CIRRELT
Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Heuristic Search for the Routing of Heterogeneous Trucks with Single and Double Container Loads

Michela Lai
Teodor Gabriel Crainic
Massimo Di Francesco
Paola Zuddas

June 2013

CIRRELT-2013-36

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Université Laval UQÀM Université du Québec à Montréal HEC MONTRĒAL École Polytechnique Montréal Université de Montréal

Heuristic Search for the Routing of Heterogeneous Trucks with Single and Double Container Loads

Michela Lai[1], Teodor Gabriel Crainic[2,*], Massimo Di Francesco[1], Paola Zuddas[1]

[1] School of Mathematics and Computer Science, University of Cagliari, Campus Aresu – Via San Giorgio 12/2, 09124 Cagliari, Italy

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Management and Technology, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal, Canada H3C 3P8

Abstract. This paper addresses a new vehicle and container routing problem variant, where container loads must be shipped from port to importers and from exporters to the port by truck carrying one or two containers, and trucks and containers must not be separated during customer service. We describe the problem and formulate an optimization model. A metaheuristic is proposed to address the model. It determines the initial solution by a variant of the Clarke-and-Wright algorithm and improves it by a sequence of local search phases. The comparison of the performance of the solutions yielded by the metaheuristic and that of a carrier's decisions show the validity and interest of the proposed method.

Keywords. Intermodal transportation, vehicle routing problem with backhauls, split vehicle routing problem, container drayage, metaheuristics.

Acknowledgements. This work was supported by Grendi Trasporti Marittimi. The authors are also grateful to Antonio Musso, who introduced to them the case-study that motivated this research. Partial support has also been provided by the Canadian Natural Sciences and Engineering Council (NSERC), through its Industrial Discovery Grants program.

Results and views expressed in this publication are the sole responsibility of the authors and do not necessarily reflect those of CIRRELT.

Les résultats et opinions contenus dans cette publication ne reflètent pas nécessairement la position du CIRRELT et n'engagent pas sa responsabilité.

* Corresponding author: Teodor-Gabriel.Crainic@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2013

1 Introduction

This paper addresses a vehicle and container routing problem motivated by a real case study, reflecting a problem class frequently encountered by carriers performing drayage operations around ports without leaving containers at customer locations or inland terminals. A carrier must plan the distribution of container loads by trucks and containers based in a port. Some trucks can carry one container, whilst the transportation capacity of other trucks is two containers. Two classes of customers must be serviced: the importers, who need to receive full container loads from the port, and the exporters, who must ship container loads to the port. Typically customers must be serviced by more than one container and must be visited more than once.

According to the carrier's policy, trucks and containers cannot be uncoupled during customer service. It is important to note that, consequently, there are not pickups or deliveries of loaded and empty containers in this problem: during customer service containers are filled or emptied and moved away by the same trucks used for moving them to and out of customer locations. This policy follows from the carrier business volumes, which do not make financially feasible the use of inland depots, nor the acquisition of a large container fleet. From the customers' point of view, this results in high-quality service, because drivers stay with the containers during unloading/unpacking and packing/loading operations and may promptly inform the carrier on possible problems during the service. Moreover, since the container loads of exporters are typically not ready before the afternoon, the carrier policy is to service importers before exporters. As a result, empty containers leaving from importers can be moved to exporters, where they are filled and shipped to the port.

According to Parragh et al. (2008), this problem belongs to the class of Vehicle Routing Problems with Clustered Backhauls (VRPCB), because in each route all deliveries must be performed before all pickups. However, each customer must be visited only once in classical VRPCB with backhauls, whereas multiple visits at each customer are allowed in the problem we introduce. There are also some similarities to drayage problems, which consist of picking up and delivering full containers from and at ports. Yet , again there are significant differences with exiting literature, as trucks and containers can be separated in typical drayage operations (Jula et al., 2005; Zhang et al., 2009), this opportunity is not available in the problem we introduce.

The objective of this paper is threefold. First, to introduce and describe a new variant of the VRP with clustered backhauls. Second, to propose an optimization model accounting for the original characteristics of this problem. The model minimizes distribution costs, such that all customers are serviced as requested, truck capacity constraints hold and importers are serviced before exporters. Third, to propose an efficient metaheuristic for this formulation allowing to address observed realistic cases within the time frame available to carriers.

The scheme of the metaheuristic is to find an initial feasible solution and, then, to improve this solution by several local search phases. The initial feasible solution is determined by a variant of the Clarke-and-Wright method, in which both routes and truck assignments are changed during the merging. In the improvement phases, we consider the search space of feasible routes and two different neighborhoods. In the first neighborhood, a node is moved from its route to another route with possible swaps in the assignments of trucks to routes. In the second neighborhood, two nodes in two different routes are swapped with possible swaps in the assignments of trucks to routes. If the search in a neighborhood improves the solution, the search is carried out again in the first neighborhood and, next, in the second one.

The significant contributions of this paper therefore are to:

- Introduce an original vehicle and container problem, which not only integrates several attributes, such as backhauls, load splits into multiple visits and heterogeneous trucks, but also accounts for characteristics not yet addressed in the literature but required in actual applications;
- Model this problem through a linear integer programming formulation;
- Propose an efficient solution method for this model. The method is a new metaheuristic based on several local search phases, in which both node movements and truck swaps are implemented.

The paper is organized as follows. In Section 2 a brief survey of related literature is provided. The problem description is presented in Section 3. The problem is modeled in Section 4. The metaheuristic is proposed in Section 5. The algorithm is tested in Section 6. Section 7 presents a summary of conclusions and describes future research perspectives.

2 Related literature

This problem belongs to the field of pickup and delivery problems, because there are two types of customers needing to ship or receive container loads (Savelsbergh and Sol, 1995). According to Parragh et al. (2008), the problem can be referred to as a Vehicle Routing Problem with Backhauls, because container loads must be shipped from the port to importers (or linehaul customers) and from exporters (or backhaul customers) to the port. More precisely, we investigate a Vehicle Routing Problem with Clustered Backhauls (VRPCB), because in each route all importers must be serviced before all exporters. In Berbeglia et al. (2007) the problem is referred to as a one-to-many-to-one pick up and delivery problem.

