



A matheuristic for a multimodal long haul routing problem

David Wolfinger^{1,2}  · Fabien Tricoire^{2,3}  · Karl F. Doerner^{1,2,4} 

Received: 5 November 2017 / Accepted: 25 July 2018 / Published online: 22 August 2018
© The Author(s) 2018

Abstract

We address a planning problem faced by logistics service providers who transport freight over long distances. Given a set of transportation requests, where the origin and the destination of each request are located far apart from each other, a logistics service provider must find feasible vehicle routes to fulfil those requests at minimum cost. When transporting freight over long distances, multimodal transportation provides a viable alternative to traditional unimodal road transportation. We introduce this new problem, which we call the multimodal long haul routing problem (MMLHRP), and present a mathematical formulation for it. Furthermore, we propose a matheuristic, using iterated local search within a column generation framework, for solving the MMLHRP. Results show that large cost savings can be achieved through multimodal transportation compared to unimodal road transportation.

Keywords Multimodal transportation · Column generation · Iterated local search

Mathematics Subject Classification 90B06 · 90C11

1 Introduction

Long-distance transportation plays an increasingly important role in freight transportation in Europe. In 2015 more than 40% of freight volumes transported on the road within the European Union were carried on distances larger than 500 km. This ratio is expected to increase even further in the future (Eurostat 2016a).

✉ David Wolfinger
david.wolfinger@univie.ac.at

¹ Christian Doppler Laboratory for Efficient Intermodal Transport Operations, University of Vienna, Vienna, Austria

² Department of Business Administration, University of Vienna, Vienna, Austria

³ Institute for Production and Logistics Management, Johannes Kepler University, Linz, Austria

⁴ Data Science @ University of Vienna, Vienna, Austria

A long-distance transport can be split up in three segments: *first-mile* (i.e., the pickup process), *long haul* (i.e., the long-distance transportation), and *last-mile* (i.e., the delivery process). Typically the first- and last-mile transports need to be conducted by road transportation as most companies do not have a direct connection to a rail network and are not located at a port. For the long haul, on the other hand, additional modes of transportation such as rail or water can be considered. The main advantages of combining several modes of transportation are both lower costs and a lower environmental impact compared to traditional unimodal road transportation (StadieSeifi et al. 2014).

Although the rail network in Europe is well elaborated and maintained, and the Rhine-Main-Danube axis provides an excellent opportunity to include inland waterways in the transport chain (especially in Central and East Europe), freight transportation in Europe is currently still mainly done via road. The statistical office of the European Union reports a modal split of 75.4, 18, and 6.6% for road transportation, rail transportation, and inland waterway transportation, respectively, for the year 2014 (Eurostat 2016b).

In this article we consider an operational problem faced by logistics service providers who transport freight over long distances. That is, given a set of transport requests where the origin and the destination of each request are located far apart from each other, the logistics service providers must find feasible vehicle routes to fulfil those requests at minimum cost. We consider the full truck load version of this problem. That is, we consider only full truck loads in the first- and last-mile, and allow consolidation of requests only on long haul vehicles. This version of the problem arises in many industrial contexts such as transportation of raw materials or steel products, where more than one short haul vehicle is needed to transport a single request. We consider a heterogeneous fleet of long haul vehicles corresponding to different modes of transportation and assume that the logistics service providers do not operate a fleet of short haul vehicles themselves, but utilize carriers. A vehicle which conducts a long-distance transport might have to travel for several days. In many cases, labour regulations prohibit a vehicle (e.g. a truck with just one driver) from travelling such a long time non-stop. In order to take such constraints into account we consider a daily operation time limit for each vehicle. Furthermore, we assume that a request may be split among several long haul vehicles (possibly associated with different transportation modes). That is, we allow a request to be split into an arbitrary number of parts. Each of those parts may be transported on a different path from the origin to the destination.

Research in the area of multimodal freight transportation planning increased considerably in recent years (StadieSeifi et al. 2014). An overview of the literature in the field of multimodal transportation planning can be found in the two most recent surveys of StadieSeifi et al. (2014) and Caris et al. (2013).

A vast body of work deals with the Service Network Design Problem (SNDP), i.e., the (optimal) selection and scheduling of transportation services and choosing associated transportation modes, and determining the routing and flow of freight through the selected service network. A service is characterized by its origin, destination, and intermediate terminals (its route), its frequency and service capacity, and its transportation mode. A mode is characterized by its speed, loading capacity, and price. Additionally, services and modes are usually assumed to have fixed costs.

Such *tactical* planning problems are typically modelled as fixed-cost capacitated multicommodity network design formulations on time-space networks. Recently, vehicle routing aspects have also been considered in SNDPs in the form of asset (e.g. vehicles) management. A limited number of vehicles is assigned to each terminal (to which they must return). Vehicle management is then mainly incorporated as “design-balance constraints” (Pedersen et al. 2009) requiring that the number of vehicles entering and leaving a terminal must be the same, or through the design of cycles (see, for instance, Andersen et al. 2009; Andersen and Christiansen 2009; Crainic et al. 2016).

However, these formulations do not consider the routing of first- and last-mile transportation. A very recent work which does consider it is by Medina et al. (2018), who essentially combine the SNDP with the Vehicle Routing Problem (VRP) and call this new problem the Service Network Design and Routing Problem (SNDRP). In this problem each supplier is assigned to a single terminal in a long haul consolidation network to which it sends goods, and each customer is assigned to a single terminal from which its shipments are delivered. Direct shipments from suppliers to terminals are assumed, but the problem allows for delivery routes that visit several customers. The first-mile together with the consolidation network are then modelled as a SNDP, while the delivery routes are modelled as a VRP. The authors present two formulations on a time-space network (one route-based and one arc-based) and solve both with a dynamic discretisation discovery algorithm.

The literature on *operational* multimodal transportation planning is still scarce, though (StadieSeifi et al. 2014). Moreover, only a small part of those studies concern multimodal *vehicle routing*. Research on multimodal routing is mostly limited to commodity flow formulations. Such formulations usually assume an unlimited availability of first- and last-mile modes of transportation, and known fixed schedules and free capacities for long haul modes of transportation. Shipments are then routed through the multimodal network by deciding upon the usage of those pre-defined scheduled and unscheduled services (see for example Barnhart and Ratliff 1993; Ziliaskopoulos and Wardell 2000; Chang 2008; Moccia et al. 2011).

Bock (2010) studies a multimodal vehicle routing problem. The model integrates multiple modes of transportation, transshipments, dynamic disturbances, and partial or total outsourcing of transportation services. However, split loads are not considered and the end of every time window is considered as soft, i.e. late arrivals are allowed. The dynamic problem is modelled as a rolling horizon scheme and solved with a variable neighbourhood search algorithm.

Two relevant related problems are the *Pickup and Delivery Problem with transshipment (PDPT)* and the *Pickup and Delivery Problem with split loads (PDPSL)*.

Regarding the *PDPT* Mitrovic-Minic and Laporte (2006) develop a two-phase heuristic solution method. They test it on small instances and show the usefulness of performing transshipments. Cortés et al. (2010) present an arc-based mixed-integer formulation for the problem which they solve by means of a branch-and-cut method. Qu and Bard (2012) solve the *PDPT* with a GRASP which uses an adaptive large neighbourhood search in the improvement phase. Masson et al. (2013) use a pure adaptive large neighbourhood search to tackle the *PDPT* in a passenger transportation application. Rais et al. (2014) propose a new mixed-integer formulation which

they solve with a commercial solver on small instances. Recently, Neves-Moreira et al. (2016) proposed a fix-and-optimize matheuristic for the *PDPT* with additional side constraints. The authors test their solution approach on real-world instances of a long haul freight transportation problem.

Regarding split loads, Nowak et al. (2008) propose a heuristic algorithm for the *PDPSL* and solve several random large-scale instances. Andersson et al. (2011) and Stålhane et al. (2012) solve the maritime version of the problem. The former suggests a solution method based on a priori generation of single ship schedules and two path flow models that deal with the selection of ship schedules and assignment of quantities to the schedules. The latter solves the problem by means of a branch-and-price-and-cut method. Both test their methods on randomly generated instances.

The contribution of our paper is threefold. First, we introduce a new problem, which we call the multimodal long haul routing problem (MMLHRP). To the best of our knowledge, no previous work has combined the routing of long haul vehicles associated with different transportation modes with the scheduling of both first- and last-mile transportation, where requests are allowed to be split and vehicle operation time limits are considered. Second, we propose a matheuristic, using iterated local search within a column generation framework, for solving the MMLHRP. Third, we present managerial insights regarding both the logistics of multimodal transport and the potential cost savings through multimodal transport. These insights are based on results of computational experiments which are based on real data.

The remainder of this article is structured as follows: In Sect. 2, we give a formal description of the problem. In Sect. 3, a set covering formulation for the MMLHRP and a column generation algorithm to solve its linear relaxation are presented. In Sect. 4, we describe a matheuristic based on column generation for the MMLHRP, while computational experiments and a managerial discussion are presented in Sect. 5. Conclusions are reported in Sect. 6.

2 Problem description

We are given a set of requests R . Each request $r \in R$ is associated with a pickup location r^+ , a delivery location r^- , a positive load d_r , and two time windows; one for the pickup location $([a_{r^+}, b_{r^+}])$ and one for the delivery location $([a_{r^-}, b_{r^-}])$.

Furthermore, a heterogeneous fleet of vehicles K is given. Each vehicle $k \in K$ has a non-negative capacity q_k , a start depot o_k at which it must start its tour, and an end depot η_k at which it must end its tour. We consider two types of vehicles: short haul vehicles (e.g. trucks) and long haul vehicles (e.g. ships or trains). Long haul vehicles are further split up into active vehicles and passive vehicles. The difference between an active and a passive vehicle is, that an active vehicle can move on its own while a passive vehicle cannot. This means, that a passive vehicle must always move together with an active vehicle. The number of passive vehicles an active vehicle pulls/pushes may vary throughout its tour, but must not exceed a maximum number u at any given point in time. Each vehicle is subject to a limit e_k on the amount of hours per day it is allowed to operate, which must not be exceeded.

To facilitate the transshipment of load from short haul vehicles to long haul vehicles, and vice versa, a set of transshipment locations T is given. We assume that there are neither waiting times nor capacity restrictions at transshipment locations. Furthermore, we assume site dependency, that is to say, only short haul vehicles are able to visit every location, while long haul vehicles may only visit their respective depots and transshipment locations.

The problem is defined on a directed graph $G = (V, A)$, where V and A are the vertex and arc sets, respectively. V is composed of the start and end depot for each vehicle, the set of pickup nodes, the set of delivery nodes, and the set of transshipment nodes. The set of arcs $A = \bigcup_{k \in K} A^k$, where $A^k \subset V \times V$, defines the feasible movements of each vehicle $k \in K$ between the different nodes in V .

With each arc $(i, j) \in A$ and each vehicle $k \in K$ is associated a travel cost c_{ijk} and a travel time t_{ijk} . Furthermore, with each transshipment node $i \in T$ is associated a transshipment cost m_i per transferred unit. We assume that the triangle inequality holds for travel costs and travel times.

Service at a customer location must start within the prescribed time window. A vehicle may arrive prior to the beginning of the time window, but, in this case, must wait until the time window starts before starting the service. A time window is also associated with the depots. The time needed for service at any location depends on both the quantity to (un)load and the vehicle type. h_{ik} denotes the time needed for vehicle $k \in K$ to (un)load one unit at vertex $i \in V$.

As already mentioned, we assume that the LSPs do not operate a fleet of short haul vehicles themselves, but utilize carriers. Consequently, only trips where a short haul vehicle carries load raise costs, and, therefore, only these trips need to be considered. Furthermore, we restrict short haul vehicle transportation to full truck loads and only allow consolidation of requests on long haul vehicles. The load of a single request may be split among several long haul vehicles or even transshipped to/from the same long haul vehicle at several transshipment locations. Each part of a request may thus be transported on a different path from the pickup location to the delivery location. Consequently, each split makes it more complicated for practitioners. That is, the number of routes, logistics operations, and administrative tasks increases with each split, as well as both the routes and the product flows become more complex. Therefore, practitioners try to avoid splitting the load of a request too often. To take this into account we assume the following transshipment policy (P1): always transship as much units of load of a request as possible. That is, in case of a transshipment to a long haul vehicle, transship units of load equal to either the free capacity of the long haul vehicle or the residual load of the request. In case of a transshipment from a long haul vehicle, unload the complete amount of units of load of the request currently on the long haul vehicle.

Following the idea of generating practical solutions, we assume that each unit of load may use at most one long haul vehicle. Thus, a unit of load can only be transported in one of two ways:

1. **unimodal**: using only a short haul vehicle to directly transport the unit of load from its pickup location to its delivery location

- multimodal:** using a short haul vehicle only for the first- and last-mile, and using one, and only one, long haul vehicle for the long-distance transportation in between

The objective is to find a set of feasible short haul vehicle trips and long haul vehicle routes that serves all requests, such that the sum of travel costs and transshipment costs is minimized.

