | 0 | -0.30% |
| | B | 10 | 238 | 1 | 0.00% | 1 | -0.84% | 1 | 2.52% |
| | C | 9 | 159 | 0 | 0.00% | 0 | 0.00% | 0 | 0.00% |
| | D | 7 | 194 | 0 | -1.55% | 0 | -1.55% | 0 | -1.55% |
| 75 | A | 16 | 491 | 0 | -1.02% | 1 | -1.63% | 0 | -0.20% |
| | B | 15 | 374 | 0 | -1.60% | 0 | -2.14% | 0 | 0.00% |
| | C | 13 | 211 | -1 | 0.47% | -1 | 0.47% | -1 | 4.27% |
| | D | 11 | 256 | 0 | -1.56% | 0 | -1.56% | 0 | -1.56% |
| 100 | A | 20 | 682 | 0 | -2.20% | 0 | -2.79% | 0 | -1.47% |
| | B | 19 | 511 | 0 | -1.76% | 0 | -1.57% | 0 | 0.59% |
| | C | 17 | 289 | 0 | 0.00% | -1 | 2.08% | 0 | 1.04% |
| | D | 14 | 323 | 0 | -0.93% | 0 | -0.93% | 0 | 0.93% |
| 150 | A | 29 | 800 | 0 | -1.00% | 1 | -2.63% | 0 | 1.63% |
| | B | 28 | 714 | 0 | -0.14% | 0 | -1.26% | 0 | 2.10% |
| | C | 26 | 448 | 1 | -2.90% | 0 | -0.45% | 0 | 0.22% |
| | D | 21 | 423 | 0 | -6.15% | 0 | -6.15% | 0 | -3.55% |
| 199 | A | 38 | 1067 | 0 | -1.78% | 0 | -3.94% | 0 | 0.66% |
| | B | 37 | 897 | 1 | -2.79% | 1 | -3.23% | 0 | 2.45% |
| | C | 34 | 553 | 0 | 0.00% | 0 | 0.00% | 0 | 0.00% |
| | D | 27 | 511 | 0 | -3.52% | 0 | -3.52% | 0 | -1.57% |
| Average gaps | | | | | -1.48% | | -1.67% | | 0.31% |

It should be noted that HNS compares reasonably well with the specialized algorithms: in most of the cases the number of vehicles is equal to the best solution. As for the total duration, which however is not the primary objective in the hierarchical approach of HNS, the gap with the other algorithms is quite low.

Finally, we performed a test motivated by our experience with the Gesenu company, where the current practice was to solve the problem as a sequence of petal patterns. On the contrary, managing containers as a shared commodity can lead to a substantial improvement, and we exploited this feature in our approach, reformulating each request T1 and T2 in terms of elementary requests T3 and T4.

**Table 12 Evaluating the impact of managing containers as a shared commodity using HNS**

| | A | | B | | C | | D | |
|---|---|---|---|---|---|---|---|---|
| | $z$ | DH | $z$ | DH | $z$ | DH | $z$ | DH |
| 50 | -1 | -41.99% | 0 | -34.75% | 0 | -21.38% | 0 | -9.95% |
| 75 | 0 | -38.92% | -1 | -33.88% | 0 | -28.91% | 0 | -14.29% |
| 100 | -1 | -39.82% | -1 | -34.06% | 0 | -32.87% | 0 | -12.81% |
| 150 | -1 | -38.77% | -1 | -35.18% | 0 | -21.61% | 0 | -14.86% |
| 199 | -1 | -47.32% | -1 | -37.90% | -1 | -23.51% | 0 | -17.04% |

Table 12 reports the results of our test showing the improvements in terms of the number of vehicles and the percentage of the deadhead with respect to the best results reported in Bodin et al. (2000), Derigs et al. (2013) and Wy and Kim (2013). The results show a significant reduction of the deadhead time. In particular, the instances with higher percentage of trips T1 and T2 (instances A and B) mostly benefit from managing containers as a shared commodity. Indeed, for all but one case in both sets A and B, the deadhead reduction allows to save one vehicle.