In traditional VRPCB, however, each customer must be visited exactly once, whereas in our problem the demand of each customer may be greater than the vehicle capacity and may be serviced by several vehicles. Therefore, multiple visits are allowed (Archetti and Speranza, 2008). To our knowledge, Vehicle Routing Problems with Backhauls and splits have been investigated only in Mitra (2005) and Mitra (2008). Unlike in our problem, the fleet of trucks is homogeneous and the delivery and pickup of loads can be in any sequence.

In the field of freight transportation, container transportation by truck between customers and intermodal terminals is known as "drayage". According to Macharis and Bontekoning (2004), it involves the delivery of a full container from an intermodal terminal to a receiver and the following collection of an empty container, as well as the provision of an empty container to the shipper and the subsequent transportation of a full trailer or container to the intermodal terminal.

According to the previous definition of drayage, it is possible to separate or not trucks and containers. The separation of trucks and containers has been investigated often (Jula et al., 2005; Chung et al., 2007; Zhang et al., 2009, 2010; Vidovic et al., 2011). Less attention has been devoted to the case in which the tractor and truck drivers stay with containers during packing or unpacking operations. Due to the lower productivity in this use of trucks, this case has been investigated only in papers motivated by specific technical restrictions (Imai et al., 2007) or particular regulatory policies (Cheung et al., 2008). In this paper, trucks and containers cannot be uncoupled because the carrier aims at providing a higher quality service, in which truck drivers are involved in the control of packing and unpacking operations.

To our knowledge, most of the papers on drayage assume that vehicles transport exactly one container at a time (Jula et al., 2005; Namboothiri and Erera, 2008; Zhang et al., 2009, 2010). However, if a given weight limitation is not exceeded or special combined chassis are used, carrying two containers per truck is allowed in many countries (Nagl, 2007). Hence, nowadays, moving two containers instead of one is an opportunity to improve the efficiency of the distribution. It is important to note that this opportunity makes routing problems more difficult to solve due to their combinatorial characteristics: a truck with a single container moves one loaded or one empty container, whereas a truck with two containers moves two loaded containers or two empty containers or a loaded container and an empty container. The possibility of moving two containers per truck was considered only in Chung et al. (2007), who worked on an one-to-one pick up and delivery problem, and Vidovic et al. (2011), who determined routes from the matching of pick up and delivery nodes.

The closest problem setting was probably faced by Imai et al. Imai et al. (2007), who studied the optimal assignment of trucks to a set of delivery and pickup pairs. As in our setting, tractors and containers cannot be uncoupled, but the capacity of trucks is limited to one container only. Caris and Janssens (2009) extended the paper by Imai et al. (2007) with time-windows at customers and depot. Their problem was modelled as a full truckload pickup and delivery problem with time windows. Even this approach is viable only if the transportation capacity of trucks is one container.

To conclude, one observers several gaps in the knowledge and literature. This paper makes a significant contribution by addressing the challenge of starting to fill these gaps.

3 Problem statement

Consider a fleet of trucks and containers based at a port. Some trucks carry one container, whilst other trucks can carry up to two containers. Trucks and containers are used to service two types of transportation requests: the delivery of container loads from the port to importers and the shipment of container loads from exporters to the same port. Typically customers need to ship or receive more than one container load. Therefore, usually each customer must be serviced by multiple containers and must be visited by more than one truck.

A relevant characteristic of this problem is the impossibility to separate trucks and containers. As a result, when importers receive container loads by trucks, containers are emptied and moved away by the same trucks used for providing container loads. Similarly, when exporters are served by empty containers, containers are filled and moved away by the same trucks used for providing empty containers.

According to the carrier's policy, importers must be serviced before exporters. As a result, routes may consist in the shipment of container loads from the port to importers, the allocation of empty containers from importers to exporters, and the final shipment of container loads from exporters to the port. Hence, trucks with one container can service up to two customers in a route (one importer and one exporter). Trucks with two containers can service up to four customers in a route (two importers and two exporters). Every pair of containers can be shipped in a truck, all containers leaving from importers can be used to service exporters and no incompatibility occurs between customers and trucks, which can service almost any customer.

It is important to note that the number of container loads to be picked up and delivered is generally different. When the number of container loads delivered to importers is larger than the number of container loads shipped by exporters, a number of empty containers must be moved back to the port. When the number of container loads delivered to importers is lower than the number of container loads shipped by exporters, a number of empty containers must be put on trucks leaving from the port, in order to service all customers.

The movement of trucks generate routing costs. In this problem, trucks with two containers lead to higher costs per unitary distance than trucks with one container. Moreover, handling costs are paid to put containers on trucks at the port. The objective is to determine the routes of trucks in order to minimize routing and handling costs, such that customers are serviced as requested, truck capacity constraints hold, and importers are serviced before exporters.

4 Modeling

We consider a port p, a set I of importers, a set E of exporters, and a set K of different trucks, each with capacity u_k. An integer demand of $d_i \geq 0$ containers is associated with each customer $i \in I \cup E$. When $i \in I$, d_i represents the number of loaded containers used to service import customer $i \in I$. Due to the problem setting, d_i is also equal to the number of empty containers available after the service to customer $i \in I$. When $i \in E$, d_i represents the number of empty containers used to service export customer $i \in E$. In this problem setting, d_i is also equal to the number of loaded containers shipped by customer $i \in E$ to port p .

