The vessel swap-body routing problem

Vinícius S. M. Gandra[a,b], Hatice Çalık[a], Túlio A. M. Toffolo[b], Marco Antonio M. Carvalho[b], Greet Vanden Berghe[a]

[a]*KU Leuven, Dept. of Computer Science; Leuven.AI, B-3000 Leuven, Belgium*
[b]*Department of Computing, Federal University of Ouro Preto, Brazil*

Abstract

This paper introduces the Vessel Swap-Body Routing problem (VSBR), a generalization of the pickup and delivery problem with time windows, which considers freight distribution between ports located throughout an inland waterway network. Subject to time windows and precedence constraints, each customer request is associated with a number of containers and must be served via a single body. Bodies are capacitated components that cannot move independently and must therefore be towed by a vessel. Bodies can be transferred between vessels at customer locations or transfer points in order to reduce overall costs. Vessels and bodies can end their routes at any location, meaning they do not need to return to a depot. Moreover, every vessel-body combination is permitted, which greatly expands the size of the solution search space. Although body transfers constitute a fundamental component of this real-world problem, the flexibility such transfers engender poses a huge logistical challenge to the human planners tasked with efficiently scheduling vessel routes. In this paper we model the VSBR as an optimization problem and introduce complementary approaches for solving it. We propose a mixed integer programming formulation and a heuristic approach with tailored neighborhoods for body transfers. To help stimulate further research, a set of instances is introduced based on real-world data and benchmarks are made publicly available.

Keywords: Transportation, heuristics, swap-body, transfers, mixed integer programming.

1. Introduction

This paper investigates a routing problem introduced to us by a shipping company operating in an inland waterway network. Given a scheduling horizon, the company must complete hundreds of *requests*. Each of these requests is associated with a number of containers which must be picked up from one customer location and delivered to another while respecting time windows. Customer locations correspond to container terminals where they retrieve the goods being shipped, thus different customers may be associated with the same location. Containers associated with requests are loaded into *bodies*: capacitated transports which can only move when towed by a *vessel*. Vessels themselves are incapable of holding containers, but instead have a capacity limit concerning the number of bodies they can tow. All containers associated with a single request must be loaded into the same body and once loaded they cannot be transferred into another. Figure 1 illustrates a vessel-body configuration where two bodies are attached to a vessel.

Bodies are not fixed to vessels and the transportation network contains locations where these bodies can be parked. This enables the possibility of conducting *body transfers*, where a body is *detached* from

Email addresses: `vinicius.gandramartinssantos@kuleuven.be` (Vinícius S. M. Gandra), `hatice.calik@kuleuven.be` (Hatice Çalık), `tulio@toffolo.com.br` (Túlio A. M. Toffolo), `mamc@ufop.edu.br` (Marco Antonio M. Carvalho), `greet.vandenberghe@cs.kuleuven.be` (Greet Vanden Berghe)

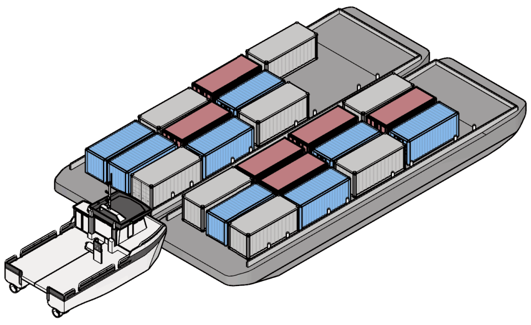


Figure 1: Example of a vessel towing two bodies.

one vessel at a special transfer point or customer location and then later *attached* to either the same vessel or a different one. Transfer points typically correspond to central locations of the network. Bodies can be detached at transfer points without any restrictions and may be picked up at any time. Body transfers significantly increase the solution space. They introduce a new decision level to the problem (where and when to perform these transfers) and require route synchronization: a vessel must first detach a body before another can visit the transfer point or customer location to attach it.

Human operators are unable to exploit the full potential of body transfers given the added logistical challenges they bring to the problem. Thus they are often ignored in practice or used in a very limited way. We believe that despite the extra computational effort needed, employing transfers has the potential to significantly reduce overall costs. Therefore, in this paper we introduce a formulation for this transportation problem and propose a solution approach which enables us to study the impact of body transfers on solution quality. We refer to this new transportation problem as the *Vessel Swap-Body Routing problem* (VSBR).