A feasible trip for a short haul vehicle corresponds to an arc in the graph G , while a feasible route for a long haul vehicle corresponds to a path from the start depot to the end depot in the graph G . Both a trip and a path are feasible if the vehicle capacity and the limit on the amount of hours per day the vehicle is allowed to operate are never exceeded. For short haul vehicles, the time windows at the customer nodes must be respected and, in case a short haul vehicle performs a direct transport of a request from its pickup location to its delivery location, the pickup node must be visited before the delivery node. Similarly, for long haul vehicles the time windows at the start depot and end depot must be respected and a (partial load of a) request must always be loaded onto the vehicle before it is unloaded from the vehicle. Furthermore, if a transshipment takes place the vehicle which receives the load must start its service at the transshipment node after the vehicle which gives the load.

Figure 1 illustrates a small problem instance (Fig. 1a) together with a feasible solution (Fig. 1b). It consists of four requests (1, 2, 3, 4) and five transfer locations (a, b, c, d, e). The transfer locations with bold borders represent the depots. The long haul of requests 1 and 2 is performed by a water bound long haul vehicle (e.g. a ship), while the long haul of request 3 is performed by a rail bound long haul vehicle (e.g. a train). For all three requests only the first-/and last-mile transports are performed by short haul vehicles. Because the origin location of request 4 is too far away from any transfer location, it is transported from its origin location to its destination location exclusively with short haul vehicles. The route of the water bound long haul vehicle (dashed line) is: $a - b - c - a$, the route of the rail bound long haul vehicle (dotted line) is: $d - e - d$, and the short haul vehicle trips (solid lines) are: $1^+ - a, 2^+ - b, 3^+ - d, 4^+ - 4^-, c - 1^-, c - 2^-, e - 3^-$.

Using the above notation, we can model the MMLHRP without policy P1 as an arc flow formulation (see Appendix 1). However, this model is of limited use as

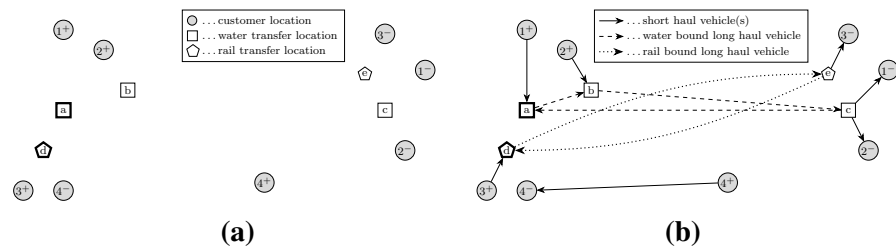


Fig. 1 Small problem instance (a) consisting of four requests (1, 2, 3, 4) and five transfer locations (a, b, c, d, e) together with a feasible solution (b). The transfer locations with bold borders represent the depots

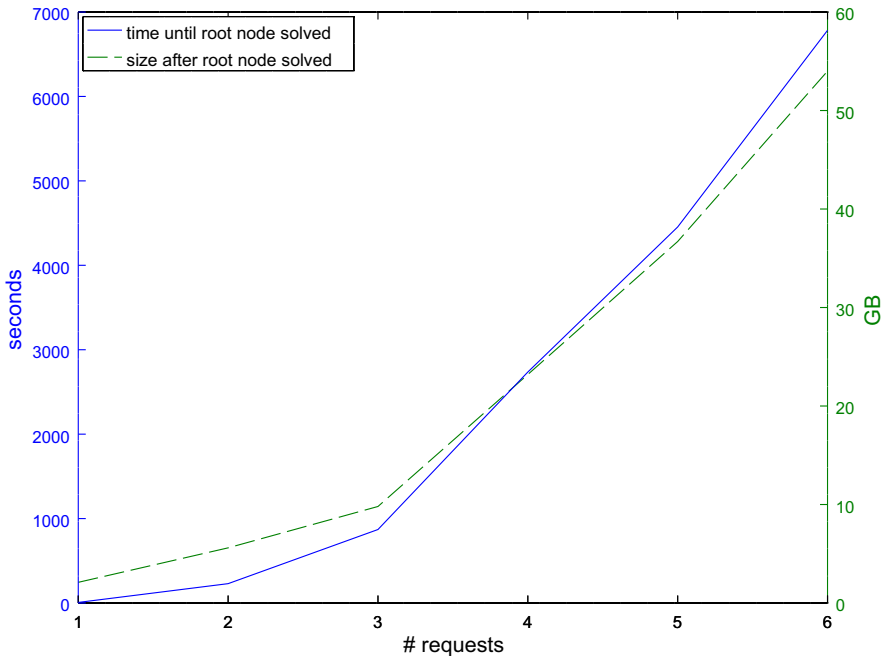


Fig. 2 Average time and memory consumption of the arc flow formulation using a commercial solver

even this simplified version of the problem is barely solvable with a commercial solver for tiny instances with respect to both solution time and memory usage. Figure 2 shows the increase in both the time until the root node is solved and the size of the process in the memory after the root node has been solved as a function of the number of requests. The depicted values are an average of five instances per number of requests. The number of locations in an instance ranges from seven in instances with one request to 29 in instances with six requests. In each instance one water bound long haul vehicle, one rail bound long haul vehicle, and a sufficient amount of short haul vehicles (between 38 and 363 vehicles) are available to transport all requests. From Fig. 2 it becomes clear that a more sophisticated solution technique, which performs better and allows us to implement policy P1, is required.

3 A set covering formulation for the MMLHRP

Because the arc flow formulation is of limited use for generating lower bounds for the MMLHRP (see Fig. 2) we use a set covering formulation instead. Besides the fact that for vehicle routing, set covering formulations yield better lower bounds than compact formulations (Semet et al. 2014), using a set covering formulation allows us to effectively solve larger problem instances. First, let us introduce some additional notation. Let Ω be the set of all feasible short haul vehicle trips from the pickup location of a request to its delivery location. Such a trip corresponds to an arc in the graph G of the

form $(r^+, r^-) : r \in R$, and represents a direct unimodal transport of a request. In addition, Ω contains all feasible sets $\mathcal{C} = \{\mathcal{H}, \mathcal{S}\}$, where \mathcal{H} is a long haul vehicle route and \mathcal{S} is a set of first- and last-mile trips of short haul vehicles. \mathcal{H} corresponds to a path in the graph G from the vehicle's start depot to its end depot. A trip $s \in \mathcal{S}$ corresponds to an arc in the graph G either of the form $(r^+, t) : r \in R, t \in T$ or of the form $(t, r^-) : r \in R, t \in T$. The former type of a trip represents a first-mile transport, while the latter represents a last-mile transport. A set \mathcal{C} is feasible if, and only if, \mathcal{H} is feasible, all $s \in \mathcal{S}$ are feasible, and \mathcal{H} and \mathcal{S} together allow for a feasible flow of each request transported by the long haul vehicle from its origin to its destination. A set \mathcal{C} , thus, represents one long haul vehicle route together with the necessary first- and last-mile transports by short haul vehicles. Consequently, a column $j \in \Omega$ consists of at least one short haul vehicle trip, and at most one long haul vehicle route.

Let $P = \bigcup_{r \in R} r^+$ be the set of pickup nodes, and δ_i be the demand of pickup node $i \in P$. Let \mathcal{K}^{SHV} be the set of short haul vehicle types (e.g., truck), \mathcal{K}^{AV} be the set of active long haul vehicle types (e.g., ship or train), and \mathcal{K}^{PV} be the set of passive long haul vehicle types (e.g., barge). Then, $\mathcal{K} = \mathcal{K}^{SHV} \cup \mathcal{K}^{AV} \cup \mathcal{K}^{PV}$ is the set of all vehicle types. We denote with \mathcal{W}_k the number of available vehicles of type $k \in \mathcal{K}$. For each column $j \in \Omega$ let \hat{c}_j be the cost of the column. μ_{ij} denotes the quantity of pickup node $i \in P$ delivered in column $j \in \Omega$, while ρ_{kj} is the number of vehicles of type $k \in \mathcal{K}$ used in column $j \in \Omega$. The non-negative integer variable θ_j indicates the number of times column $j \in \Omega$ is in the solution.

With the above notation, the MMLHRP can be formulated as follows:

$$\min \sum_{j \in \Omega} \hat{c}_j \theta_j \tag{1}$$

subject to

$$\sum_{j \in \Omega} \mu_{ij} \theta_j \geq \delta_i \quad \forall i \in P \tag{2}$$

$$\sum_{j \in \Omega} \rho_{kj} \theta_j \leq \mathcal{W}_k \quad \forall k \in \mathcal{K} \tag{3}$$

$$\theta_j \in \mathbb{N} \quad \forall j \in \Omega \tag{4}$$

The objective function (1) aims at minimizing the total costs. Constraints (2) ensure that the demand of each request is satisfied, while constraints (3) restrict the number of used vehicles in a solution to the number of available vehicles. Constraints (4) impose integrality requirements on the decision variable θ_j .

Due to the large size of Ω , solving the problem defined by (1)–(4) explicitly is impractical. Instead, we solve its LP-relaxation, called *Master Problem (MP)*, with a column generation approach. To obtain the LP-relaxation the integrality requirements (4) are replaced by

$$\theta_j \geq 0 \quad \forall j \in \Omega \tag{5}$$

The aim of the column generation approach is to solve the *MP* by finding a subset $\Omega' \subseteq \Omega$ of columns, such that solving $MP(\Omega')$ also solves *MP*. $MP(\Omega')$ is called the *Restricted Master Problem* (RMP).

The approach starts with an initial set of columns and then iteratively solves the RMP and a *pricing subproblem* to produce columns with negative reduced cost, using dual information from the RMP. In each iteration these new columns are added to the RMP to obtain new dual information. This procedure is repeated until no new columns can be found. At that point an optimal solution of the RMP, and therefore of *MP*, has been found, which represents a lower bound for the problem (1)–(4).

3.1 The pricing subproblem

The pricing subproblem aims at finding a variable θ_j that has the lowest reduced cost with respect to a given dual solution to the RMP. Let γ_i and β_k be the dual variables associated with constraints (2) and (3), respectively. The reduced cost \bar{c}_j of a variable θ_j are given by

$$\bar{c}_j = \hat{c}_j - \sum_{i \in P} \mu_{ij} \gamma_i - \sum_{k \in \mathcal{K}} \rho_{kj} \beta_k \tag{6}$$

The subproblem is defined on graph G and its formulation makes use of the variables and sets as defined in Appendix 1. Some of which we repeat here for the sake of clarity:

Variables

- f_{ijrk} amount of request $r \in R$ transported by vehicle $k \in K$ over arc $(i, j) \in A$
- $w_{irkk'}$ amount of request $r \in R$ transshipped from vehicle $k \in K$ to vehicle $k' \in K$ at node $i \in T$
- $x_{ijk} \begin{cases} 1 & \text{iff vehicle } k \in K \text{ traverses arc } (i, j) \in A \\ 0 & \text{otherwise} \end{cases}$

Sets and parameters

- | | | | |
|-------|-------------------------------|--------------|---|
| B | set of short haul vehicles | c_{ijk} | cost of traversing arc $(i, j) \in A^k$ for vehicle $k \in K$ |
| C | set of active vehicles | m_i | transshipment cost per unit at transship. node $i \in T$ |
| E | set of passive vehicles | r^+ | pickup node of request $r \in R$ |
| K | $B \cup C \cup E$ | r^- | delivery node of request $r \in R$ |
| q_k | capacity of vehicle $k \in K$ | o_k/η_k | start-/end-depot of vehicle $k \in K$ |

Using these variables \hat{c}_j can be given by

$$\hat{c}_j = \sum_{(i,j) \in A^k} \sum_{k \in K} c_{ijk} x_{ijk} + \sum_{i \in T} \sum_{r \in R} \sum_{k \in K} \sum_{k' \in K} m_i w_{irkk'}$$

Consequently, the subproblem then is

$$\begin{aligned} \min \quad & \sum_{(i,j) \in A^k} \sum_{k \in K} c_{ijk} x_{ijk} + \sum_{i \in T} \sum_{r \in R} \sum_{k \in K} \sum_{k' \in K} m_i w_{irkk'} \\ & - \sum_{r \in R} \sum_{j \in V} \sum_{k \in K} \gamma_{r^+} f_{r^+ j r k} - \sum_{k \in \mathcal{K}} \beta_k \left(\sum_{\substack{k' \in K : \\ \text{type}(k') = k}} \sum_{\substack{j \in V \setminus \\ \{\eta_{k'}\}}} x_{o_{k'} j k'} \right) \end{aligned} \tag{7}$$

subject to

$$\text{route- and transshipment-feasibility constraints (26)–(44) \quad (see Appendix 1)} \tag{8}$$

$$\text{transshipment policy constraints} \tag{9}$$

Constraints (9) ensure that transshipment policy P1 is satisfied. P1 says to always transship as much units of load of a request as possible. For a more detailed description of P1 see Sect. 2. The function $\text{type}(x)$ returns the vehicle type $k \in \mathcal{K}$ of vehicle $x \in K$. The purpose of this function is purely notational, namely to establish a link between the set of vehicles K of the arc flow formulation and the set of vehicle types \mathcal{K} of the set covering formulation.