Consider a direct graph $G = (N, A)$, where $N = \{p \cup I \cup E\}$ and the set of arcs A includes all allowed ways to move trucks: from the port to any importer and any exporter; from an importer to the port, any other importer and any exporter; and from an exporter to the port and any other exporter. More formally, the set A is defined as $A = A_1 \cup A_2$, where

$$A_1 = \{(i, j) | i \in p \cup I, j \in N, i \neq j\}$$
$$A_2 = \{(i, j) | i \in E, j \in p \cup E, i \neq j\}.$$

The routing cost c^k_{ij} of truck $k \in K$ on arc $(i, j) \in A$ is supposed to be nonnegative. Moreover, let h^k_{pj} be the nonnegative handling cost of a container put on truck $k \in K$ at the port p to service customer $j \in N$.

The following decision variables are defined:

- x^k_{ij}: Routing selection variable equal to 1 if arc $(i, j) \in A$ is traversed by truck $k \in K$, 0 otherwise;
- y^k_{ij}: Integer variable representing the number of loaded containers moved along arc $(i, j) \in A$ by truck $k \in K$;
- z^k_{ij}: Integer variable representing the number of empty containers moved along arc $(i, j) \in A$ by truck $k \in K$.

The problem can be formulated as follows:

$$min \sum_{k\in K}\left[\sum_{(i,j)\in A} c_{ij}^k x_{ij}^k + \sum_{j\in N} h_{pj}^k (y_{pj}^k + z_{pj}^k)\right] \tag{1}$$

$s.t.$

$$\sum_{k\in K}\sum_{l\in N} y_{il}^k = \sum_{k\in K}\sum_{j\in p\cup I} y_{ji}^k - d_i \qquad \forall i \in I \tag{2}$$

$$\sum_{k\in K}\sum_{l\in N} z_{il}^k = \sum_{k\in K}\sum_{j\in p\cup I} z_{ji}^k + d_i \qquad \forall i \in I \tag{3}$$

$$\sum_{l\in N} y_{il}^k \leq \sum_{j\in p\cup I} y_{ji}^k \qquad \forall i \in I, \forall k \in K \tag{4}$$

$$\sum_{l\in N} z_{il}^k \geq \sum_{j\in p\cup I} z_{ji}^k \qquad \forall i \in I, \forall k \in K \tag{5}$$

$$\sum_{k\in K}\sum_{l\in p\cup E} y_{il}^k = \sum_{k\in K}\sum_{j\in N} y_{ji}^k + d_i \qquad \forall i \in E \tag{6}$$

$$\sum_{k\in K}\sum_{l\in p\cup E} z_{il}^k = \sum_{k\in K}\sum_{j\in N} z_{ji}^k - d_i \qquad \forall i \in E \tag{7}$$

$$\sum_{l\in p\cup E} y_{il}^k \geq \sum_{j\in N} y_{ji}^k \qquad \forall i \in E, \forall k \in K \tag{8}$$

$$\sum_{l\in p\cup E} z_{il}^k \leq \sum_{j\in N} z_{ji}^k \qquad \forall i \in E, \forall k \in K \tag{9}$$

$$\sum_{(ji)\in A} (y_{ji}^k + z_{ji}^k) = \sum_{(il)\in A} (y_{il}^k + z_{il}^k) \qquad \forall i \in I \cup E, \forall k \in K \tag{10}$$

$$y_{ij}^k + z_{ij}^k \leq u_k x_{ij}^k \qquad \forall (i,j) \in A, \forall k \in K \tag{11}$$

$$\sum_{j\in N} x_{ji}^k - \sum_{l\in N} x_{il}^k = 0 \qquad \forall i \in N, \forall k \in K \tag{12}$$

$$\sum_{j\in N} x_{pj}^k \leq 1 \qquad \forall k \in K \tag{13}$$

$$\sum_{k\in K}\sum_{i\in I\cup E} z_{ip}^k - \sum_{k\in K}\sum_{i\in I\cup E} z_{pi}^k = \sum_{i\in I} d_i - \sum_{i\in E} d_i \tag{14}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in A, \forall k \in K \tag{15}$$

$$y_{ij}^k \in Z^+ \cup 0 \qquad \forall (i,j) \in A, \forall k \in K \tag{16}$$

$$z_{ij}^k \in Z^+ \cup 0 \qquad \forall (i,j) \in A, \forall k \in K \tag{17}$$

Container loading at port p and truck operating costs are minimized in the objective function (1).

Constraints from (2) to (5) concern the movement of containers to importers. Constraints (2) and (3) are the flow conservation constraints of loaded and empty containers, respectively, at each importer node. Constraint (4) and (5) check the number of loaded and empty containers in each truck entering and leaving from importers: the number of loaded containers cannot be increased after a service at each importer, whereas the number of empty containers cannot be reduced.

Constraints from (6) to (9) concern the allocation of containers to exporters. Constraints (6) and (7) are the flow conservation constraints of loaded and empty containers, respectively, for each exporter node. Constraint (8) and (9) control the number of loaded and empty containers in each truck entering and leaving from exporters: the number of loaded containers cannot be reduced after a service at each exporter, whereas the number of empty containers cannot be increased.

Constraint (10) guarantees that the number of containers carried by each truck does not change after visiting a customer.

Constraint (11) imposes that the number of containers moved by each truck is not larger than its transportation capacity u_k. In this problem, u_k takes value 1 or 2. Constraints (12) are the flow conservation constraints for trucks at each node. Constraint (13) guarantees that trucks are not used more than once. Constraint (14) represents the flow conservation of empty containers at port p.

Finally, constraints (15), (16) and (17) define the domain of decision variables.

5 Solution method

Vehicle Routing Problems are known to be difficult and our problem is not an exception. Since exact methods may not be able to solve real problem instances, we propose a metaheuristic, in which solutions are defined in terms of truck routes. The metaheuristic consists of two phases: in the first phase, also called *First Phase*, a feasible solution is determined by a variant of the Clarke and Wright method, in which routes are merged and assigned to trucks; in the second phase, also called *Improvement Phase*, this solution is improved by several local search phases. The search space of the local search phases is the set of truck assignments to routes satisfying all constraints (2)...(17). Two neighborhoods are used: in the first, a node is moved from its current route and inserted into another route by the best-insertion method and trucks are reassigned to routes involved in this local move. The search in this neighborhood is called *Node Move* search phase. In the second, two nodes are swapped between two different routes and trucks are reassigned to routes involved in this local move. The search in this neighborhood is called *Node Exchange* search phase.