The VSBR is a generalization of the Pickup and Delivery Problem (PDP) with time windows. Important insights concerning how to model and solve the VSBR can therefore be drawn from other PDP generalizations. One such generalization is the Pickup and Delivery Problem with Transshipment (PDPT), which also involves the transfer of goods between routes and complex route synchronization. However, unlike how it is possible to transfer individual requests from one vehicle to another in the PDPT, the entire set of requests held by a body must be transferred from one vessel to another in the VSBR. Another important generalization worth considering is the Truck-and-Trailer Routing Problem (TTRP) where detachable trailers are towed by trucks and each trailer is assigned to a specific truck. By contrast, bodies in the VSBR are treated independently from vessels and therefore a given body can be towed by any vessel. Thus, all vessel-body combinations are permitted as long as vessel capacities (which often exceed two bodies) are respected. Moreover, it is also worth noting that in the VSBR all customers can be visited by all possible vessel-body combinations. Body transfers are optional and only employed in an attempt to reduce overall costs. Conversely, detaching trailers in the TTRP is performed to ensure feasibility: certain customers can only be reached by trucks without any trailer attached. Although related to the VSBR and worth reviewing, these other PDP generalizations are unable to address the important characteristics of the inland waterway distribution scenario we investigate in this paper.

In order to clarify the remaining details of the VSBR, Figure 2 illustrates a solution with body transfers in a toy network of 12 customer locations (the triangles) and one transfer point (the blue diamond). Customers with pickup services (p_i) are depicted as upward triangles, while downward triangles correspond to delivery services (d_i). The route of each body is illustrated by means of colored directed edges, while nodes serviced by a given body have the same color. The three vessels responsible for towing bodies have their routes illustrated with distinct dashed edges and their initial positions are depicted by way of gray squares. The VSBR considers open routes, where vessels and bodies depart from a given location and may end at any location without having to return to a central depot. When en route, vessels are assumed to travel at the same constant speed, regardless of the number of bodies connected and containers loaded. Loading/unloading containers into/from bodies takes a certain length of time which is assumed to be the same for each container. Similarly, detaching/attaching a body also takes a fixed length of time independent of the configuration of attached bodies. For simplicity, time windows as well as travel times are omitted from Figure 2 and each customer location corresponds to a single service.

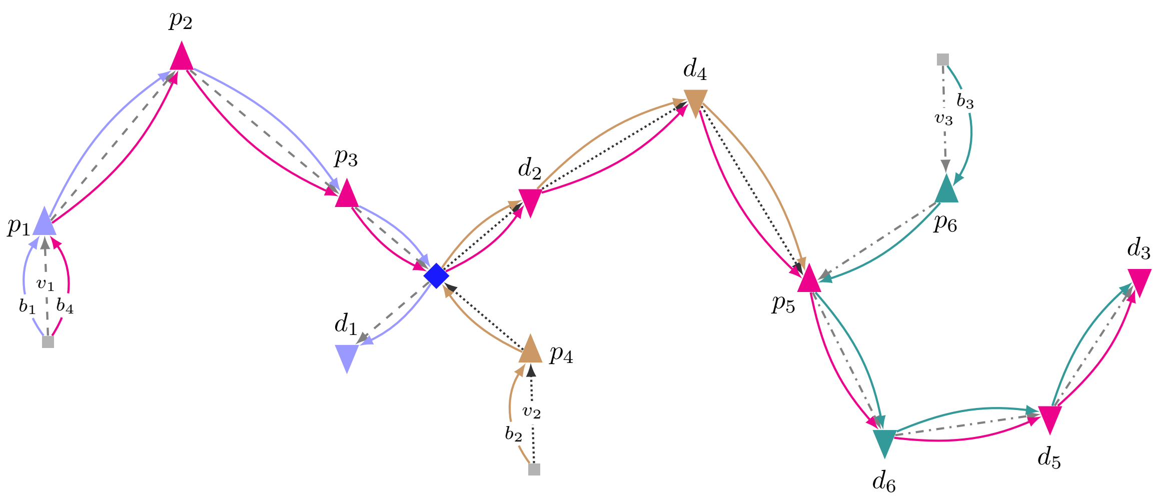


Figure 2: VSBR solution with body transfers.