Due to the property of a column that it contains at most one long haul vehicle route and because all vehicles of the same type are assumed to be identical, one subproblem for each vehicle type arises. Each subproblem can be solved independently from the others. Also, since passive vehicles cannot move on their own, no separate subproblem is required for them. Rather, passive vehicles are taken into account together with active vehicles in one subproblem. Thus, the set of vehicles K in each subproblem contains only short haul vehicles and at most one long haul vehicle.

3.2 Solving the pricing subproblem

We model each pricing subproblem which considers a long haul vehicle type as a *Shortest Path Problem with Resource Constraints* (SPPRC). The subproblem which considers only short haul vehicles, on the other hand, can be solved by generating for each request $r \in R$ a column j which contains only the arc (r^+, r^-) . The cost of such a column is $\hat{c}_j = c_{r^+ r^- k'}$, where $k' \in B$. The quantity delivered in such a column is $\mu_{r^+ j} = \min(q_{k'}, \delta_{r^+})$, while the number of vehicles used is $\rho_{kj} = 1$, where

$k \in \mathcal{K}^{SHV}$. We use the set of all such columns as the initial set Ω' for the column generation approach.

Each SPPRC is defined on an auxiliary graph $G' = (V', A')$. The vertex set V' contains the source vertex s and the sink vertex z , representing the start and end depot. Additionally, V' contains two sets of vertices Π^+ and Π^- . Each vertex $\pi_{ir}^+ \in \Pi^+$ (called pickup-transfer-node, PTN) represents at least one (possibly several) short haul vehicle trip(s) (r^+, i) plus a transshipment of request $r \in R$ from the short haul vehicle(s) to the long haul vehicle at transshipment location $i \in T$. Each vertex $\pi_{ir}^- \in \Pi^-$ (called delivery-transfer-node, DTN) represents at least one (possibly several) short haul vehicle trips (i, r^-) plus a transshipment of request $r \in R$ from the long haul vehicle to the short haul vehicle(s). With each vertex $v \in V'$ is associated a time window $[a_v, b_v]$. The time windows are generated during pre-processing in the following way:

- $a_s = a_{o_k}, b_s = b_{o_k}, a_z = a_{\eta_k},$ and $b_z = b_{\eta_k},$ where $k \in C \cup E$
- $a_{\pi_{ir}^+} = a_{r^+} + t_{r^+ik},$ where $k \in B$
- $b_{\pi_{ir}^+} = \max_{j \in T} (b_{r^-} - t_{jr-k} - t_{ijk'}),$ where $k \in B$ and $k' \in C \cup E$
- $a_{\pi_{ir}^-} = 0$
- $b_{\pi_{ir}^-} = b_{r^-} - t_{ir-k},$ where $k \in B$

The arc set A' contains one arc from vertex s to each PTN, one arc for each feasible connection between and within all PTNs and DTNs, and one arc from each DTN to vertex z . In total, A' contains 12 different types of arcs. Each arc type corresponds to a feasible action which the long haul vehicle can do. Table 1 gives a detailed description of each arc type.

Figure 3 gives an example of an auxiliary graph G' . It illustrates a small problem instance (Fig. 3a) together with the corresponding auxiliary graph for the water bound long haul vehicle (Fig. 3b). For the sake of clarity only one arc of each arc type is given as an example in Fig. 3b. The figure also indicates vertices which can be eliminated because they will never be part of an optimal solution (vertices with dashed borders). In general, every pair of a PTN and a DTN, where the travel cost of the corresponding first- and last-mile transports is larger than the travel cost of the direct transport from the pickup location to the delivery location of the request will not be part of any optimal solution. Consequently, every vertex which forms a part of only such non-eligible pairs will not be part of any optimal solution, and can therefore be eliminated.

In the SPPRC one seeks to find a cheapest (shortest) path from the source vertex s to the sink vertex z . A common way to solve the SPPRC is to use a labeling algorithm (Feillet et al. 2004; Ropke and Cordeau 2009; Desaulniers 2010; Archetti et al. 2011). Such a dynamic programming algorithm builds feasible (partial) paths in G' which start at the source vertex s and end at some vertex $i \in V'$. Each existing partial path is extended along the arcs leaving its end vertex. The algorithm starts with the partial path containing only vertex s and proceeds

Table 1 Description of arc types in A'

Type	Arc	Domain	Action of long haul vehicle	Cost
1	(s, π_{ir}^+)	$\forall i \in T, \forall r \in R$	Leave the depot	$c_{s, \pi_{ir}^+} = c_{o_k ik}$, where $k \in C \cup E$
2	(π_{ir}^+, π_{ip}^+)	$\forall i \in T, \forall r, p \in R, r \neq p$	Receive load of two distinct requests at the same transshipment location	$c_{\pi_{ir}^+, \pi_{ip}^+} = 0$
3	(π_{ir}^+, π_{jp}^+)	$\forall i, j \in T, \forall r, p \in R, i \neq j, r \neq p$	Receive load of two distinct requests at two distinct transshipment locations	$c_{\pi_{ir}^+, \pi_{jp}^+} = c_{ijk}$, where $k \in C \cup E$
4	(π_{ir}^-, π_{ip}^-)	$\forall i \in T, \forall r, p \in R, r \neq p$	Unload two distinct requests at the same transshipment location	$c_{\pi_{ir}^-, \pi_{ip}^-} = 0$
5	(π_{ir}^-, π_{jp}^-)	$\forall i, j \in T, \forall r, p \in R, i \neq j, r \neq p$	Unload two distinct requests at two distinct transshipment locations	$c_{\pi_{ir}^-, \pi_{jp}^-} = c_{ijk}$, where $k \in C \cup E$
6	(π_{ir}^+, π_{ip}^-)	$\forall i \in T, \forall r, p \in R, r \neq p$	Receive load of a request then unload another request at the same transshipment location	$c_{\pi_{ir}^+, \pi_{ip}^-} = 0$
7	(π_{ir}^+, π_{jr}^-)	$\forall i, j \in T, \forall r \in R, i \neq j$	Receive load of a request at one transshipment location then unload it at another transshipment location	$c_{\pi_{ir}^+, \pi_{jr}^-} = c_{ijk}$, where $k \in C \cup E$
8	(π_{ir}^+, π_{jp}^-)	$\forall i, j \in T, \forall r, p \in R, i \neq j, r \neq p$	Receive load of a request at one transshipment location then unload another request at another transshipment location	$c_{\pi_{ir}^+, \pi_{jp}^-} = c_{ijk}$, where $k \in C \cup E$
9	(π_{ir}^-, π_{ip}^+)	$\forall i \in T, \forall r, p \in R, r \neq p$	Unload a request then receive load of another request at the same transshipment location	$c_{\pi_{ir}^-, \pi_{ip}^+} = 0$
10	(π_{ir}^-, π_{jr}^+)	$\forall i, j \in T, \forall r \in R, i \neq j$	Unload a request at one transshipment location then receive load of the same request at another transshipment location	$c_{\pi_{ir}^-, \pi_{jr}^+} = c_{ijk}$, where $k \in C \cup E$
11	(π_{ir}^-, π_{jp}^+)	$\forall i, j \in T, \forall r, p \in R, i \neq j, r \neq p$	Unload a request at one transshipment location then receive load of another request at another transshipment location	$c_{\pi_{ir}^-, \pi_{jp}^+} = c_{ijk}$, where $k \in C \cup E$
12	(π_{ir}^-, z)	$\forall i \in T, \forall r \in R$	Return to the depot	$c_{\pi_{ir}^-, z} = c_{im_k k}$, where $k \in C \cup E$

to create all feasible paths in G' . In order to speed up the algorithm, dominated paths can be discarded throughout the search. Path p dominates path p' if both end at the same vertex, the cost of p is less then or equal to the cost of p' , and all feasible extensions of p' to the sink vertex z are also feasible for p .

A feasible partial path from vertex s to any vertex $i \in V'$ is represented by a label associated with vertex i . A label $\mathcal{L} = (i, \zeta, \mathfrak{R})$ typically has one component ζ to represent the (reduced) cost of the path and a resource vector \mathfrak{R} . We define

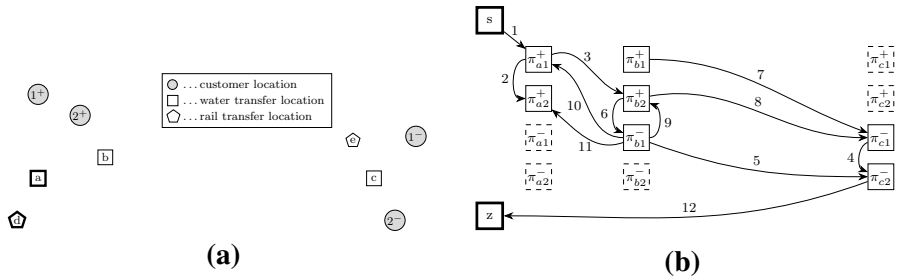


Fig. 3 Small problem instance (a) consisting of two requests (1, 2) and five transfer locations (a, b, c, d, e) together with an illustration of the corresponding auxiliary graph G' for the water bound long haul vehicle (b). The transfer locations with bold borders represent the depots. For the sake of clarity only one arc of each arc type in A' is given as an example. The numbers on the arcs indicate the arc type as given in Table 1. Dashed borders indicate vertices which can be eliminated, because they will never be part of an optimal solution

$\mathfrak{R} = \left(t, l, \mathcal{O}, \mathcal{U}, (d_1^+, \dots, d_{|R|}^+), (d_1^-, \dots, d_{|R|}^-), g \right)$. Each component of \mathfrak{R} represents the consumption of a resource (e.g., time) along the path until vertex i . We adapt the label-setting algorithm from Ropke and Cordeau (2009) and consider labels of the form $\mathcal{L}_k = \left(i, \zeta, t, l, \mathcal{O}, \mathcal{U}, (d_1^+, \dots, d_{|R|}^+), (d_1^-, \dots, d_{|R|}^-), g \right)$ where

- $k \in \mathcal{K}^{AV}$, indicates the type of long haul vehicle the label represents
- $i \in V'$ is the vertex of the label
- ζ is the accumulated cost
- t is the time when leaving vertex i
- l is the load of the vehicle when leaving vertex i
- $\mathcal{O} \subseteq R$ is the set of open requests. A request is considered to be open when the amount of its demand which has already been picked up is larger than the amount delivered.
- $\mathcal{U} \subseteq R$ is the set of unreachable requests. A request r is considered to be unreachable if either the whole load of it has already been picked up, or if every extension $(i, \pi_{jr}^+) : j \in T$ would violate the time window at vertex π_{jr}^+ .
- $d_r^+, r \in R$, indicates how many units of the load of request r have already been transhipped to the vehicle
- $d_r^-, r \in R$, indicates how many units of the load of request r have already been transhipped from the vehicle
- g gives the number of passive vehicles in use when leaving vertex i

In accordance with Ropke and Cordeau (2009) we use the notation $\zeta(\mathcal{L}_k)$ to refer to the cost of label \mathcal{L}_k and consequently $i(\mathcal{L}_k), t(\mathcal{L}_k), l(\mathcal{L}_k), \mathcal{O}(\mathcal{L}_k), \mathcal{U}(\mathcal{L}_k), d_r^+(\mathcal{L}_k), d_r^-(\mathcal{L}_k)$, and $g(\mathcal{L}_k)$ for the resources. Additionally, let $q(\mathcal{L}_k)$ denote the total capacity of the vehicle when leaving vertex i (including the capacity of all passive vehicles currently in use), $\tau(\mathcal{L}_k)$ denote the physical location of the label (i.e., the depot or transshipment location), $r(i)$ refer to the request associated with vertex $i \in V', \lambda_i$

denote the quantity of the request associated with vertex $i \in V'$ transferred to/from the vehicle at vertex i (note that λ_i is a decision variable), $h(i,k)$ denote the time needed by vehicle type $k \in \mathcal{K}$ for (un)loading one unit of load at node i , and $r^+(i)$ and $r^-(i)$ refer to the pickup location and the delivery location of request $r \in R$ associated with vertex $i \in V'$, respectively.

According to Ropke and Cordeau (2009), extending a label \mathcal{L}_k along an arc $(i(\mathcal{L}_k),j)$ is feasible only if one of the following three conditions is satisfied:

$$j \in \Pi^+ \wedge r(j) \notin \mathcal{U}(\mathcal{L}_k) \tag{10}$$

$$j \in \Pi^- \wedge r(j) \in \mathcal{O}(\mathcal{L}_k) \tag{11}$$

$$j = z \wedge \mathcal{O}(\mathcal{L}_k) = \emptyset \tag{12}$$

Condition (10) states that if j is a PTN the respective request r is still reachable. Condition (11) states that if j is a DTN at least a part of the respective request r is currently on the vehicle. Condition (12) states that if j is the sink vertex then all load which has been picked up along the path has also been delivered.