## 7. Conclusions

Recycling is the compulsory final step of most products life cycle and an essential part of integrated waste management. Bulky items are part of this process through a network of collection points (RDSs) established all over the territory, where large containers devoted to different materials are hosted. The Municipal Agency is in charge of the material final trip from an RDS to an MRF. A fleet of dedicated vehicles is devoted to bring full containers to MRFs to be emptied, and to bring them back to an RDS. To take advantage of economies of scale, Agencies tend to manage all the RDSs in the same region as a whole, yielding large scale vehicle routing problems which can not be efficiently solved by state of the art MILP solvers.

We addressed a real life problem in this class, formalized it as a special VRP on a bipartite graph, analyzed its structure, compared it to similar problems in the wider class of Rollon-Rolloff vehicle routing problems, emphasized those issues that most affect the problem hardness, and discussed for the first time the impact of limited additional containers. In particular, we studied a hierarchical bi-objective (fleet size and total distance), distance-constrained VRP, provided a specialized variant of the Clarke and Wright constructive procedure able to manage limited additional containers, and proposed a neighborhood based metaheuristic which exploits the problem structure, alternatively switches from one objective to the other along the search path, and periodically destroys and rebuilds parts of the solution.

The main algorithm components are experimentally evaluated on real and realistic instances, the largest of which fail to be solved by a MILP solver. We are increasingly competitive with the solver as the instance size increases, especially regarding fleet size. On the larger instances, the solution quality is addressed by comparing with a lower bound. Providing tight bounds to the fleet size objective is a challenging task due to the distance constraint, which makes even the feasibility check of a partition a hard optimization problem.

Our approach was also tested on instances from the literature against sophisticated algorithms previously proposed for a slightly different problem, i.e., the Rollon-Rolloff VRP where spare containers are not limited and are stationed at the MRF. We proved competitive on solution quality although our approach is not tailored for that case. Moreover, we experimentally showed the potential improvement that can be gained by treating containers as a general commodity that can be exchanged from site to site.

Computational results provided experimental evidence to our claim that this class of challenging real problems can be efficiently tackled by specialized though simple ad hoc algorithms, able to deal with the global constraint related to limited additional containers, allowing Municipal Agencies to optimally deal with a demanding operational task which has to be solved on a weekly base. In particular, decomposing each service request concerning a full container to be emptied into the removal of the full container and the delivery of an empty one, provides additional chances for optimization, specially in case of an integrated management of several RDSs, yielding large scale problems that our solution approach is able to handle.

## Acknowledgments

The authors wish to thank the Gesenu S.p.A. for fruitful collaboration and for providing us the data set and allowing their use in this paper. Besides, the authors wish to thank Matteo Boccafoli and Mattia Lorandi for their contribution in the experiments and helpful discussions. A grateful thanks to Roberto Baldacci for providing the literature instances. Finally, the authors would like to thank also the anonymous referees for their valuable comments that helped in improving the paper.

## References

Aarts, E., J.K. Lenstra, eds. 1997. *Local Search in Combinatorial Optimization*. John Wiley and Sons, Chichester, UK.

Archetti, C., M.G. Speranza. 2004. Vehicle routing in the 1-skip collection problem. *Journal of the Operational Research Society* **55**(7) 717–727.

Aringhieri, R., M. Bruglieri, F. Ciarletti, F. Malucelli, M. Nonato, M. Pera, R. Sorrentino. 2004a. Optimisation of the collection and disposal procedures for waste materials from collection centers in Umbria: Modelling and solving a real life waste collection problem. *Proceedings of ISWA World Congress*.

Aringhieri, R., M. Bruglieri, F. Malucelli, M. Nonato. 2004b. An asymmetric vehicle routing problem arising in the collection and disposal of special waste. L. Liberti, F. Maffioli, eds., *Proceedings of CTW 2004*, *Electronic Notes in Discrete Mathematics*, vol. 17. 41–47.