Bodies b_1 and b_4 begin their routes attached to vessel v_1. Meanwhile, bodies b_2 and b_3 begin attached to vessels v_2 and v_3, respectively. Note that body b_4 serves the pickup and delivery services of requests 2, 3 and 5. To serve all of these requests, body b_4 is transferred between all three vessels. Towed by vessel v_1, b_4 serves p_2 and p_3 before being detached at the transfer point and attached to v_2. After serving d_2, body b_4 is detached for the second time at customer p_5 and serves the request while detached. Note that bodies detached at a customer's location must have a service to perform at that location (either pickup or delivery of containers). When such a transfer occurs, the vessel to which the body is attached arrives at the customer's location either before or during its time window. The vessel then detaches the necessary body and is free to continue on with its route without having to wait for the customer's time window to open or the service to complete. The detached body serves the customer as soon as the service time window is open. After the customer is served, the same or a different vessel may visit the customer to pick up the body. Finally, b_4 is attached one final time and towed by v_3 in order to deliver its final two requests. Note that if body transfers were not allowed, the vessel that picks up service p_3 would need to traverse almost the entire network to serve its paired service d_3.

Despite the fact that all requests were served in this small example, this might not always be the case. The homogeneous fleet of vessels and bodies available to serve requests is considered fixed and given a priori. Therefore given the capacity constraints of the available fleet it may prove impossible to serve every request within its time window, in which case some requests must be outsourced and served by trucks. Outsourcing requests in this manner incurs high costs and should thus be avoided whenever possible. The objective function of the VSBR is therefore designed to minimize the sum of (i) vessel travel costs and (ii) outsourcing costs.

Figure 2 not only makes it clear why human operators currently avoid dealing with the complexity of body transfers, but also highlights the potential for those transfers to improve solutions. Transferring requests in batches is not necessarily an exclusive feature of the VSBR and could occur in the context of waterway, rail or road transportation. The challenging new decision level the VSBR involves, in addition to the range of transportation systems within which it may occur, makes us confident that it is worthwhile introducing this new problem to the academic community. The following section presents an overview of related research, comparing and contrasting them with the VSBR. Section 3 introduces the necessary notation and presents a mixed integer programming formulation for the problem. To tackle instances of realistic size, a solution approach is introduced and described in Section 4. In order to assess the impact of body transfers on solution quality, we report the results of computational experiments in Section 5. Given the lack of benchmark instances for the VSBR, instance sets are generated based on real-world data. Finally, Section 6 concludes the paper and outlines some directions for future research.

2. Related work

The combination of multiple starting locations, open routes, numerous vessel-body assignments and body transfers makes the VSBR a unique vehicle routing problem. Nevertheless, other routing problems can be identified in which specific requests must be served considering transfers and different vehicle combinations. The most similar problems are the pickup and delivery problem with transshipment (PDPT) and the VRP with trailers and transshipment. Distinctions can be made regarding various characteristics such as what is being transferred, transport combinations and accessibility constraints. This section presents an overview of these related problems in order to identify some similarities and key differences with respect to the VSBR.

2.1. Pickup and delivery problems with transshipment

Several variants of pickup-and-delivery problems (PDPs) exist and have been studied extensively. Important surveys of these variants have been conducted by Cordeau et al. (2008), Parragh et al. (2008) and Battarra et al. (2014). A generalization of the PDP arises when requests are permitted to be transferred at so-called transfer points in order to minimize vehicle routing costs. Shiri et al. (2020) studied the impact of allowing transfers in pickup-and-delivery systems with different modeling (objective functions), system design (number of locations and transfer points) and operational parameters (capacity, cost and number of vehicles). This study revealed the gains that are possible in many different scenarios when permitting transfers to take place.

Transfers significantly increase the difficulty of these problems as they introduce precedence and synchronization constraints between multiple routes. If a request is transferred at a transfer point, the vehicle which picks up this request must visit the transfer point after the vehicle which dropped it off. Noting the interdependence of routes in the PDPT and the difficulty of efficiently checking the feasibility of candidate transfers, Masson et al. (2013b) proposed a method which checks the feasibility of solutions in constant time. This method can be used in the VSBR to efficiently avoid cross synchronization between body transfers. For a more thorough review of multiple synchronization constraints in VRPs and their applications we refer interested readers to the survey by Drexl (2012).