The properties ensuring that the vehicle capacity and the time window are respected need to be adapted as follows:

$$j \in \Pi^- \vee (l(\mathcal{L}_k) < q(\mathcal{L}_k)) \vee (g(\mathcal{L}_k) < u) \tag{13}$$

$$(j \in \Pi^- \wedge (t(\mathcal{L}_k) + t_{i(\mathcal{L}_k),jk} + h(j,k)\lambda_j \leq b_j)) \vee (j \in \Pi^+ \wedge t(\mathcal{L}_k) + t_{i(\mathcal{L}_k),jk} \leq b_j) \tag{14}$$

In order for a label extension to be feasible, properties (13) and (14) need to hold as well. Property (13) states that the vehicle capacity is respected if either j is a DTN, or the current load on the vehicle is less than the capacity of the vehicle, or the current number of passive vehicles in use is less than the maximum number of passive vehicles allowed. Property (14) states that in case j is a DTN, the time window at node j is respected if the time when leaving node i plus the travel time from node i to node j plus the service time at node j is less than or equal to the end of the time window at node j . Otherwise, in case j is a PTN, the time window is respected if the time when leaving node i plus the travel time from node i to node j is less than or equal to the end of the time window at node j . The difference in the calculation of the left hand side is a consequence of the different definitions of b_j for a PTN and a DTN.

If the extension of label \mathcal{L}_k along arc $(i(\mathcal{L}_k),j)$ is feasible then a new label \mathcal{L}'_k is created and the vertex of the label, the cost, and all resources must be updated as follows:

$$i(\mathcal{L}'_k) = j \tag{15}$$

$$\zeta(\mathcal{L}'_k) = \zeta(\mathcal{L}_k) + c_{i(\mathcal{L}_k),j} + g(\mathcal{L}_k)c_{i(\mathcal{L}_k),j} + \begin{cases} -\beta_k & \text{iff } j = z \\ \left\lceil \frac{\lambda_j}{q_{k'}} \right\rceil c_{r^+(j),\tau(j),k'} + \\ \lambda_j m_j - \left\lceil \frac{\lambda_j}{q_{k'}} \right\rceil \beta_{k''} - \\ \gamma_{r^+(j)} \lambda_j^- \\ \text{addIPV}(\mathcal{L}'_k, \mathcal{L}_k) \beta_{k''} & \text{iff } j \in \Pi^+ \\ \left\lceil \frac{\lambda_j}{q_{k'}} \right\rceil c_{\tau(j),r^-(j),k'} + \\ \lambda_j m_j - \left\lceil \frac{\lambda_j}{q_{k'}} \right\rceil \beta_{k''} & \text{iff } j \in \Pi^- \\ k' \in B, k'' \in \mathcal{K}^{SHV} \\ k''' \in \mathcal{K}^{PV} \end{cases} \quad (16)$$

$$t(\mathcal{L}'_k) = t(\mathcal{L}_k) + \text{addTime}\left(\max\left(a_j, t(\mathcal{L}_k) + \text{addTime}\left(t(\mathcal{L}_k), t_{i(\mathcal{L}_k),j,k}\right)\right), h(i(\mathcal{L}_k), k) \lambda_j\right) \quad (17)$$

$$\mathcal{U}(\mathcal{L}'_k) = \mathcal{U}(\mathcal{L}_k) \cup \left\{ r \in R : \nexists \pi_{ir}^+ : t(\mathcal{L}'_k) + \text{addTime}\left(t(\mathcal{L}'_k), t_{j,\pi_{ir}^+,k}\right) \leq b_{\pi_{ir}^+} \right\} \cup \left\{ r(j) : d_{r(j)}^+ = d_{r(j)} \right\} \quad (18)$$

$$l(\mathcal{L}'_k) = \begin{cases} l(\mathcal{L}_k) + \lambda_j & \text{iff } j \in \Pi^+ \\ l(\mathcal{L}_k) - \lambda_j & \text{iff } j \in \Pi^- \end{cases} \quad (19)$$

$$\mathcal{O}(\mathcal{L}'_k) = \begin{cases} \mathcal{O}(\mathcal{L}_k) \cup \{r(j)\} & \text{iff } j \in \Pi^+ \\ \mathcal{O}(\mathcal{L}_k) \setminus \{r(j)\} & \text{iff } j \in \Pi^- \end{cases} \quad (20)$$

$$d_r^+(\mathcal{L}'_k) = \begin{cases} d_r^+(\mathcal{L}_k) + \lambda_j & \text{iff } j \in \Pi^+ \wedge r = r(j) \\ d_r^+(\mathcal{L}_k) & \text{otherwise} \end{cases} \quad (21)$$

$$d_r^-(\mathcal{L}'_k) = \begin{cases} d_r^-(\mathcal{L}_k) + \lambda_j & \text{iff } j \in \Pi^- \wedge r = r(j) \\ d_r^-(\mathcal{L}_k) & \text{otherwise} \end{cases} \quad (22)$$

$$g(\mathcal{L}'_k) = \begin{cases} \left\lceil \frac{l(\mathcal{L}'_k)}{q_{k'}} \right\rceil & \text{iff } j \in \Pi^+, k' \in \mathcal{K}^{PV} \\ g(\mathcal{L}_k) \text{ and create additional labels where} \\ g(\mathcal{L}'_k) = x, \forall x \in \mathbb{N} : \left\lceil \frac{l(\mathcal{L}'_k)}{q_{k'}} \right\rceil \leq x < g(\mathcal{L}_k) & \text{iff } j \in \Pi^-, k' \in \mathcal{K}^{PV} \end{cases} \quad (23)$$

Updating rule (15) sets j as the vertex of the new label. Rule (16) updates the cost (ζ). The cost of the new label is equal to the cost of the old label plus the travel cost of the active vehicle and the passive vehicles along arc $(i(\mathcal{L}_k), j)$. Furthermore, if j is the sink vertex z the dual variable for the active vehicle needs to be deducted. If $j \in V' \setminus \{s, z\}$, additionally the travel cost of each short haul vehicle used for the corresponding first-/last-mile transportations and the transshipment cost must be added, as well as the dual variable for each used short haul vehicle deducted. Finally, if j is a PTN (i.e. if a request is transshipped onto the vehicle) the dual variable of both the pickup node and any additional passive vehicle must be deducted from the cost of the new label. To determine the number of additional passive vehicles we use the function $\text{addIPV}(x, y)$, which returns the number of additional passive vehicles used in label x compared to label y . It is defined as

$$\text{addIPV}(x, y) = \max(g(x) - g(y), 0)$$

The departure time (t) of the new label is set according to rule (17). It is equal to the departure time of the old label plus the travel time of the vehicle along arc $(i(\mathcal{L}_k), j)$, plus the waiting time at j , plus the service time at j , plus the appropriate mandatory resting time. The appropriate amount of resting time depends on both the current time and the duration of the performed activity (i.e., (un)loading a request or traversing an arc). Therefore, the calculation of the new departure time needs to be done in two steps. First, the arrival time at node j is calculated. Then, given the arrival time, the start of the time window at j , and the service time, the departure time of the new label can be calculated. In both steps we use the function $\text{addTime}(x, y)$, which returns the amount of additional time needed to perform an activity with duration y , given the current time x . It takes the limit on the operation time of the vehicle into account and is defined as

$$\text{addTime}(x, y) = \begin{cases} \omega + \aleph + \left\lfloor \frac{y}{e_k} \right\rfloor \xi + \left(y - \left\lfloor \frac{y}{e_k} \right\rfloor e_k \right) & \text{iff } \omega < 0 \\ \omega + \aleph + \left\lfloor \frac{y-\omega}{e_k} \right\rfloor \xi + \left(y - \omega - \left\lfloor \frac{y-\omega}{e_k} \right\rfloor e_k \right) & \text{iff } 0 \leq \omega < y \\ y & \text{otherwise,} \end{cases}$$

where \aleph denotes the amount of time units per day the vehicle must rest, ξ denotes the total amount of time units per day, and ω is the residual time the vehicle may still work on the current day given by

$$\omega = e_k - ([x] \bmod \xi) - (x - [x])$$

Rule (18) updates the set of unreachable requests (\mathcal{U}) of the new label. It is equal to the union of the set of unreachable requests of the old label and the set of requests for which there is no time feasible extension from node j to any of their corresponding PTNs. Additionally, if all of the demand of the request which is associated with node j has already been picked up, then the set of unreachable requests of the new label also contains this request. Rules (19)–(22) update the load of the vehicle (l), the set of open requests (\mathcal{O}), and the amount of each request which has already been transshipped to/from the vehicle (d_r^+/d_r^-), respectively. Rule (23) updates the number of passive vehicles in use (g). If j is a PTN, the number of passive vehicles in use

is equal to the minimum number of passive vehicles required to carry the total load of the vehicle. If j is a DTN, the number of passive vehicles in use of the new label is equal to the number of passive vehicles in use of the old label. Furthermore, in this case, we create one additional label for each possible value of g between the minimum number of passive vehicles required and $g(\mathcal{L}_k)$. Each of those labels is identical to the new label except for the value of g . Creating those additional labels is necessary in order to cover all feasible active-passive-vehicle combinations with which the new label can be extended further.

The amount of request $r \in R$ transshipped to/from a long haul vehicle at node $i \in V'$ (i.e., λ_i) is determined as follows:

$$\lambda_i = \begin{cases} \min \left((d_{r(i)} - d_{r(i)}^+), (q(\mathcal{L}_k) + \min(u - g, \mathcal{W}_k - g)q_k - l(\mathcal{L}_k)) \right) & \text{iff } i \in \Pi^+ \\ d_{r(i)}^+ - d_{r(i)}^- & \text{iff } i \in \Pi^- \end{cases}$$

λ_i is determined in accordance with transshipment policy P1. That is, if i is a PTN the amount of the corresponding request transshipped to the long haul vehicle is equal to the minimum of the amount of the request which has not yet been transshipped to the vehicle and the remaining capacity of the vehicle. The remaining capacity of the vehicle is equal to the current capacity of the vehicle, plus a possible increase of the capacity through addition of passive vehicles, minus the current load on the vehicle. If i is a DTN, the amount of the corresponding request transshipped from the vehicle is equal to the amount of the request which is currently on the vehicle.

In order to discard dominated labels during the search we use the following dominance conditions. Label \mathcal{L}_k dominates label \mathcal{L}'_k if all of the following conditions hold:

$$\begin{aligned} \tau(\mathcal{L}_k) &= \tau(\mathcal{L}'_k), & \zeta(\mathcal{L}_k) &\leq \zeta(\mathcal{L}'_k), & t(\mathcal{L}_k) &\leq t(\mathcal{L}'_k), \\ \mathcal{O}(\mathcal{L}_k) &\subseteq \mathcal{O}(\mathcal{L}'_k), & \forall r \in R : d_r^+(\mathcal{L}_k) &\leq d_r^+(\mathcal{L}'_k), & \mathcal{U}(\mathcal{L}_k) &\subseteq \mathcal{U}(\mathcal{L}'_k), \text{ and} \\ g(\mathcal{L}_k) &= g(\mathcal{L}'_k). \end{aligned}$$

3.3 The column generation algorithm

As already mentioned above, we initialize the RMP with a set of columns which contains one column for each request $r \in R$, corresponding to direct full truck load trips from their pickup location r^+ to their delivery location r^- . Whenever the label-setting algorithm finds at least one column with negative reduced cost, we add them to the RMP and re-solve it in order to obtain new dual information before we solve the next subproblem. Furthermore, we do not solve the subproblems to optimality in each iteration of the column generation algorithm, but stop after a sufficient amount of columns has been found by the label-setting algorithm. When the label-setting algorithm cannot find new columns with negative reduced cost, the column generation algorithm stops and we have obtained a lower bound for the MMLHRP.

4 Matheuristic

With the column generation algorithm presented in the previous section we are able to compute lower bounds for small sized instances in a reasonable amount of time (see Sect. 5). For medium, large, or even real-world sized instances the algorithm is not applicable, though. Due to the large amount of labels generated by the label-setting algorithm, the column generation algorithm becomes impractical in terms of both run time and memory consumption. Therefore, we substitute the label-setting algorithm with an iterated local search (ILS) algorithm to solve the subproblem (i.e. to generate negative reduced cost columns). When using the ILS algorithm to solve the subproblem, the column generation algorithm is not exact any more.

Due to the fact that it is not guaranteed that the optimal solution of the MP will be found when using the ILS algorithm, we refrain from using a branch-and-price procedure for obtaining an integer solution. Instead, we use the columns generated by the column generation algorithm for the MP to solve the problem defined by (1)–(4).

4.1 Iterated local search algorithm

ILS is a metaheuristic generating a sequence of local optima. It is an iterative method which works on an incumbent solution y and considers different neighbourhoods of y in each iteration (for an elaborate description of ILS see Lourenço et al. (2010)). A solution, in our case, is a path in G' which starts at the source vertex s and ends at the sink vertex z . The pseudo code for the ILS is depicted in Algorithm 1.