In both the *First Phase* and the *Improvement Phase* the metaheuristic determines new solutions by evaluating pairs of routes in the current solution, selecting one of them and assigning trucks to the selected pair of routes. In the *First Phase*, the metaheuristic selects which pair of routes should be merged and which truck should be used in the new route. In the *Improvement Phase*, the metaheuristic evaluates how nodes are moved and swapped between any pair of routes and selects the pair of routes involved in the local movement. The trucks assigned to the routes of the selected pair can be swapped, if the solution improves.

It is important to note that there is not a single way to compare the pairs of routes. Most importantly, the different comparison criteria may result in the selection of different pair of routes at any step. In this paper, two criteria are proposed for the selection of the pairs of routes: the first criterion, which is called $Criterion1$, selects the pairs of routes, such that the objective function in the new solution is minimum; the second criterion, which is called $Criterion2$, selects the pair of routes, such that the total traveled distance in the new solution is minimum. Since one does not know a priori which criterion finally determines the best solution in terms of objective function, both of them are considered in the *First Phase* and the *Improvement Phase*.

The pseudo-code of the metaheuristic algorithm is illustrated in Table 1, where:

initSol represents the initial solution;

initList is the list of savings determined by merging routes in the initial solution;

cwSol(Criterion) represents the current solution in the *First Phase* for the considered criterion;

cwList(Criterion) is the list of savings derived from the current solution $cwSol(Criterion)$ in the *First Phase* for the considered criterion;

ClarkeWright($cwSol(Criterion), cwList(Criterion)$**)** is the function implementing the variant of the Clarke and Wright method in the *First Phase*. The input parameters are the current solution $cwSol(Criterion)$ and the current list of savings $cwList(Criterion)$. The output is the new solution $cwSol(Criterion)$;

Sol(Criterion) is the current solution in the *Improvement Phase* for the considered criterion;

improve is a boolean variable taking value true if the *Improvement Phase* improves the solution, false otherwise. It is initialized to true, in order to perform the *Improvement Phase* at least once.

improveNodeMove is a boolean variable taking value true if the *Node Move* search phase improves the current solution in the *Improvement Phase*, false otherwise. It is initialized to false.

improveNodeExchange is a boolean variable taking value true if the *Node Exchange* search phase improves the current solution in the *Improvement Phase*, false otherwise. It is initialized to false.

nodeMove($Sol(Criterion), improveNodeMove$**)** is the function implementing the *Node Move* search phase. The input parameters are the current solution $Sol(Criterion)$ and the variable $improveNodeMove$. It returns the new current solution $Sol(Criterion)$ and the updated value of $improveNodeMove$.

nodeExchange($Sol(Criterion), improveNodeExchange$**)** is the function implementing the *Node Exchange* search phase. The input parameters are the current solution $Sol(Criterion)$ and the variable $improveNodeExchange$.
It returns the new current solution $Sol(Criterion)$ and the updated value of $improveNodeExchange$.

s* is the best solution found.

5.1 Initialization

The initial solution *cwSol* is made up with direct trips, in which vehicles carrying one container start from the port, go directly to a customer and then turn back directly to the port. Although in this solution all customers are served as requested, it is unfeasible when the number of trucks with one container is not sufficient.

In order to implement our variant of the Clarke and Wright method (see Section 5.2), the list *initList* is initialized. Each entry of *initList* represents:

- The identifiers of the pair of routes which may be merged;
- The maximum saving deriving from their merging. To clarify, since a pair of routes can sometimes be merged in several ways, we consider only the new route resulting in the maximum saving;
- The minimum capacity $\check{v}$ of the new route generated after the merging. In this specific problem, if the new route consists of 1 importer and 1 exporter demanding 1 container each, the minimum capacity is 1 container, whereas it is 2 containers otherwise.

Given two generic routes r_1 and r_2, the maximum saving generated by their merging is computed as:

$$cost(r_1) + cost(r_2) - cost(r)$$

```
procedure MAIN
    Initialize                                                          ▷ See section 5.1
    Create $initSol$ with direct trips performed by trucks carrying one container only
    Create the savings list $initList$
    EndInitialize
    for Criterion = 1, 2 do
        $cwSol(Criterion) \leftarrow initSol$
        $cwList(Criterion) \leftarrow initList$
        ClarkeWright($cwSol(Criterion), cwList(Criterion)$)             ▷ See section 5.2
        $improve \leftarrow true$
        $Sol(Criterion) \leftarrow cwSol(Criterion)$
        while $improve == true$ do                                      ▷ See section 5.3
            $improve \leftarrow false$
            $improveNodeMove \leftarrow false$
            $improveNodeExchange \leftarrow false$
            nodeMove($Sol(Criterion), improveNodeMove$)
            nodeExchange($Sol(Criterion), improveNodeExchange$)
            if $improveNodeMove == true$ OR $improveNodeExchange == true$ then
                $improve \leftarrow true$
            end if
        end while
    end for
    $s* \leftarrow \emptyset$
    if $Sol(1) <= Sol(2)$ then
        $s* \leftarrow Sol(1)$
    else
        $s* \leftarrow Sol(2)$
    end if
    return $s*$
end procedure
```

Table 1: The structure of the Metaheuristic

where $cost(r_1)$ represents the cost of route r_1, $cost(r_2)$ the cost of route r_2 and $cost(r)$ the cost of the route obtained by merging r_1 and r_2. Savings are ordered in the *initList* in a non-increasing fashion.