Table 1 provides an overview of recent research concerning PDPT variants which focuses on their transfer characteristics. In the PDPT, a request i given by the pair (r_i^P, r_i^D) and served by a single vehicle may be divided into two requests served by two vehicles given by pairs (r_i^P, t) and (t, r_i^D), where t is a transfer point. Most of the papers listed in Table 1 transfer requests (goods or passengers) at most once at pre-determined transfer points, while routes have fixed start and end points. Variants differ with respect to time windows (PDPTWT), split delivery, maximum route duration and hard time windows at transfer points.

Table 1: Overview of PDPT methods and transfer details.

	Problem	Method	What is transferred?	#Transfers	Transfer locations	Start and end locations	Extra constraints
Mitrović-Minić and Laporte (2006)	PDPTWT	H	Request	1/r	TP	F	
Cortés et al. (2010)	PDPTWT	E	Request	1/r	TP	F	Time window at transfer points
Takoudjou et al. (2012)	PDPTWT	E	Request	1/r	TP	F	
Qu and Bard (2012)	PDPTWT	H	Request	1/r	TP	F	
Masson et al. (2013a)	PDPTWT	H	Request	1/r	TP	F	
Masson et al. (2014)	PDPTWT	H	Request	1/r	TP	F	Max route duration and travel time
Rais et al. (2014)	PDPT + PDPTWT	E	Request	1/r	TP	F and NF	Split delivery
Danloup et al. (2018)	PDPTWT	H	Request	1/r	TP	F	
Zhang et al. (2020)	PDPTWT	E and H	Request	1/r	TP	F	
Wolfinger (2021)	PDPTWSLT	E and H	Request	1+/r	TP	F	Split delivery
This paper	VSBR	E and H	Body containing multiple requests	one or more per body	TP and customer locations	NF	

PDPTWSLT: PDPT with time windows and split deliveries. (N)F: (Not)Fixed locations to start and end routes.
1(+)/r: one (or more) transfer per request. TP: transfer points. E/H: Exact/Heuristic approach.

In contrast to the PDPT, transfers in the VSBR are not conducted at an individual request level. Instead only entire bodies, which contain multiple requests, may be transferred. The VSBR also allows bodies to be transferred multiple times, either at transfer points or customer locations, and their routes do not need to end at a fixed location. For example, in Figure 2, body b_4 is transferred twice. The transfer takes place when b_4 is detached after picking up two requests, which leads to multiple changes in the route of the vessel to which it is subsequently attached. These differences concerning what is being transferred and how many requests are involved in each transfer makes it challenging to incorporate efficient transfer insertion methods proposed for PDPTs into the VSBR.

2.2. Vehicle routing problems with trailers and transshipment

The VSBR exhibits some characteristics of truck-and-trailer routing problems (TTRPs). These represent a generalization of the VRP where lorries may extend their capacity by towing at most one trailer: a capacitated component that depends on lorries to move. Customers may be visited by either a lorry or a lorry-trailer combination (LTC). Due to accessibility constraints, some customers can only be visited by lorries and are referred to as lorry customers (LCs), while others can be visited by lorries with or without a trailer and are referred to as trailer customers (TCs). In order to visit LCs, an LTC may detach its trailer at a transfer point and serve LCs with a sub-tour that starts and ends at this transfer point. Loads may be transferred between the lorry and its respective trailer at a transfer point. At the end of the sub-tour, the lorry then picks up the trailer and continues on with its route which must end at a depot. The goal of the TTRP is to determine routes for lorries and LTCs so that every customer is served and routing costs are minimized while respecting loading capacities, accessibility constraints and time windows (if any). The surveys conducted by Prodhon and Prins (2014) and Cuda et al. (2015) contain sections on TTRPs, while Drexl (2013) has shown how several VRPs with different synchronization constraints may be modeled as variants of the TTRP.