Algorithm 1: ILS heuristic

```

1  $y = \text{path}(s, z)$ 
2  $NegCostPath = \emptyset$ 
3 repeat
4    $y' = \text{LocalSearch}(\text{Perturbation}(y))$ 
5   if  $\text{accept}(y', y)$  then
6      $y = y'$ 
7   end
8   if  $\text{cost}(y') < 0$  then
9      $NegCostPath = NegCostPath \cup \{y'\}$ 
10  end
11 until stopping criterion met;
12 return  $\{y \in NegCostPath : \text{cost}(y) \leq 0.9 \cdot \min_{x \in NegCostPath} \text{cost}(x)\}$ 

```

We initialize the algorithm with the path (s, z) in line 1. Line 4 is the interesting part of the algorithm. First, the (current) incumbent solution is perturbed and then local search is applied to the perturbed solution. Both the *Perturbation* phase and the *LocalSearch* phase consider the following two neighbourhoods:

1. *Insertion*—adding a new request to the path
2. *Removal*—removing a request from the path

Because the dual variables from the MP are associated with the PTNs, it is likely that some arcs entering a PTN have a negative cost. Therefore, both types of moves can yield cost reductions.

In the *Perturbation* phase the method performs n random moves with equal probability of a move being an insertion or a removal of a request ($n = 5$ in practice). In case of an insertion, we first randomly choose a request which is not yet fully served. Then, we randomly choose a corresponding PTN and a corresponding DTN. After this, we insert the PTN at a randomly chosen position in the path. We insert the DTN at the first feasible position in the path after the PTN. In case of a removal, we randomly choose a PTN in the path and remove it together with the corresponding DTN.

In the local search the method employs a steepest descent on both neighbourhoods, i.e. in each iteration we perform the move which brings the best improvement of the path cost, be it an insertion or a removal of a request. For this purpose we first calculate for all not yet fully served requests the change in the path cost incurred by inserting the corresponding PTN-DTN-pairs at every feasible position in the path. Then, we calculate for each request which is already served on the path (i.e. PTN-DTN-pair) the change in the path cost incurred by removing it from the path. Finally, we perform the move which yields the highest cost reduction. The local search stops when no improving neighbour can be found, i.e. when the change in the path cost of every feasible move is greater than or equal to zero.

In line 5 we determine if the new solution should be accepted. We use the so-called random walk as acceptance criterion, where a new local optimum is always accepted as a new incumbent solution. Furthermore, in each iteration we check if the new solution has negative reduced cost, and store it if it does (lines 8 and 9).

In line 11 we check if a stopping criterion is met. We apply the adaptive stopping criterion proposed by Cacchiani et al. (2014). It automatically adapts the number of iterations to the difficulty of the problem and works as follows: if a negative cost path has been found after n iterations, then the algorithm stops; else, the total number of iterations is increased by the factor ψ , in order to allocate more time to the algorithm. Additionally, it is necessary to define a total limit on the number of iterations n_{MAX} which can never be exceeded. We set $n = 10$, $\psi = 10$, and $n_{MAX} = 1000$ in our implementation.

At the end of the algorithm (line 12) all paths with reduced cost within 10% of the best solution found are returned to the RMP.

4.2 Hybrid label-setting/ILS algorithm

Additionally, we implemented a hybrid version of the algorithm which combines the label-setting algorithm and the ILS (ILS/Lbl) to solve the subproblem. It starts with ILS to solve each subproblem. If ILS does not find a negative reduced cost column for at least one subproblem, we switch to the label-setting algorithm. As soon as the

label-setting algorithm finds a new column we switch back to ILS. The algorithm stops when the label-setting algorithm cannot find new columns. Note that this is an exact algorithm.

5 Computational experiments

In this section, we summarize the computational experiments performed to assess both the performance of our solution method and the potential savings achievable through multimodal transport. The algorithm was implemented in C++ using CPLEX 12.7 as MIP-solver. All experiments were performed on an Intel Xeon 2.6 GHz CPU with a run time limit of 8 h. Because ILS is stochastic, we perform five runs whenever we use a subproblem algorithm involving ILS.

In the following three subsections we first present the instances which we use for the computational experiments (Sect. 5.1). In Sect. 5.2 we present computational results with respect to the performance of our solution method. Finally, in Sect. 5.3 we present managerial findings.

5.1 Test instances

We generated a set of test instances mimicking real-world long-distance transportation within and between countries along the Danube. For this purpose we created a set of customer locations and a set of transshipment locations. The set of customer locations contains 55 locations, where each location corresponds to a random address within the industrial area of a major city from the following six countries: Austria, Slovakia, Hungary, Romania, Serbia, and Bulgaria. Five locations were selected from Serbia and ten locations from each of the other countries. The set of transshipment locations contains the location of the train station in each of the selected cities as well as the locations of nine major ports along the Danube.

The pickup and the delivery location for each request are randomly selected from the set of customer locations. The demand of a request is randomly generated as well, with $d \sim U(20, 1500)$ and $d \in \mathbb{N}$. Each instance considers a time horizon of 30 days. The time windows of a request were randomly placed within this interval. The width of each time window is either 8, 72, or 120 h, which was chosen randomly with an equal probability for each width-size. We created ten sets of requests for each of the following size categories (= number of requests): 2, 5, 10, 30, 50, 100, and 200. In order to analyse if it is beneficial to consider more transshipment locations than just the closest one for each customer, we generated from each set of requests an instance which contains only the closest train station and the closest port for each customer location (named “closest”), and another instance which contains all 64 transshipment locations (named “all”).

We consider the following vehicle types in our instances: truck, ship, push craft, barge, train, and single wagon. Trucks represent the short haul vehicles, and the rest represent the long haul vehicles. We consider ships, push crafts, trains, and single

Table 2 Parameters used in test instances

Vehicle type	Travel cost (€)	Capacity (tons)	Max. operating time per day (h)
Truck	0.73/km	20	10
Train	6.95/km	1300	24
Wagon	0.32/km	50	24
Ship	43.76/h	1000	14
Push craft	35.67/h	0	24
Barge	10.45/h	1000	24

wagons as active vehicles, and barges as passive vehicles. However, we only allow push crafts to push barges. Furthermore, push crafts are special vehicles in the sense that they have no capacity ($q_k = 0$) and are only used to push barges. In the real world it is often possible that a logistics service provider attaches a single wagon to someone else's train which travels to the desired destination train station. This is why we consider single wagons in addition to complete trains. Table 2 summarizes the parameters used for each vehicle type. All costs and parameters were determined based on real world data in collaboration with an Austrian logistics service provider.

We used a travel cost of 0.73 €/km and a capacity of 20 tons for a truck in our experiments. The daily maximum operation time was assumed to be 10 h, which corresponds to the current labour regulations for commercial drivers in Austria. The distance- and travel time-matrices were calculated with the software MS MapPoint 2013. We used the following average velocities:

- Motorway: 65 km/h
- Highway: 60 km/h
- City: 20 km/h

The duration for a complete (un)loading of a truck was set to 24 min (= 50 tons/h). For both the train and the wagon we assumed a daily maximum operation time of all of the 24 h. The capacity of a wagon was set to 50 tons. An Austrian logistics service provider advised us to assume a length of 26 wagons for a train, as this was the most common length in their experience. Thus, we set the capacity of a train to 1300 tons (= 26 wagons). We used travel costs of 6.95 and 0.32 €/km for a train and a wagon, respectively. The distance matrix was constructed by using the online train-kilometre calculator <http://jizdenka.idos.cz>. In practice, freight trains are often time constrained due to the higher priority accorded to passenger trains. In order to avoid an over-estimation of the rail capacity, or respectively, an under-estimation of the delivery time we assumed a very low average velocity for trains of just 15 km/h, which we then used to create the travel time matrix. For a ship we assumed a daily maximum operation time of 14 h, because freight ships on the Danube typically have only one captain. For the push craft (and barge) we assumed a limit of 24 h, because they usually have two captains and can therefore be operated around the clock. The capacity of both

a ship and a barge was set to 1000 tons, while a push craft has no capacity. We set the maximum number of barges a push craft may push at the same time to six, which corresponds to the smallest maximum number of barges a push craft can push along all segments of the Danube considered in our test instances. It is customary to express travel cost for water vehicles in monetary units per time unit (as opposed to per distance unit). Thus, we assumed travel cost for ships, push crafts, and barges of 43.76, 35.67, and 10.45 €/h, respectively. The travel time matrices were constructed by using the online travel time calculator <http://www.danube-logistics.info/travel-time-calculator/>. We assumed transshipment cost of 3 €/ton at all ports and train stations, and an (un)loading speed of 125 tons/h for all long haul vehicles.

From each of the above described instances we created 11 versions which differ with respect to the availability of long haul vehicles:

1. no long haul vehicles available (named “N”)
2. only one ship available (named “S”)
3. only one push craft with six barges available (named “P”)
4. only one train available (named “T”)
5. only 26 wagons available (named “G”)
6. the available capacity of water-bound long haul vehicles equals 30% of the total demand, no rail-bound long haul vehicles available (named “W30”)
7. the available capacity of rail-bound long haul vehicles equals 30% of the total demand, no water-bound long haul vehicles available (named “R30”)
8. the available capacity of both water-bound and rail-bound long haul vehicles each equals 30% of the total demand (named “WR30”)
9. no restriction on the availability of water-bound long haul vehicles, no rail-bound long haul vehicles available (named “W”)
10. no restriction on the availability of rail-bound long haul vehicles, no water-bound long haul vehicles available (named “R”)
11. no restriction on the availability of long haul vehicles (named “∞”)

A sufficient number of short haul vehicles is always available in each instance in order to guarantee feasibility. In total we created 1,540 instances. We use the following naming convention for the instances: $a - b - c - d$

- $a \in \{2, 5, 10, 30, 50, 100, 200\}$: size category
- $b \in \{0, 1, \dots, 9\}$: set of requests
- $c \in \{\text{closest, all}\}$: available transshipment locations
- $d \in \{N, S, P, T, G, W30, R30, WR30, W, R, \infty\}$: available long haul vehicles

The wild card “*” refers to all instances in a category, while “[!x]” refers to all instances in a category except for “x”, e.g. “2-*-all-[!N]” refers to all instances of size 2, where all transshipment locations are present, and long haul vehicles are available.

All instances are available at: <http://plis.univie.ac.at/research/test-instances/>

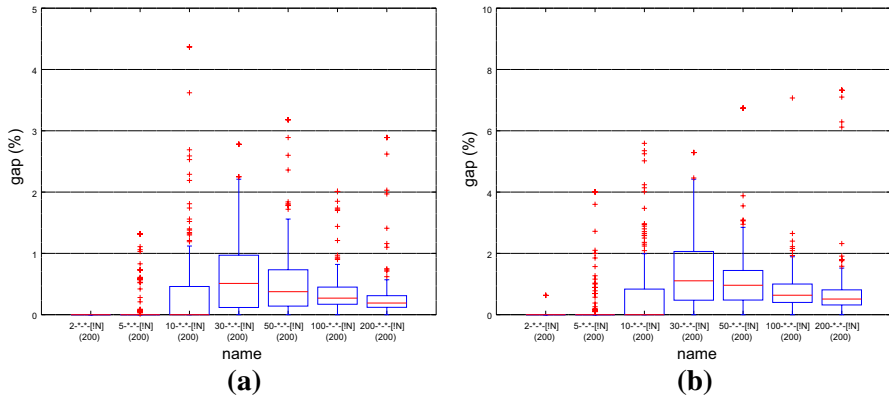


Fig. 4 Boxplots of the percentage gaps (y-axis) between the objective function value of the best run and the worst run when ILS is used as the subproblem algorithm; one boxplot for each instance size (x-axis). The numbers in brackets represent the number of instances per instance size

5.2 Computational results

In the first set of experiments we evaluate the stability of the ILS. For that purpose we calculate for each instance the percentage gap between the objective function value of the best run and the worst run. Figure 4 shows boxplots of the percentage gaps regarding the MP solutions (Fig. 4a) and the integer solutions (Fig. 4b) for each instance size. The number in brackets under the instance names represents the total number of instances in this size category. In order to obtain a better visual representation of the boxplots we truncated the y-axis at 5 and 10%, respectively. This cut off three outliers of Fig. 4a (max. outlier = 8.34%) and ten outliers of Fig. 4b (max. outlier = 21.13%). The ILS appears to be quite stable. For more than 90% of the instances the gap between the best and the worst run is less than 1% regarding MP solutions and less than 2% regarding integer solutions. Using the hybrid ILS/Lbl algorithm decreases the gaps even further. However, due to the involvement of the label-setting algorithm we could only solve instances with up to ten requests. Because the gap is equal to 0% for more than 75% of the instances regarding both MP solutions and integer solutions, we do

Table 3 Stability of the ILS/Lbl algorithm: MP solutions

Name	gap (%)		
	Min	Avg	Max
2-***-[!N] (200)	0.00	0.00	0.00
5-***-[!N] (200)	0.00	0.00	0.23
10-***-[!N] (200)	0.00	0.12	4.86

Columns *Min*, *Avg*, and *Max* give the minimum, average, and maximum percentage gap between the objective value of the best run and the worst run regarding MP solutions. The number in brackets next to the name represents the number of instances of the respective size

Table 4 Stability of the ILS/Lbl algorithm: integer solutions

Name	Gap (%)		
	Min	Avg	Max
2-***-[!N] (200)	0.00	0.01	2.25
5-***-[!N] (200)	0.00	0.02	1.62
10-***-[!N] (200)	0.00	0.29	6.39

Columns *Min*, *Avg*, and *Max* give the minimum, average, and maximum percentage gap between the objective value of the best run and the worst run regarding integer solutions. The number in brackets next to the name represents the number of instances of the respective size

not present the data as boxplots but only report the minimum, average, and maximum gap for each instance size in Tables 3 and 4, respectively.