5.2 First phase

In order to reduce the number of routes to the number of available trucks, our variant of the Clarke and Wright method is implemented as follows:

$Step_0$ The current solution $cwSol(Criterion)$ and the associated saving list $cwList(Criterion)$ are considered;

Step$_1$ If the number of routes is lower than the number of trucks or if there are no entries in the *cwList*(*Criterion*), save the current solution *cwSol*(*Criterion*) and stop, otherwise go to *Step*$_2$;

Step$_2$ Consider the pair of routes at the top of the list *cwList*(*Criterion*) and check its minimum capacity $\check{v}$. If there is an available truck having capacity $\check{v}$, merge the pair of routes, assign the truck to the new route and go to *Step*$_5$, otherwise go to *Step*$_3$;

Step$_3$ If a route of the pair has capacity $\check{v}$, merge the pair of routes, assign the truck to the new route and go to *Step*$_5$, otherwise go to *Step*$_4$;

Step$_4$ Scroll the list and check the next entry of *cwList*(*Criterion*). If *cwList*(*Criterion*) is not empty go to *Step*$_2$, otherwise return the new current solution *cwSol*(*Criterion*) and stop;

Step$_5$ Save the new current solution *cwSol*(*Criterion*), re-compute the saving list *cwList*(*Criterion*) and repeat from *Step*$_1$.

After the previous steps, the solution may not be feasible, because the final number of routes was not controlled. If the number of routes is lower than or equal to the number of trucks, the current solution is feasible and the metaheuristic switches to the *Improvement Phase* (section 5.3). If this is not the case, the algorithm turns all routes performed by one-container trucks into direct trips. The new current solution *cwSol*(*Criterion*) is now made up of all routes with two-container trucks as in the previous solution and direct trips performed by one-container trucks. The *cwList* is re-computed and the previous steps of the Clarke and Wright's method are repeated without including one-container trucks in *Step*$_3$, in order to not cycle back to the previous solution. After this phase, the solution is feasible and the metaheuristic proceeds with the *Improvement Phase*.

5.3 Improvement Phase

The *Improvement Phase* consists of several local search phases using two neighborhoods. The first neighborhood is the set of feasible solutions obtained by moving a node from a route to another route by the best insertion method, and reassigning the trucks to routes involved in the local move. To clarify, when a node is moved and truck assignments are not changed, solutions may be unfeasible, but sometimes feasible solutions can be obtained by swapping the trucks assigned to the routes involved in the move. The second neighborhood is the set of feasible solutions obtained by exchanging two nodes between two routes and by reassigning the trucks to routes. As in the first neighborhood, the trucks assigned to routes can be swapped, in order to determine the feasible solutions.

The *Improvement Phase* starts from the feasible solution determined in the *First Phase* and performs the *Node Move* search phase. It starts from the first route, which is in the top of the list of solution routes. If there are improving moves for the first route, the best one is selected and implemented. The routes involved in this move are inserted into the bottom of the list of routes in the new solution, because the list of routes is scrolled down from the top and local moves should not be limited to a subset of routes. After an improving move, the routes list is updated and the search restarts from the top of the list. If there are not improving moves for the first route, the best not improving move is saved and the search goes on with the following route.

After a series of local moves, we encounter a local optimum, where improving moves are no longer available. The best not-improving move is implemented, the route list is updated and the search restarts from the top of the list. In order to not repeat moves during the search, moves already implemented are disallowed and saved in two separate memories: one for the *improving moves* and one for the *not improving moves.* To not disallow possible moves for too long, *improving move*are deleted from their memory whenever a *not improving move* is implemented, whereas *not improving move* are deleted when a better local optimum is determined.

After the consecutive implementation of the maximum number of not improving moves, the search in the first neighborhood stops and is carried out in the second neighborhood. If the search in a neighborhood improves the solution, the search is carried out again in the first neighborhood and, next, in the second one.

The template of the local search in a neighborhood is:

$Step_0$ $notImpIt = 0$;

$Step_1$ If *notImpIt* is equal to *maxnotImpIt* , the method stops its execution. Otherwise increase $notImpIt$ by 1 e go to next step;

$Step_2$ Search the best *improving move*, if any, and go to next step, otherwise go to $Step_4$ (during this search, the best *not improving move* is saved);

$Step_3$ Implement the best *improving move* and go to $Step_2$;

$Step_4$ If the solution has been improved, save the new local optimum $Sol(Criterion)$ and set $notImpIt = 0$. Go to $Step_5$;

$Step_5$ Select the best *not improving move*, if any, implement it, and go to $Step_1$. Otherwise the method stops its execution.

where

notImpIt is the number of iterations for which a not improving move has been consecutively implemented. When a better local optimum is found, *notimpIt* is set to 0; when a not improving move is implemented, *notimpIt* is increased by 1.

maxnotImpIt is the maximum number allowed of iterations for not improving moves.

6 Experimentation

The objective of the experimentation is to analyze the performance of the proposed solution method and to verify that problems of real-life dimensions are solvable in reasonable time. The solution method is tested on five real instances provided by a carrier operating in the area of Genoa (Italy), as well as on several randomly generated problem instances, that display structures closely similar to real problems.

We have generated 70 artificial instances, which are divided into five classes:

- 6 instances with 10 customers;
- 10 instances with 20 customers;
- 14 instances with 30 customers;
- 18 instances with 40 customers;
- 22 instances with 50 customers.

The number of containers requested by each customer is uniformly generated from 1 to 5. In each class the coordinates of nodes are fixed. The instances of a class differ in the number of importers and exporters, as well as in the number of each type of trucks. In each instance, we use the minimum number of trucks, such that all container loads are serviced. In half of the instances of each class, two-container trucks service two-thirds of container loads, whilst the remaining part is serviced by one-container trucks. The other half of instances of each class considers only two-container trucks.