A variant of the TTRP is the Swap-Body Vehicle Routing Problem (SB-VRP), which was introduced in the first Vehicle Routing and Logistics Optimization (VeRoLog[1]) solver challenge in 2014. Similar to the TTRP, the SB-VRP also considers vehicles with either one or two swap-bodies in addition to customers with accessibility constraints. A truck towing two swap-bodies is called a train in the SB-VRP. Besides parking and picking up the swap-body at a transfer point, a train may detach the front swap-body instead of the one at the back. A third class of customers is also considered which has a higher demand than the capacity of a single swap-body and must therefore be visited by a train. The objective function of the SB-VRP also incorporates more terms such as fixed usage cost for trucks and trailers as well as additional costs for conducting operations at transfer points. However, the SB-VRP permits neither swap operations at TCs nor transferring loads between swap-bodies. Given the similarities between the TTRP and SB-VRP, we believe the swap-body terminology was coined given its vehicle-independent phrasing which may denote trailers, railroad cars or barges.

Table 2: TTRP methods and transfer details.

Reference	Problem	Method	Transfer location	Edge cost	Fleet size	Fixed vehicle-body combination
Chao (2002)	TTRP	E and H	D and TC	Same	L	✓
Scheuerer (2006)	TTRP	H	D and TC	Same	L	✓
Lin et al. (2009)	TTRP	H	D and TC	Same	L	✓
Caramia and Guerriero (2010)	TTRP	E and H	D and TC	Same	L	✓
Lin et al. (2011)	TTRPTW	H	D and TC	Same	U	✓
Derigs et al. (2013)	TTRP + TTRPTW	H	D and TC	Same	L	✓
Villegas et al. (2013)	TTRP	E and H	D and TC	Same	L	✓
Parragh and Cordeau (2017)	TTRPTW	E and H	D and TC	Same	L	✓
Rothenbächer et al. (2018)	TTRP + TTRPTW	E	TC and TP	VD	L	✓
Toffolo et al. (2018)	SB-VRP	H	TP	VD	U	✓
Drexl (2021)	PD-TTRPTW	H	TP	VD	U	✓
This paper	VSBR	E and H	TP and any customer location	Same	L	–

D: depot. TC: trailer customer. VD: vehicle-dependent. TP: transfer points.
U/L: Unlimited/Limited fleet of vehicles. E/H: Exact/Heuristic approach.

Recent research regarding the TTRP is documented in Table 2, focusing on transshipment characteristics. A few authors considered the TTRP with time windows (TTRPTW), while Drexl (2021) is the only one to consider the TTRPTW with pickup and delivery. In addition to the constraints already introduced, routes may be subject to maximum duration, lorries may have fixed trailer assignments (meaning trailer transfers between lorries are not allowed), or it may be possible for a trailer to be detached multiple times during an

[1]EURO Working Group on Vehicle Routing and Logistics Optimization (`http://www.verolog.eu/`)

LTC route. The papers documented in Table 2 typically consider transfers only at depots and TC locations, while the routing cost does not depend on the vehicle used. However, some papers include dedicated transfer points and edge costs depending on the vehicle used (only lorry or LTC). Considering the TTRP proposed by Chao (2002) as the baseline, variants may consider an unlimited and/or heterogeneous fleet of vehicles and forbid load transfer between lorries and their corresponding trailers. Given that Toffolo et al. (2018)'s algorithm won first place in the VeRoLog competition, their paper is included in Table 2 as representative of the SB-VRP.

The VSBR also considers a vehicle which moves independently while towing a capacitated component that can be detached at special locations. However, unlike TTRPs, the VSBR considers neither customer accessibility constraints nor fixed assignments concerning vessels and bodies. Therefore, any vessel-body combination is valid and may visit any customer location. Consider a hypothetical generalized problem HYP where V is the set of all vehicles (vessels or trucks), B is the set of bodies/trailers, N is the set of nodes, $V_b \subset V$ is the subset of vehicles that can tow a body/trailer b, and $N_b \subset N$ is the subset of nodes that b can access. The TTRP is then a special case of HYP where an arbitrary trailer b can only be towed by a specific vehicle v_b and therefore $V_b = \{v_b\}$. By contrast, the VSBR would correspond to a second special case where $V_b = V$ and $N_b = N$. One can further conclude that neither of these two special cases can be reduced to the other.

Another crucial difference between the two problems concerns how vessels in the VSBR may tow multiple bodies at the same time. Moreover, (un)loading operations can be performed without the presence of the vessel when a body is detached at a customer location. For TTRPs, trailers are used to increase overall capacity, minimize total routing costs, and reduce the number of necessary vehicles. Transfers are thus employed as a mechanism to temporarily reduce vehicle size and access customers at restricted locations. In the VSBR by contrast, bodies are the only capacitated vehicles capable of transporting containers and transfers are used to minimize travel and outsourcing costs. When one takes into account these key differences between TTRPs and the VSBR, the development of a dedicated approach for the VSBR is clearly required.