Aringhieri, R., M. Bruglieri, F. Malucelli, M. Nonato. 2008. Optimization of the collection and disposal of recyclable waste. *24 Hours Operations Research*. Http://www.24hor.org/.

Aringhieri, R., M. Dell'Amico. 2005. Comparing metaheuristic algorithms for the SONET network design problems. *Journal of Heuristics* **11**(1) 35–57.

Baldacci, R., L. Bodin, A. Mingozzi. 2006. The multiple disposal facilities and multiple inventory locations rollon-rolloff vehicle routing problem. *Computers & Operations Research* **33**(9) 2667–2702.

Beliën, J., L. De Boeck, J. Van Ackere. 2013. Municipal solid waste collection and management problems: A literature review. *Transportation Science* **48**(1) 78–102.

Bodin, L., A. Mingozzi, R. Baldacci, M. Ball. 2000. The rollon–rolloff vehicle routing problem. *Transportation Science* **34**(3) 271–288.

De Muelemeester, L., G. Laporte, F.V. Louveaux, F. Semet. 1997. Optimal sequencing of skip collections and deliveries. *Journal of the Operational Research Society* **48** 57–64.

Derigs, U., M. Pullmann, U. Vogel. 2013. A short note on applying a simple ls/lns-based metaheuristic to the rollonrolloff vehicle routing problem. *Computers and Operations Research* **40**(3) 867–872.

Elbek, M., S. Wøhlk. 2016. A variable neighborhood search for the multi-period collection of recyclable materials. *European Journal of Operational Research* **249**(2) 540–550.

Gendreau, M., A. Hertz, G. Laporte. 1994. A tabu search heuristic for the vehicle routing problem. *Management Science* **40**(10) 1276–1290.

Hauge, K., J. Larsen, R. M. Lusby, E. Krapper. 2014. A hybrid column generation approach for an industrial waste collection routing problem. *Computers & Industrial Engineering* **71** 10–20.

Imai, A., E. Nishimura, J. Current. 2007. A lagrangian relaxation-based heuristic for the vehicle routing with full container load. *European journal of operational research* **176**(1) 87–105.

Jula, H., M. Dessouky, P. Ioannou, A. Chassiakos. 2005. Container movement by trucks in metropolitan networks: modeling and optimization. *Transportation Research Part E: Logistics and Transportation Review* **41**(3) 235–259.

Laporte, G., M. Desrochers, Y. Norbert. 1984. Two exact algorithms for the distance-constrained vehicle routing problem. *Networks* **14** 161–172.

Laporte, G., M. Gendreau, J.Y. Potvin, F. Semet. 2000. Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research* **7** 285–300.

le Blanc, I., M. van Krieken, H. Krikke, H. Fleuren. 2006. Vehicle routing concepts in the closed-loop container network of arna case study. *OR Spectrum* **28**(1) 53–71.

Li, C., D. Simchi-Levi, M. Desrochers. 1992. On the distance constrained vehicle routing problem. *Operations Research* **40**(4) 790–799.

Martello, S., P. Toth. 1990. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley-Intersci. Ser. Discrete Math. Optim., John Wiley and Sons.

Nagarajan, V., R. Ravi. 2012. Approximation algorithms for distance constrained vehicle routing problems. *Networks* **59**(2) 209–214.

Toth, P., D. Vigo, eds. 2002. *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications, SIAM, Philadelpia, PA.

Vigo, D. 1996. A heuristic algorithm for the asymmetric capacitate vehicle routing problem. *European Journal of Operational Research* **89** 108–126.

Wy, J., B. Kim. 2013. A hybrid metaheuristic approach for the rollonrolloff vehicle routing problem. *Computers and Operations Research* **40**(8) 1947–1952.

Wy, J., B. Kim, S. Kim. 2013. The rollon-rolloff waste collection vehicle routing problem with time windows. *European Journal of Operational Research* **224**(3) 466–476.