The metaheuristic is implemented in the programming language C++. Tests are performed by Cplex 12.2 running on a four-CPU server 2.67 GHz 64 GB RAM. Although a major requirement for the carrier is to determine solutions in about 10 minutes, Cplex is set to stop after 3 hours, because we are interested in evaluating the quality of solutions provided by the metaheuristic.

Two parameters must be calibrated in the algorithm:

***maxnotImpIt* (*Node Move*)** , which is the maximum number of consecutive *not improving move*in the *Node Move* search phase;

***maxnotImpIt* (*Node Exchange*)** , which is the maximum number of consecutive *not improving move*in the *Node Exchange* search phase;

The calibration is performed by two coefficients α or β taken from Table 2. Parameter *maxnotImpIt*(*Node Move*) is computed as the product between the number of routes in the solution and α. Parameter *maxnotImpIt* (*Node Exchange*) is computed as the product between the number of routes in the solution and β. Eight calibrations, which are denoted by $C1 \ldots C8$ in Table 2, are performed by running all artificial instances with the associated values of α or β. For example, in the first calibration $C1$, both α and β have value 1. The selected values of α of β are shown in the last column of Table 2.

	Tested value of α and β								Selected
	$C1$	$C2$	$C3$	$C4$	$C5$	$C6$	$C7$	$C8$	
α	1	2	3	4	5	6	7	8	5
β	1	2	3	4	5	6	7	8	5

Table 2: Calibration Scheme

Computational results are indicated in Tables 3 and 4, where:

- $|I|$: Number of importers;
- $|E|$: Number of exporters;
- $|K1|$: Number of available trucks with capacity 1 container;
- $|K2|$: Number of available trucks with capacity 2 containers;

METAHEURISTIC

- *o.f.*: Objective function returned by the metaheuristic;
- *Node Move* : Total improvement obtained in the *Node Move* search phases;
- *Node Exchange* : Total improvement obtained in the *Node Exchange* search phases;
- *Best Criterion*: The criterion determining the best solution: 1 for *Criterion*1, 2 for *Criterion*2 and X if the criteria are equivalent.
- *Gap between criteria*: The difference between solutions returned by the two criteria;
- *t (s)*: Seconds to solve instances;

- *% Gap from CPLEX*: Percentage gap with respect to the best solution provided by Cplex;

  CPLEX

- *Optimality Gap*: The gap between upper and lower bounds determined by Cplex. If the optimal solution is found for a problem instance, we indicate how many seconds it takes for Cplex to solve that instance;
- *n.s.*: No feasible solution determined by Cplex within 3 hours.

Tables 3 and 4 show that only few instances with 10 customers can be optimally solved by Cplex within 10 minutes, as required by the carrier. Cplex does not provide feasible solutions within 3 hours for a few instances with 30 customers, many instances with 40 customers and all instances with 50 customers.

The metaheuristic can solve all instances in less than one minute. Generally speaking, the *Node Exchange* search phase improves the solution more often than the *Node Move*. Sometimes the best solution is returned by $Criterion1$, whereas at times $Criterion2$ is better. Moreover, sometimes the best solution is much better than the other one. Therefore, the intuition of considering both criteria in the algorithm seems to be appropriate.

Gaps between the best upper bounds provided by Cplex within 3 hours and the metaheuristic solutions are lower than 1% in the case of instances with 10 and 20 customers. When the solutions of the metaheuristic are better than Cplex upper bounds, gaps are reported with a negative value. As problem sizes increase, the metaheuristic provides significantly better solutions than Cplex.

Real instances, which have about 40 customers, cannot be solved by Cplex within 10 minutes. In this case the metaheuristic is compared to the performance of the carrier's decisions in terms of total traveled distances. Results are shown in Table 5, in which the following notation is used:

- *Instances* The instance considered;
- $|I|$ Number of importers;
- $|E|$ Number of exporters;
- $|K1|$ Number of available trucks carrying 1 container;
- $|K2|$ Number of available trucks carrying 2 containers;
- *Carrier Distances* The total traveled distance according to the carrier's decisions (Km);

- *Distances* The total traveled distance according to the metaheuristic (Km);
- *Saving(km)*: Difference between the metaheuristic and the carrier's decisions (Km);
- *Saving(%)*: Percentage gap between the metaheuristic and the carrier's decisions;

Table 5 shows that in each instance the metaheuristic improves the carrier's performance. The improvement seems to be particularly relevant when $|I|$ increases and becomes closer to $|E|$, due to the larger search space of feasible routes.

7 Conclusion

The distribution of containers between customers and ports is not a new issue, but it was mainly investigated from the point of view of large carriers, which operate inland networks with several depots, where containers can be temporarily kept in stock, and may leave containers at customer locations for unloading and loading operations. In order to maximize their productivity, containers are picked up or delivered and trucks bypasses packing and unpacking operations. The related literature reflects this fact as most papers separate trucks and containers during customer service.

This paper sheds light on a different policy adopted by medium-size carriers: the distribution of trucks carrying the same containers along their route. The paper shows that this policy gives rise to a challenging vehicle and container routing problem. It exhibits an original configuration of attributes, which has not been investigated heretofore in the literature. Hence, no existing approaches can be adopted.

The contributions of this paper to the literature therefore are to:

- Present an original vehicle and container problem, because it integrates clustered backhauls, load splits into multiple visits, and heterogeneous trucks with different transportation capacities, and because it accounts for characteristics not yet addressed in the literature but required in actual applications;
- Introduce a linear integer programming model for this problem;
- Define and implement an effective solution method for this model. The method is a new metaheuristic based on a sequence of local search phases. The most innovative part of the proposed algorithm consists in the particular local moves proposed, which integrate truck assignments. More precisely, while existing local searches evaluate node moves and swaps between pairs of routes to select the"best" pair of routes involved in the local move (e.g., Caris and Janssens, 2009), the proposed

algorithm swaps both nodes and trucks in the selected pairs of routes, if the solution improves.