3. Notation and problem formulation

The VSBR considers a set of customer locations, a set V of vessels, a set B of bodies and a set R of requests. Each request $r \in R$ is associated with a number of containers ρ_r that must be picked up from one customer and delivered to another by a single body in a single trip. The pickup should be conducted within time window $[t_r^{P-}, t_r^{P+}]$ and the delivery should be conducted within time window $[t_r^{D-}, t_r^{D+}]$. The VSBR considers serving these requests within a predetermined time horizon. Depending on the start and end of this horizon and the time windows of individual requests, three types of request are possible:

R^1 is the set of requests with both pickup and delivery services,

R^2 is the set of requests with only a delivery service,

R^3 is the set of requests with only a pickup service, where $R = R^1 \cup R^2 \cup R^3$.

Requests are expected to be served by vessels using the bodies attached to them. However, when a request cannot be served by any vessel within its time window, it is outsourced and assumed to be served by trucks at a high cost π. Outsourced requests are therefore not routed in this problem formulation. The service time of each request r is proportional to ρ_r and is denoted by s_r.

Each vessel $v \in V$ can travel with at most Q^V bodies, each having capacity Q^B. These bodies can be attached to or detached from vessels at any transfer point or a service location to be served by these bodies. The time needed for attaching or detaching a body is T^C. When transferring bodies at a service location, the detaching/attaching of such bodies may occur before/after the start/end of the service's time window. Therefore, at the end of a scheduling horizon bodies may be left unattached to a vessel. As a result, bodies may also start a scheduling horizon unattached.

3.1. A modified network construction

In order to formulate the VSBR we introduce a network $G = (N, A)$, as depicted in Figure 3. The network comprises of node set $N = I \cup S \cup J^0 \cup \{*\}$ and arc set A. We associate each transfer, pickup and delivery service with two nodes due to the possibility of detaching a body at the corresponding service location and reattaching it later to the same or another vessel. During the timespan between detaching and reattaching a body to the same vessel, this vessel may conduct services associated with other requests.

Let us define $I = P \cup P' \cup D \cup D'$, depicted as circles in the network, such that P and P' denote the sets of nodes associated with pickup services, while D and D' denote the sets of nodes associated with delivery services. Note that if the containers associated with a request are already in a body, then we only need the delivery nodes for this request (see request 2 in Figure 3). Similarly, if a request only has a pickup service, then we only need to create pickup nodes for this request (see request n in Figure 3).

Set S, illustrated by diamonds, contains the transfer nodes which are not customer nodes. Let us assume that each body b can be detached at a transfer node s at most m times. For each visit of b to s, we create two transfer nodes, say s^1 and s^2, whose physical locations are identical to those of s. Only body b can be detached at s^1, while b then moves from s^1 to s^2 detached from any vessel before it can be picked up again from s^2. Let $S_b = S_b^1 \cup S_b^2$ be the set of such transfer nodes created for body b and let $s_k^2 \in S_b^2$ denote the second copy of node $s_k^1 \in S_b^1$. Then $S = \bigcup_{b \in B} S_b$.

Set J^0 contains a node o_v (o_b) associated with the initial location of each vessel v (body b). Vessels do not need to return to their initial location. In order to simplify the modeling, we introduce a dummy node "$*$" to the network and ensure that every vessel and body ends their route at this dummy node. Initial and final location nodes are represented by squares in Figure 3.