In the following, whenever results involving ILS are reported they correspond to the best of five runs regarding the objective function value, and to the worst of five runs regarding the run time. In the practical case computers have multiple cores, which means that multiple runs can be performed in parallel. Thus, we believe it reasonable to use the solution with the best objective function value, since this is obviously the solution a practitioner would implement. Furthermore, we believe it is reasonable to use the slowest run time instead of the actual run time of the run which produced the best solution, because it is only possible to determine the best solution after the slowest run has finished.

Now we evaluate the performance of the subproblem algorithms. For that purpose we solve only the linear relaxation of the problem (i.e., the MP) and then compare both the computing times and the quality of the (approximation of the) lower bound. Table 5 shows, for each subproblem algorithm, the average computing times and solution quality. The first column gives the instance names (and the number of instances in brackets). Columns *CPU* give the average computing

Table 5 Comparison: computing times and solution quality

Name	Lbl-setting CPU	ILS				ILS/Lbl			
		CPU	Gap (%)			CPU	Gap (%)		
			Min	Avg	Max		Min	Avg	Max
2-***-[!N] (200)	0.2	0.1	0.0	0.0	7.9	0.1	0.0	0.0	0.0
5-***-[!N] (200)	11.7	0.3	0.0	0.0	4.8	6.9	0.0	0.0	0.0
10-***-[!N] (200)	3798.9*	3.5	- 11.1	- 0.2	3.2	2675.0+	- 11.1	- 0.4	0.0

Columns *CPU* give the average computing time in seconds to solve an instance; Columns *Min*, *Avg*, and *Max* give the minimum, average, and maximum percentage gap between the objective value obtained when using the ILS (or ILS/Lbl) and the objective value obtained when using the label-setting algorithm. The number in brackets next to the name represents the number of instances of the respective size

*19 Instances could not be solved within the run time limit (8 h)

+14 Instances could not be solved within the run time limit (8 h)

Table 6 Label-setting: impact time window width and period length

Name	30 day period			60 day period		
	8 h	120 h	240 h	8 h	120 h	240 h
2-***-∞ (20)	0.3	0.4	0.4	0.3	0.4	0.4
5-***-∞ (20)	6.9	21.4	40.1	22.5	23.0	43.7
10-***-∞ (20)	242.9	3139.0*	8784.3 ⁺	1554.0	3891.3	10665.9 ⁺

Columns *30 day period* and *60 day period* give the average computing time in seconds to solve the MP for instances with a period length of 30 and 60 days, respectively. Columns *8 h*, *120 h*, and *240 h* refer to instances where each time window has a width of 8, 120 and 240 h, respectively. The number in brackets next to the name represents the number of instances of the respective size

*1 Instance could not be solved within the run time limit (8 h)

⁺4 Instances could not be solved within the run time limit (8 h)

Table 7 ILS: impact time window width and period length

Name	30 day time horizon			60 day time horizon		
	8 h	120 h	240 h	8 h	120 h	240 h
2-***-∞ (20)	0.1	0.1	0.1	0.1	0.1	0.2
5-***-∞ (20)	0.4	0.7	1.1	1.2	1.5	2.1
10-***-∞ (20)	2.1	5.7	8.8	10.6	12.4	19.2
30-***-∞ (20)	48.3	128.4	200.2	244.5	388.9	518.2
50-***-∞ (20)	259.4	649.8	944.0	1337.9	2056.3	2874.6
100-***-∞ (20)	1944.8	4503.6	6985.7	11647.1	17086.7	22404.0*
200-***-∞ (20)	9449.7	25268.5	28800.0 ⁺	28800.0 ⁺	28800.0 ⁺	28800.0 ⁺

Columns *30 day period* and *60 day period* give the average computing time in seconds to solve the MP for instances with a period length of 30 and 60 days, respectively. Columns *8 h*, *120 h*, and *240 h* refer to instances where each time window has a width of 8, 120 and 240 h, respectively. The number in brackets next to the name represents the number of instances of the respective size

*2 Instances could not be solved within the run time limit (8 h)

⁺20 Instances could not be solved within the run time limit (8 h)

time in seconds to solve an instance using the label-setting algorithm, the ILS, or the ILS/Lbl combination, respectively. Columns *Min*, *Avg*, and *Max* give the minimum, average, and maximum percentage gap between the objective value obtained when using the ILS (ILS/Lbl combination) and the objective value obtained when using the label-setting algorithm. The ILS performs well in terms of both run time and solution quality. For all instance sizes the average gap is close to 0% and the average computation time is only a fraction of the computation time of the label-setting algorithm. The negative gap in the last row is explained by the fact that the label-setting algorithm could not solve 19 of the instances of size 10 within the run time limit. The ILS and the ILS/Lbl combination found better solutions for 13, respectively 15, of those instances.

Table 8 ILS/Lbl-setting: impact time window width and period length

Name	30 day period			60 day period		
	8 h	120 h	240 h	8 h	120 h	240 h
2-*** ∞ (20)	0.2	0.3	0.3	0.2	0.3	0.4
5-*** ∞ (20)	2.8	6.0	12.5	8.0	9.7	16.1
10-*** ∞ (20)	123.0	1329.6	6299.2*	601.8	1593.3	6977.2 ⁺

Columns *30 day period* and *60 day period* give the average computing time in seconds to solve the MP for instances with a period length of 30 and 60 days, respectively. Columns *8 h*, *120 h* and *240 h* refer to instances where each time window has a width of 8, 120 and 240 h, respectively. The number in brackets next to the name represents the number of instances of the respective size

*3 Instances could not be solved within the run time limit (8 h)

⁺1 Instance could not be solved within the run time limit (8 h)

We now evaluate the impact of the width of the time windows and the length of the time horizon on the performance of the subproblem algorithms. For that purpose we created new instances based on the ∞ -instances. We kept everything the same except the width of the time windows, the position of the time windows, and the length of the time horizon. In each new instance all time windows have the same length. In total we created six new versions from each ∞ -instance: one version where the width of the time windows is equal to 8, 120, and 240 h, respectively; combined with a length of the time horizon of 30 and 60 days. We again solve only the MP and then compare the computing times. Tables 6, 7, and 8 show, for each subproblem algorithm, the average computing times in seconds. Columns *30 day period* and *60 day period* give the average computing time to solve the MP for instances with a time horizon of 30 and 60 days, respectively. Columns *8 h*, *120 h*, and *240 h* refer to instances where each time window has a width of 8, 120 and 240 h, respectively. Wider time windows and longer time horizons correlate with larger solutions spaces. Therefore, it is not surprising that both the width of the time windows and the length of the time horizon have a considerable impact on the performance of all three subproblem algorithms. What is interesting, however, is that the width of the time windows has a noticeably larger impact on the label-setting algorithm than on the ILS, while for the length of the time horizon it is the other way around. An increase of the time window width from 8 to 120 h, from 120 to 240 h, and from 8 to 240 h, results in 3.8, 1.9, and 9.0 times longer computation times on average for the label-setting algorithm. For the ILS the computation times increase by a factor of only 1.8, 1.4, and 2.5 on average, respectively, for the same increases of the time window width. The computation times of the hybrid ILS/Lbl-setting algorithm increase by a factor of 3.3, 2.5, and 12.1 on average, respectively. Increasing the time horizon from 30 to 60 days almost doubles the run time on average for the label-setting algorithm and the ILS/Lbl-setting algorithm, while it more than doubles the run time on average for the ILS. If we consider all seven size categories of the instances (not just the smallest three), then the run time of the ILS almost triples on average.

Table 9 Comparison: “closest”–“all”

Name	Gap “closest” vs. “all” (%)		#requests non-closest		Run time factor		
	Avg	Max	Avg	Max	Min	Avg	Max
	2-***-[!N] (100)	0.03	0.63	0.1	1	0.0	13.2
5-***-[!N] (100)	0.07	6.68	0.2	1	0.0	13.3	59.5
10-***-[!N] (100)	0.22	7.57	0.4	3	0.1	5.9	18.9
30-***-[!N] (100)	0.19	2.95	0.7	5	0.2	2.1	34.8
50-***-[!N] (100)	0.04	1.27	0.4	5	0.2	1.4	6.8
100-***-[!N] (100)	- 0.06	0.98	0.3	7	0.0	3.0	50.9
200-***-[!N] (100)	0.01	0.80	0.5	5	0.0	2.6	52.1

Average and maximum percentage gap between the objective function values when only the closest transshipment locations are considered and when all transshipment locations are considered. Column *#requests non-closest* gives the average and maximum number of requests which are transshipped at a location which is not the closest one. Column *run time factor* gives the factor by which the run time increases (at the minimum, on average, and at the maximum) when all transshipment locations are considered. The number in brackets next to the name represents the number of instances of the respective size

In the following, all analyses compare only integer solutions and are based on results obtained with ILS as the subproblem algorithm.

5.3 Managerial discussion

Now we address the question of whether it can be beneficial to consider more potential transshipment locations than just the closest one of each customer. For that purpose we compare the objective function values of the solutions of the “closest” instances with the “all” instances. Table 9 gives the average and maximum percentage gap between those solutions, whereby the solution values of the “closest” instances are the reference values. For all instance sizes, the solutions where all transshipment locations are considered are on average less than 0.25% better. The maximum difference of ~7.5% is achieved at an instance of size 10 and decreases to less than 1% for larger instances. However, this is to be expected as the number of additional ‘non-closest’ transshipment locations decreases with an increasing number of requests. Table 9 also shows that on average the number of requests transshipped at a location which is not the closest one is less than one. Additionally, Table 9 shows that the moderate decrease in costs is achieved through a comparatively large increase in computation time. It takes on average 2–3 times longer to solve larger instances which contain all transshipment locations.

The negative value of the average percentage gap for instances of size 100 can be explained as follows. In general, given two sets of columns, A and B, it is possible that set A yields a better LP solution than set B, but a worse integer solution than set B. This is what happens here. For some “all” instances, the column generation

Table 10 Average delivery time increase: unimodal–multimodal

Name	Delivery time increase (%)						
	d = S	d = P	d = T	d = G	d = W	d = R	d = ∞
2-***-d	150.7	87.8	82.3	71.3	148.4	79.3	79.3
5-***-d	118.3	111.4	105.7	119.6	123.1	120.7	119.0
10-***-d	96.4	54.3	54.9	59.2	76.5	59.5	59.5
30-***-d	61.9	59.1	46.6	55.5	58.5	48.5	53.9
50-***-d	41.6	35.2	26.7	38.5	44.4	42.0	45.1
100-***-d	58.2	48.0	28.7	42.3	53.9	46.1	50.1
200-***-d	75.8	52.0	41.1	54.1	63.5	53.7	49.4
Average	86.1	64.0	55.2	62.9	81.2	64.2	65.2

Columns 2–8 give the average percentage delivery time increase of a request for selected multimodal cases compared to the unimodal case (a-***-N), respectively

approach produces a set of columns that yields a better LP solution than the set of columns produced for the corresponding “closest” instance, but a worse integer solution.