A commercial solver was used to address a number of artificial instances, but was able to solve only those with few customers. The metaheuristic has been tested on both artificial and real instances. According to our tests, it is more effective than the commercial solver in solving artificial instances similar to real problem instances. Moreover, the comparison with the carrier's decisions shows that the metaheuristic represents a promising tool to improve its current decision-making processes, because it leads to significant savings and determines routes quickly.

Research is in progress to address problems with multiple trips for trucks, incompatibilities between container loads and trucks, as well as larger transportation capacities. New solution methods will be developed accounting for the specific characteristics of these problems.

Acknowledgments

This work was supported by Grendi Trasporti Marittimi. The authors are also grateful to Antonio Musso, who introduced to them the case-study that motivated this research. Partial support has also been provided by the Canadian Natural Sciences and Engineering Council (NSERC), through its Industrial Discovery Grants program.

References

C. Archetti and M. G. Speranza. The split delivery vehicle routing problem: A survey. In B. Golden, S. Raghavan, and E. Wasil, editors, *The vehicle routing problem: latest advances and new challenges*, pages 103–122. Springer Science+Business Media, New York, USA, 2008.

G. Berbeglia, J. F. Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems: a classification scheme and survey. *Top*, (15):1–31, 2007.

A. Caris and G. K. Janssens. A local search heuristic for the pre- and end-haulage of intermodal container terminals. *Computers & Operations Research*, (36):2763–2772, 2009.

R. K. Cheung, N. Shi, W. B. Powell, and H. P. Simao. An attributedecision model for cross-border drayage problem. *TransportationnResearch Part E: Logistics and Transportation Review*, (44):217–234, 2008.

K. H. Chung, C. S. Ko, J. Y. Shin, H. Hwang, and K. H. Kim. Development of mathematical models for the container road transportation in Korean trucking industries. *Computers & Industrial Engineering*, (53):252–262, 2007.

A. Imai, E. Nishimura, and J. Current. A lagrangian relaxation-based heuristic for the vehicle routing with full container load. *European Journal of Operational Research*, (176):87–105, 2007.

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

C. Macharis and Y. M. Bontekoning. Opportunities for or in intermodal freight transport research: A review. *European Journal of Operational Research*, (153):400–416, 2004.

S. Mitra. An algorithm for the generalized vehicle routing problem with backhauling. *Asia-Pacific Journal of Operational Research*, (22):153–169, 2005.

S. Mitra. A parallel clustering technique for the vehicle routing problem with split deliveries andpickups. *Journal of the Operational Research Society*, (59):1532–1546, 2008.

P. Nagl. Longer combination vehicles (lcv) for asia and pacific region: Some economic implications. *UNESCAP Working Papers, United Nations*, 2007.

R. Namboothiri and A. L. Erera. Planning local container drayage operations given a port access appointment system. *Transportation Research Part E: Logistics and Transportation Review*, (44):185–202, 2008.

S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems part i: Transportation between customers and depot. *Journal fr Betriebswirtschaft*, 58 (2):21–51, 2008.

M. W. P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, (29):17–29, 1995.

M. Vidovic, G. Radivojevic, and B. Rakovic. Vehicle routing in container pickup and delivery processes. *Procedia Social and Behavioral Sciences*, (20):335–343, 2011.

R. Zhang, W. Y. Yun, and I. K. Moon. A reactive tabu search algorithm for the multi-depot container truck transportation problem. *Transportation Research Part E: Logistics and Transportation Review*, (45):904–914, 2009.

R. Zhang, W. Y. Yun, and H. Kopfer. Heuristic-based truck scheduling for inland container transportation. *OR Spectrum*, (32):787–808, 2010.

				METAHEURISTIC							CPLEX
$\|I\|$	$\|E\|$	$\|K1\|$	$\|K2\|$	o.f	Node Move	Node Exchange	Best Criterion	Gap between criteria	t (s)	% Gap from CPLEX	% Optimality Gap
						10 CUSTOMERS					
2	8	2	9	20,135.28	426.54	0.00	2	44.48	0.00	0.00	0.00 (376.08 s)
5	5	2	7	20,381.79	247.52	54.57	1	14.40	0.00	0.19	0.00 (5.21 s)
8	2	5	9	21,072.44	0.00	271.45	1	90.22	0.00	0.80	0.00 (277.09 s)
2	8	0	10	20,677.64	0.00	248.63	1	5.63	0.00	0.03	3.76
5	5	0	8	19,960.83	0.00	136.49	X	0.00	0.00	0.00	0.00 (29.70 s)
8	2	0	12	20,697.67	131.29	232.18	X	0.00	0.00	0.00	3.83
						20 CUSTOMERS					
2	18	8	22	41,214.33	0.00	0.00	X	0.00	0.00	0.00	2.51
5	15	7	19	36,607.42	0.00	0.00	X	0.00	0.00	0.00	0.00 (5,845.77 s)
10	10	5	14	32,333.11	0.00	306.83	1	426.18	1.00	0.56	2.26
15	5	7	19	37,657.58	0.00	0.00	2	15.10	1.00	0.02	2.87
18	2	5	24	42,781.65	0.00	461.78	2	7.17	0.00	0.00	2.53
2	18	0	26	40,806.04	0.00	0.00	X	0.00	0.00	-0.91	5.74
5	15	0	23	36,048.28	0.00	0.00	X	0.00	0.00	0.00	2.99
10	10	0	17	31,691.35	0.00	144.39	2	8.11	0.00	0.06	3.91
15	5	0	23	37,301.67	0.00	0.00	X	0.00	1.00	-0.75	6.01
18	2	0	27	43,015.11	0.00	0.00	2	8.76	0.00	-0.04	4.96
						30 CUSTOMERS					
2	28	13	33	64,525.58	0.00	0.00	1	294.72	3.00	-5.06	7.54
5	25	12	30	60,416.51	0.00	0.00	1	52.44	8.00		n.s.
10	20	10	25	52,645.37	0.00	0.00	1	601.20	2.00	0.43	2.98
15	15	8	19	50,158.37	1,502.78	556.68	1	158.14	2.00	1.13	2.69
20	10	10	26	53,571.86	445.20	449.54	1	307.32	2.00	-6.38	9.08
25	5	12	32	62,111.73	0.00	0.00	1	127.90	1.00	-0.26	3.21
28	2	14	35	67,227.58	0.00	0.00	1	114.90	1.00		n.s.
2	28	0	40	62,782.75	0.00	0.00	X	0.00	1.00	-3.43	7.02
5	25	0	36	58,974.78	0.00	333.19	2	96.96	1.00	-7.96	11.43
10	20	0	30	51,763.93	0.00	436.31	2	932.64	2.00	-4.72	9.38
15	15	0	23	48,940.66	0.13	909.49	2	184.18	1.00	-0.85	6.61
20	10	0	31	53,095.51	322.75	565.86	1	279.02	1.00	-1.99	7.16
25	5	0	38	60,565.79	148.25	0.00	1	58.92	0.00	-3.71	7.07
28	2	0	42	65,305.66	0.00	0.00	X	0.00	1.00	-1.34	3.79