We now evaluate the increase in the delivery time of a request when it is transported multimodally compared to unimodally. For that purpose, we compare for each request its delivery times in the solutions of the different vehicle availabilities. Table 10 shows the average percentage increase of the delivery time of a request for selected multimodal cases compared to the unimodal case. In order to be able to make comparisons between the different vehicle types, Table 10 only includes requests which are transported by a long haul vehicle in every multimodal case. The delivery time of requests which are transported by a ship increases by roughly 86% on average compared to when they are delivered by a direct unimodal transport. Using a push craft the delivery time increases only by 64% on average. The smaller

Table 11 Average cost reduction: unimodal–multimodal

Name	Cost reduction (%)									
	d = S	d = P	d = T	d = G	d = W30	d = R30	d = WR30	d = W	d = R	d = ∞
2-***-d (20)	12.8	17.7	26.4	37.0	17.9	33.7	33.7	17.9	40.2	40.2
5-***-d (20)	10.5	16.1	32.9	42.9	15.4	42.7	42.9	17.2	49.7	49.7
10-***-d (20)	10.1	19.2	30.8	38.3	18.9	42.3	42.6	20.7	46.5	46.5
30-***-d (20)	7.6	18.3	23.0	28.8	23.1	48.3	49.1	27.2	50.7	51.0
50-***-d (20)	4.9	14.2	16.2	21.1	20.6	48.6	49.0	24.8	50.8	50.9
100-***-d (20)	3.3	10.3	9.9	12.7	22.6	49.4	49.7	25.9	50.7	50.8
200-***-d (20)	1.9	6.5	5.8*	7.2	22.5	50.2	50.6	25.5	50.9	51.2
Average	7.3	14.6	20.7	26.8	20.1	45.0	45.4	22.7	48.5	48.6

Columns 2–11 give the average percentage cost reduction for each of the ten multimodal cases compared to the unimodal case (a-***-N), respectively. The number in brackets next to the name represents the number of instances of the respective size

*1 Instance could not be solved within the run time limit (8 h)

increase of the delivery time by a push craft is because it can operate 24 h per day as opposed to the ship which can only operate for 14 h per day. Looking at trains and single wagons, the table shows that both cause an even smaller average increase in delivery time; namely ~55 and ~62%. In general, water transportation causes an average increase in the delivery time roughly 1.3 times larger than rail transportation (see “W” vs. “R”).

We now want to determine the potential cost savings through multimodal transport. For that purpose we compare, for each set of requests, the objective function values of the solutions of the different vehicle availabilities. Table 11 presents the average percentage cost reduction for each of the ten multimodal cases compared to the unimodal case (*-**-N). Using a single ship leads to a cost reduction of roughly 7% on average. With a single push craft (and six barges) the cost reduction can be doubled, which is, of course, a direct consequence of the larger capacity. Both a single train and 26 wagons achieve an even larger decrease in costs, with savings of ~20 and ~26%, respectively.

Furthermore, rail transportation achieves average cost reductions more than twice as high as water transportation (see “W30” vs. “R30” and “W” vs. “R”). This large difference in potential cost savings is partially due to time restrictions: water transportation is slower than rail transportation (see Table 10). Also, for most customer locations the closest port is further away than the closest train station, which increases the travel time in the first- and last-mile. Therefore, because of time windows, some requests cannot be transported by water, but can be transported by rail. Furthermore, the larger distance to the next port not only increases the travel time of the short haul, but also its cost. Thus, for some requests it is not worthwhile to transport them by water, due to the high short haul costs, while it is worthwhile with rail transportation. On average, 98.1% of requests (or 98.4% of total demand) may be transported by rail in each of our instances, while only 71.7% of requests (or 71.3% of total demand) may be transported by water. The remaining 1.9% (1.6%), respectively 28.3% (28.7%), can or should not be transported by rail, respectively water, because of time restrictions or too large short haul costs.

Table 11 shows further that when there is no restriction on the amount of available long haul vehicles, cost reductions of more than 50% can be achieved. However, almost identical cost savings can be achieved when there is no restriction on

Table 12 Transportation mode and long haul vehicle type split

Name	Proportion of total demand transported (%)							
	Water	Rail	Road	Ship	Push craft	Train	Wagon	Truck
*-**-R30 (140)	0.0	71.5	28.5	0.0	0.0	35.3	36.2	28.5
*-**-WR30 (140)	3.4	68.5	28.1	0.9	2.4	34.1	34.4	28.1
*-**-R (140)	0.0	77.5	22.5	0.0	0.0	42.9	34.6	22.5
*-**-∞ (140)	2.4	75.1	22.5	0.6	1.8	41.0	34.1	22.5

Columns 2–9 give the average proportion of total demand transported by the respective mode of transportation, and type of long haul vehicle, in the respective long haul vehicle availability setting

Table 13 Average cost reduction (without wagons): unimodal–multimodal

Name	Cost reduction (%)								
	d = S	d = P	d = T	d = W30	d = R30	d = WR30	d = W	d = R	d = ∞
2-***-d (20)	12.8	17.7	26.4	17.9	28.3	28.4	17.9	28.3	28.4
5-***-d (20)	10.5	16.1	32.9	15.4	38.0	38.2	17.2	40.8	40.5
10-***-d (20)	10.1	19.2	30.8	18.9	35.3	36.8	20.7	39.4	39.4
30-***-d (20)	7.6	18.3	23.0	23.1	43.3	44.7	27.2	45.3	45.9
50-***-d (20)	4.9	14.2	16.2	20.6	42.8	44.1	24.8	45.2	45.6
100-***-d (20)	3.3	10.3	9.9	22.6	43.5	44.7	25.9	45.8	46.2
200-***-d (20)	1.9	6.5	5.8*	22.5	44.4	45.7	25.5	46.2	46.7
Average	7.3	14.6	20.7	20.1	39.4	40.4	22.7	41.6	41.8

Columns 2–10 give the average percentage cost reduction for each of the nine multimodal cases compared to the unimodal case (a-***-N), respectively

*1 Instance could not be solved within the run time limit (8 h)

Table 14 Transportation mode and long haul vehicle type split (without wagons)

Name	Proportion of total demand transported (%)							
	Water	Rail	Road	Ship	Push craft	Train	Wagon	Truck
***-R30 (140)	0.0	66.4	33.6	0.0	0.0	66.4	0.0	33.6
***-WR30 (140)	7.3	61.1	31.6	2.4	4.9	61.1	0.0	31.6
***-R (140)	0.0	71.9	28.1	0.0	0.0	71.9	0.0	28.1
***-∞ (140)	5.6	66.3	28.1	1.5	4.1	66.3	0.0	28.1

Columns 2–9 give the average proportion of the total demand transported by the respective mode of transportation, or type of long haul vehicle, in the respective long haul vehicle availability setting

the availability of rail-bound long haul vehicles (see “R”). The same conformity in cost reductions can be observed between “R30” and “WR30”. Water transportation, it seems, is only of interest if rail transportation is not possible. Table 12 provides further insights on this issue. It shows, for cases “R30”, “WR30”, “R”, and “∞”, the proportion of the total demand which is transported on each mode of transportation. Rail transportation accounts, on average, for more than two-thirds of the total demand transported on the long haul, while water transportation amounts to only 2.4–3.4%. In fact, facilitating water transportation in addition to rail transportation leads to only a slight increase (“R30” vs. “WR30”) or even no increase at all (“R” vs. “∞”) of the amount of demand transported in a multimodal fashion.

Tables 11 and 12 additionally show that wagons are heavily used and have a large impact on costs in the above presented solutions. Using wagons in the way we propose relies on a strong assumption. Attaching a single wagon to someone else’s train which travels to the desired destination train station might in fact not be possible in some real world cases, at least not all the time. Therefore, we reran

all experiments again without the possibility of using wagons. Table 13 presents again the average percentage cost reduction for the multimodal cases compared to the unimodal case, but without the usage of wagons in cases “R30”, “WR30”, “R”, and “ ∞ ”. The average cost reductions decrease by roughly five percentage points in all four cases. The level of conformity in cost reductions between “R30” and “WR30”, as well as between “R” and “ ∞ ”, stays the same, though. Regarding the proportions of total demand transported on each mode of transportation, Table 14 shows that there is only a slight shift towards water transportation. The majority of the demand previously transported by wagons is now covered by trains. Nonetheless, this shift constitutes a doubling of water transportation. Approximately 5% of total demand cannot be absorbed by any other long haul vehicle and is, therefore, transported by road.

6 Conclusion

In this article we present a new problem called the multimodal long haul routing problem (MMLHRP). We propose both an arc flow formulation and a set-covering formulation for this problem. For the set-covering formulation we further propose a column generation algorithm to solve its linear relaxation, i.e., to obtain lower bounds. We solve the subproblem in this approach with both a label-setting algorithm and an ILS. In order to obtain feasible integer solutions we propose a matheuristic which is based on the column generation framework. Computational experiments show that the ILS as the subproblem algorithm performs well compared to the label-setting algorithm. Indeed, using ILS as the subproblem algorithm made the MMLHRP tractable for large instances.

Additional computational experiments show that it is not beneficial to consider more transshipment locations than just the closest one of a customer. Furthermore, we find that considerable cost reductions can be achieved through multimodal transport compared to unimodal transport. In our computational studies, cost reductions of more than 50% could be achieved in the most extreme cases, depending on both the number of requests and the number of available long haul vehicles. The experiments further show that water transportation hardly contributes in terms of both transported demand and cost reductions if rail transportation is available as well. In general, rail transportation achieves average cost reductions more than twice as high as water transportation.

Further research regarding the investigated problem may analyse the cost of the introduced policies and assumptions which aim at generating more practical solutions. Allowing a unit of load to use several long haul vehicles in succession is likely to further reduce overall costs. The same is true for not regulating the number of splits. Indeed, the solutions will probably not be as practical, but the trade-off might be worth it. Another interesting research direction would be to consider an additional “green” objective, such as minimizing CO_2 emissions. Especially in multimodal vehicle routing, minimizing costs and minimizing emissions of pollutants are conflicting objectives. For example, trucks typically have significantly higher routing costs than ships but are at the same time significantly less polluting than ships.

Acknowledgements Open access funding provided by University of Vienna. The financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged. We thank Michael Wasner for bringing this problem to our attention and for his valuable real-world input. We would also like to thank the three anonymous referees for their helpful comments.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

Appendix 1: Arc flow formulation

The MMLHRP can be modelled by means of an arc flow formulation using the following variables:

- s_{ik} Start time of service of vehicle $k \in K$ at node $i \in V$
- z_{ik} Number of rests of vehicle $k \in K$ until node $i \in V$
- v_{ik} Sum of travel- and service times of vehicle $k \in K$ until node $i \in V$
- f_{ijrk} Amount of request $r \in R$ transported by vehicle $k \in K$ over arc $(i, j) \in A$
- $w_{irkk'}$ Amount of request $r \in R$ transshipped from vehicle $k \in K$ to vehicle $k' \in K$ at node $i \in \bar{T}$
- σ_{ik} Service time of vehicle $k \in K$ at node $i \in V$
- $x_{ijk} \begin{cases} 1 & \text{iff vehicle } k \in K \text{ traverses arc } (i, j) \in A \\ 0 & \text{otherwise} \end{cases}$
- $y_{irkk'} \begin{cases} 1 & \text{iff request } r \in R \text{ is transshipped from vehicle } k \in K \\ & \text{to vehicle } k' \in K \text{ at node } i \in \bar{T} \\ 0 & \text{otherwise} \end{cases}$
- $\alpha_{ijkk'} \begin{cases} 1 & \text{iff active vehicle } k \in C \text{ pushes passive vehicle } k' \in E \text{ over arc } (i, j) \in A \\ 0 & \text{otherwise} \end{cases}$

Tables 15 and 16 provide a summary of the used notation.

For modelling purposes we define for variables x, f , and α :

$$x_k(F) \equiv \sum_{(i,j) \in F} x_{ijk}, f_{rk}(F) \equiv \sum_{(i,j) \in F} f_{ijrk}, \text{ and } \alpha_{kk'}(F) \equiv \sum_{(i,j) \in F} \alpha_{ijkk'}$$

With the above notation the MMLHRP can be formulated as follows:

$$\min \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} c_{ijk} x_{ijk} + \sum_{i \in \bar{T}} \sum_{r \in R} \sum_{k \in K} \sum_{k' \in K} m_i w_{irkk'} \tag{24}$$

subject to

$$\sum_{k \in K} f_{rk}(\delta_k^+(r^+)) = \sum_{k \in K} f_{rk}(\delta_k^-(r^-)) = d_r \quad \forall r \in R \tag{25}$$

Table 15 Sets for the arc flow formulation

R	Set of requests	A^k	Feasible movements of vehicle $k \in K$ between the different nodes in V
P	Set of pickup nodes ($P = \bigcup_{r \in R} r^+$)	A	Arc set ($A = \bigcup_{k \in K} A^k$)
D	Set of delivery nodes ($D = \bigcup_{r \in R} r^-$)	B	Set of short haul vehicles
T	Set of transshipment nodes (physical)	C	Set of active vehicles
$T'(i)$	Set of node duplicates of transship. node $i \in T$	E	Set of passive vehicles
\bar{T}	$\bigcup_{i \in T} T'(i)$	K	$B \cup C \cup E$
N	$P \cup D \cup \bar{T}$	$\delta_k^+(S)$	$\{(i, j) \in A^k i \in S, j \notin S\}$: set of arcs $a \in A^k$ leaving set $S \subseteq V$
V	$N \cup \{\bigcup_{k \in K} o_k, \bigcup_{k \in K} \eta_k\}$, where o_k and η_k are the start- and end-depots, respectively	$\delta_k^-(S)$	$\{(i, j) \in A^k i \notin S, j \in S\}$: set of arcs $a \in A^k$ entering set $S \subseteq V$

Table 16 Parameters for the arc flow formulation

a_i	Start of time window at node $i \in V$	d_r	Load of request $r \in R$
b_i	End of time window at node $i \in V$	q_k	Capacity of vehicle $k \in K$
c_{ijk}	Cost of traversing arc $(i, j) \in A^k$ for vehicle $k \in K$	r^+	Pickup node of request $r \in R$
e_k	Amount of hours vehicle $k \in K$ is allowed to operate per day	r^-	Delivery node of request $r \in R$
g_k	Amount of hours vehicle $k \in K$ must rest per day	t_{ijk}	Travel time over arc $(i, j) \in A^k$ of vehicle $k \in K$
m_i	Transshipment cost per weight unit at transshipment node $i \in \bar{T}$	u	Maximum number of passive vehicles allowed per active vehicle
h_{ik}	Loading time (time units per weight unit) at node $i \in N$ of vehicle $k \in K$		

$$x_k(\delta_k^+(o_k)) = x_k(\delta_k^-(\eta_k)) = 1 \quad \forall k \in K \setminus B \tag{26}$$

$$x_k(\delta_k^+(i)) - x_k(\delta_k^-(i)) = 0 \quad \forall i \in \bar{T}, \quad \forall k \in K \setminus B \tag{27}$$

$$\sum_{k \in K} f_{rk}(\delta_k^+(i)) - \sum_{k \in K} f_{rk}(\delta_k^-(i)) = 0 \quad \forall i \in \bar{T}, \forall r \in R \tag{28}$$

$$v_{jk} \geq v_{ik} + \sigma_{ik} + t_{ijk} - M(1 - x_{ijk}) \quad \forall (i, j) \in A^k, \forall k \in K \tag{29}$$

$$z_{ik} \geq \frac{v_{ik}}{e_k} \quad \forall i \in V, \forall k \in K \tag{30}$$

$$s_{ik} \geq v_{ik} + (g_k z_{ik}) \quad \forall i \in V, \forall k \in K \tag{31}$$

$$s_{jk} \geq s_{ik} + t_{ijk} + (g_k z_{jk} - g_k z_{ik}) + \sigma_{ik} - M(1 - x_{ijk}) \quad \forall (i, j) \in A^k, \forall k \in K \quad (32)$$

$$\sum_{r \in R} f_{ijrk} \leq q_k x_{ijk} \quad \forall (i, j) \in A^k, \forall k \in K \quad (33)$$

$$s_{ik} + \sigma_{ik} - M(1 - y_{irkk'}) \leq s_{ik'} \quad \forall i \in \bar{T}, \forall r \in R, \forall k, k' \in K \quad (34)$$

$$\sum_{r \in R} w_{irkk'} \leq \min\{q_k, q'_k\} \cdot y_{irkk'} \quad \forall i \in \bar{T}, \forall k, k' \in K \quad (35)$$

$$f_{rk}(\delta_k^+(i)) = f_{rk}(\delta_k^-(i)) + \sum_{k' \in K} w_{irk'k} - \sum_{k' \in K} w_{irkk'} \quad \forall i \in \bar{T}, \forall r \in R, \forall k \in K \quad (36)$$

$$\sigma_{ik} \geq \sum_{r \in R} \sum_{k' \in K} (w_{irkk'} + w_{irk'k}) h_{ik} \quad \forall i \in \bar{T}, \forall k \in K \quad (37)$$

$$\sigma_{r^+,k} \geq f_{rk}(\delta_k^+(r^+)) h_{r^+,k} \quad \forall r \in R, \forall k \in B \quad (38)$$

$$\sigma_{r^-,k} \geq f_{rk}(\delta_k^-(r^-)) h_{r^-,k} \quad \forall r \in R, \forall k \in B \quad (39)$$

$$a_i \leq s_{ik} \leq b_i \quad \forall i \in P \cup D \cup \{o_k, \eta_k\}, \forall k \in K \quad (40)$$

$$x_{ijk} = \sum_{k' \in C} \alpha_{ijkk'} \quad \forall (i, j) \in A^{k'}, \forall k' \in E \quad (41)$$

$$\sum_{k' \in E} \alpha_{ijkk'} \leq x_{ijk} u \quad \forall (i, j) \in A^k, \forall k \in C \quad (42)$$

$$s_{jk} - s_{jp} \leq M(1 - \alpha_{ijkk'}) \quad \forall (i, j) \in A^k, \forall k \in C, \forall k' \in E \quad (43)$$

$$s_{jp} - s_{jk} \leq M(1 - \alpha_{ijkk'}) \quad \forall (i, j) \in A^k, \forall k \in C, \forall k' \in E \quad (44)$$

$$x_{k+1}(\delta_k^+(o_k)) \leq x_k(\delta_k^+(o_k)) \quad \forall k \in B \setminus \{|B|\} \quad (45)$$

$$x_{k+1}(\delta_k^+(o_k)) \leq x_k(\delta_k^+(o_k)) \quad \forall k \in C \setminus \{|C|\} \quad (46)$$

$$x_{k+1}(\delta_k^+(o_k)) \leq x_k(\delta_k^+(o_k)) \quad \forall k \in E \setminus \{|E|\} \tag{47}$$

$$x_k(\delta_k^-(i' + 1)) \leq x_k(\delta_k^-(i')) \quad \forall i \in T, \forall i' \in T'(i) \setminus \{|T'(i)|\}, \forall k \in C \tag{48}$$

$$s_{ik} \in \mathbb{R}^+ \quad \forall i \in V, \forall k \in K \tag{49}$$

$$z_{ik} \in \mathbb{N} \quad \forall i \in V, \forall k \in K \tag{50}$$

$$v_{ik} \in \mathbb{R}^+ \quad \forall i \in V, \forall k \in K \tag{51}$$

$$\sigma_{ik} \in \mathbb{R}^+ \quad \forall i \in V, \forall k \in K \tag{52}$$

$$x_{ijk} \in \{0, 1\} \quad \forall (i, j) \in A^k, \forall k \in K \tag{53}$$

$$f_{ijrk} \in \mathbb{R}^+ \quad \forall (i, j) \in A^k, \forall k \in K, \forall r \in R \tag{54}$$

$$w_{irkk'} \in \mathbb{R}^+ \quad \forall i \in \bar{T}, \forall r \in R, \forall k, k' \in K \tag{55}$$

$$y_{irkk'} \in \{0, 1\} \quad \forall i \in \bar{T}, \forall r \in R, \forall k, k' \in K \tag{56}$$

$$\alpha_{ijkk'} \in \{0, 1\} \quad \forall (i, j) \in A^k, \forall k \in C, \forall k' \in E \tag{57}$$

The objective function (24) aims at minimizing the total cost. Constraints (25) guarantee that every request is fully satisfied. Constraints (26) make sure, that every long haul vehicle leaves its start depot and returns to its end depot. This constraint is not necessary for short haul vehicles, since we only consider trips where a short haul vehicle carries load. Constraints (27) guarantee the flow conservation of long haul vehicles. Again, this constraint is not necessary for short haul vehicles, because we are not looking for a path for short haul vehicles but for single trips (which correspond to single arcs). Constraints (28) make sure that the same amount of load of a request which arrives at a transshipment node also leaves the transshipment node. Constraints (29)–(32) keep track of time and guarantee that a vehicle never exceeds the amount of hours per day it is allowed to operate. Constraints (29) keep track of the sum of travel- and service times of a vehicle until a node, while constraints (30) determine the number of rests a vehicle has to make until a node based on this sum. Constraints (31) determine the arrival time of a vehicle at a node. Finally, constraints (32) together with constraints (31) keep track of time overall by determining the start time of service of a vehicle at a node, taking the travel-, service-, and rest-times until this node into account. Constraints (33) ensure that the capacity of a vehicle is never exceeded. Constraints (34) ensure time synchronization at transshipment. That is, the vehicle receiving load must start its service after the vehicle giving the load. Constraints (35) guarantee that the amount transferred from one vehicle to another does not exceed the capacity of either vehicle. Constraints (36) make

sure that a vehicle leaves a transshipment node with the same amount of a request it arrived with, plus the amount of the request it received from other vehicles, minus the amount it gave to other vehicles. Constraints (37)–(39) determine the service time at each node. Constraints (40) ensure that the time windows are respected. Constraints (41) guarantee that a passive vehicle never moves on its own, but always together with an active vehicle. Constraints (42) ensure that the maximum number of passive vehicles per active vehicle is never exceeded. Constraints (43)–(44) regulate time synchronization between active and passive vehicles. Constraints (45)–(47) break the symmetry in the model with respect to the vehicles. Constraints (48) break the symmetry in the model with respect to the node duplicates of transfer locations. Constraints (49)–(57) give the domain of the decision variables.

References

- Andersen J, Christiansen M (2009) Designing new european rail freight services. *J Oper Res Soc* 60(3):348–360
- Andersen J, Crainic TG, Christiansen M (2009) Service network design with management and coordination of multiple fleets. *Eur J Oper Res* 193(2):377–389
- Andersson H, Christiansen M, Fagerholt K (2011) The maritime pickup and delivery problem with time windows and split loads. *INFOR* 49(2):79–91
- Archetti C, Bouchard M, Desaulniers G (2011) Enhanced branch and price and cut for vehicle routing with split deliveries and time windows. *Transp Sci* 45(3):285–298
- Barnhart C, Ratliff HD (1993) Modeling intermodal routing. *J Bus Log* 14(1):205
- Bock S (2010) Real-time control of freight forwarder transportation networks by integrating multimodal transport chains. *Eur J Oper Res* 200(3):733–746
- Cacchiani V, Hemmelmayr VC, Tricoire F (2014) A set-covering based heuristic algorithm for the periodic vehicle routing problem. *Discret Appl Math* 163:53–64
- Caris A, Macharis C, Janssens GK (2013) Decision support in intermodal transport: a new research agenda. *Comput Ind* 64(2):105–112
- Chang TS (2008) Best routes selection in international intermodal networks. *Comput Oper Res* 35(9):2877–2891
- Cortés CE, Matamala M, Contardo C (2010) The pickup and delivery problem with transfers: formulation and a branch-and-cut solution method. *Eur J Oper Res* 200(3):711–724
- Crainic TG, Hewitt M, Toulouse M, Vu DM (2016) Service network design with resource constraints. *Transp Sci* 50(4):1380–1393
- Desaulniers G (2010) Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. *Oper Res* 58(1):179–192
- Eurostat (2016a) Database by themes, leaf: annual road freight transport, by distance class. Online, accessed December, 7th 2016
- Eurostat (2016b) Database by themes, leaf: modal split of freight transport. Online, accessed December, 7th 2016
- Feillet D, Dejax P, Gendreau M, Gueguen C (2004) An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems. *Networks* 44(3):216–229
- Lourenço HR, Martin OC, Stützle T (2010) Iterated local search: framework and applications. In: *Handbook of metaheuristics*. Springer, New York, pp 363–397
- Masson R, Lehuédé F, Péton O (2013) An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transp Sci* 47(3):344–355
- Medina J, Hewitt M, Lehuédé F, Péton O (2018) Integrating long-haul and local transportation planning: the service network design and routing problem. *EURO Journal on Transportation and Logistics* pp 1–27

- Mitrovic-Minic S, Laporte G (2006) The pickup and delivery problem with time windows and transshipment. *INFOR* 44(3):217
- Moccia L, Cordeau JF, Laporte G, Ropke S, Valentini MP (2011) Modeling and solving a multimodal transportation problem with flexible-time and scheduled services. *Networks* 57(1):53–68
- Neves-Moreira F, Amorim P, Guimarães L, Almada-Lobo B (2016) A long-haul freight transportation problem: synchronizing resources to deliver requests passing through multiple transshipment locations. *Eur J Oper Res* 248(2):487–506
- Nowak M, Ergun Ö, White CC III (2008) Pickup and delivery with split loads. *Transp Sci* 42(1):32–43
- Pedersen MB, Crainic TG, Madsen OB (2009) Models and tabu search metaheuristics for service network design with asset-balance requirements. *Transp Sci* 43(2):158–177
- Qu Y, Bard JF (2012) A grasp with adaptive large neighborhood search for pickup and delivery problems with transshipment. *Comput Oper Res* 39(10):2439–2456
- Rais A, Alvelos F, Carvalho MS (2014) New mixed integer-programming model for the pickup-and-delivery problem with transshipment. *Eur J Oper Res* 235(3):530–539
- Ropke S, Cordeau JF (2009) Branch and cut and price for the pickup and delivery problem with time windows. *Transp Sci* 43(3):267–286
- Semet F, Toth P, Vigo D (2014) Classical exact algorithms for the capacitated vehicle routing problem. *Vehicle Routing Probl Methods Appl* 18:37
- Stålhane M, Andersson H, Christiansen M, Cordeau JF, Desaulniers G (2012) A branch-price-and-cut method for a ship routing and scheduling problem with split loads. *Comput Oper Res* 39(12):3361–3375
- StadieSeifi M, Dellaert NP, Nuijten W, Van Woensel T, Raoufi R (2014) Multimodal freight transportation planning: a literature review. *Eur J Oper Res* 233(1):1–15
- Ziliaskopoulos A, Wardell W (2000) An intermodal optimum path algorithm for multimodal networks with dynamic arc travel times and switching delays. *Eur J Oper Res* 125(3):486–502