Table 3: Artificial instances: 10, 20 and 30 customers

				METAHEURISTIC							CPLEX
$\|I\|$	$\|E\|$	$\|K1\|$	$\|K2\|$	o.f	Node Move	Node Exchange	Best Criterion	Gap between criteria	t (s)	% Gap from CPLEX	% Optimality Gap
40 CUSTOMERS											
2	38	20	49	96,786.96	182.60	0.00	2	11.26	9.00		n.s.
5	35	18	45	89,790.61	79.02	166.66	1	210.38	14.00		n.s.
10	30	14	38	75,791.53	298.07	264.11	1	147.79	11.00		n.s.
15	25	12	31	67,958.61	0.00	853.21	1	437.45	10.00		n.s.
20	20	12	29	69,465.86	688.83	837.90	2	699.74	9.00		n.s.
25	15	14	36	76,566.41	0.00	477.06	1	574.50	9.00		n.s.
30	10	17	43	87,129.18	330.23	331.50	1	446.68	11.00		n.s.
35	5	19	48	96,135.51	0.00	0.00	X	0.00	18.00		n.s.
38	2	20	51	100,694.59	278.59	15.50	2	146.47	26.00		n.s.
2	38	0	59	94,185.86	0.00	155.79	X	0.00	7.00		n.s.
5	35	0	54	87,477.82	0.00	136.83	X	0.00	10.00		n.s.
10	30	0	45	74,220.82	172.50	679.59	2	261.49	11.00		n.s.
15	25	0	37	66,582.24	0.00	853.21	1	9.65	8.00		n.s.
20	20	0	35	68,060.10	0.00	690.89	2	267.46	9.00		n.s.
25	15	0	43	74,941.72	0.00	1,045.09	2	201.28	7.00	-19.16	18.69
30	10	0	51	85,442.64	0.00	57.17	2	105.16	7.00		n.s.
35	5	0	58	93,870.67	20.22	0.00	X	0.00	25.00		n.s.
38	2	0	61	97,832.13	0.00	178.24	X	0.00	23.00	-13.26	13.68
50 CUSTOMERS											
2	48	22	56	129,245.05	2.69	0.00	1	142.91	39.00		n.s.
5	45	21	54	124,640.83	0.00	187.81	1	229.38	44.00		n.s.
10	40	18	50	115,950.50	0.00	178.44	1	378.54	26.00		n.s.
15	35	17	42	101,052.21	0.00	652.91	1	1,617.57	36.00		n.s.
20	30	13	37	91,561.14	0.00	495.20	1	1,280.39	17.00		n.s.
25	25	11	32	83,273.59	25.25	408.15	2	857.91	17.00		n.s.
30	20	12	32	85,351.20	0.00	210.60	2	663.43	16.00		n.s.
35	15	15	39	97,350.77	0.00	57.34	2	296.27	20.00		n.s.
40	10	17	46	108,653.18	0.00	446.74	1	54.47	21.00		n.s.
45	5	20	50	116,059.43	15.87	310.08	1	164.20	40.00		n.s.
48	2	22	55	126,499.99	15.87	0.00	1	169.28	42.00		n.s.
2	48	0	67	124,536.09	0.00	317.35	2	494.13	25.00		n.s.
5	45	0	65	120,406.58	165.01	371.93	2	159.23	31.00		n.s.
10	40	0	59	112,487.21	0.00	0.00	2	196.24	19.00		n.s.
15	35	0	51	98,374.91	577.13	566.67	1	994.41	19.00		n.s.
20	30	0	44	89,079.62	218.64	703.37	1	782.34	19.00		n.s.
25	25	0	38	81,241.01	0.00	1,551.66	2	180.11	17.00		n.s.
30	20	0	38	83,161.03	0.00	3,172.85	1	493.30	17.00		n.s.
35	15	0	47	93,779.99	921.21	1,127.10	1	253.52	29.00		n.s.
40	10	0	55	104,644.95	0.44	255.56	X	0.00	19.00		n.s.
45	5	0	60	111,911.64	0.00	431.58	2	50.18	37.00		n.s.
48	2	0	66	121,785.65	0.00	431.58	2	50.18	38.00		n.s.

Table 4: Artificial instances: 40 and 50 customers

Instances	$\|I\|$	$\|E\|$	$\|K1\|$	$\|K2\|$	Carrier	Metaeuristic		
					Distances	Distances	Saving(km)	Saving(%)
Instance 1	9	34	2	40	16891	16403	488	2.89
Instance 2	13	32	8	31	14986	13097	1889	12.61
Instance 3	4	34	4	39	15094	14437	657	4.35
Instance 4	8	35	6	36	14232	12961	1271	8.93
Instance 5	4	31	3	41	14251	13660	591	4.15

Table 5: Real instances