Let $r_i \in R$ denote the request associated with $i \in I$. We then construct arc set $A = A^1 \cup A^2 \cup A^3 \cup A^4 \cup A^5 \cup A^*$ as follows:

$A^1 = \{(i, j) : i \in J^0 \cup S, j \in N, i \neq j\}$ is the set of arcs from vessel/body starting nodes and swap nodes to any other node,

$A^2 = \{(i, i + n) : i \in P\} \cup \{(i, j) : i \in P, j \in N \setminus \{u \in I : r_u = r_i\}\}$ is the set of arcs from each node $i \in P$ to every other node except the delivery nodes associated with r_i,

$A^3 = \{(i, i + n) : i \in P'\} \cup \{(i, j) : i \in P', j \in N \setminus \{u \in I : r_u = r_i\}\}$ is the set of arcs from each node $i \in P'$ to its corresponding delivery node in D and to every other node except those associated with r_i,

$A^4 = \{(i, i + n) : i \in D\} \cup \{(i, j) : i \in D, j \in N \setminus \{u \in I : r_u = r_i\}\}$ is the set of arcs from each node $i \in D$ to its copy in D' and to every other node except the pickup nodes associated with r_i,

$A^5 = \{(i, j) : i \in D', j \in N \setminus \{u \in I : r_u = r_i\}\}$ is the set of arcs from each node $i \in D'$ to every other node except those associated with r_i,

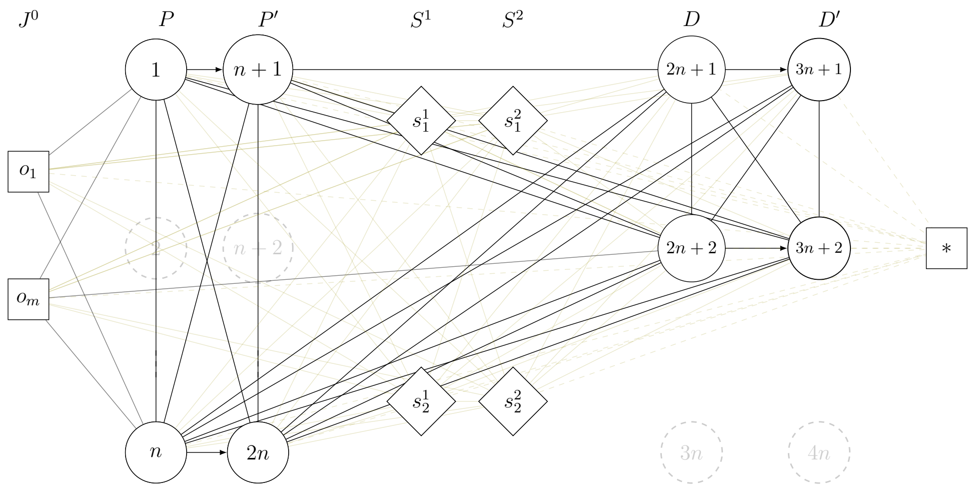


Figure 3: A modified network representation for the model

$A^* = \{(i,*), i \in N \setminus \{*\}\}$ is the set of arcs from every node to the sink node.

The travel time and the travel cost for each arc $(i,j) \in A$ is denoted by e_{ij} and c_{ij}, respectively. The travel cost is calculated as $c_{ij} = e_{ij} * \epsilon$, where ϵ is a coefficient which expresses cost per time unit and is used to convert time into cost. For each dummy arc $(i,j) \in A^*$, we set $e_{ij} = c_{ij} = 0$. Let L denote the set of physical locations and $l_i \in L$ denote the location associated with node $i \in N$. Then $e_{ij} = c_{ij} = 0$ and $\forall(i,j) \in A : l_i = l_j$, whereas $e_{i(i+n)} = s_{r_i}$ for $i \in P \cup D$. We further define $A^D = A^* \cup \{(i,j) \in A : l_i = l_j\}$ as the set of arcs bodies can traverse without being attached to any vessel. If node $i \in D \cup D'$ is associated with a request whose pickup service is already completed (meaning $r_i \in R_2$), then $b_i = b_{r_i} \in B$ denotes the body that holds the containers of this request.

3.2. Decision variables

For the introduced model we define the following sets of decision variables:

$z_{ij}^v = 1$ if vessel $v \in V$ traverses arc $(i,j) \in A$, 0 otherwise.

$\beta_{ij}^b = 1$ if body $b \in B$ traverses arc $(i,j) \in A$, 0 otherwise.

$\omega_{ij}^{bv} = 1$ if body $b \in B$ is attached to vessel $v \in V$ and traverses arc $(i,j) \in A$, 0 otherwise.

$x_{ij}^{br} = 1$ if the containers of request $r \in R$ traverse arc $(i,j) \in A$ inside body $b \in B$, 0 otherwise.

$\delta_{ij}^b = 1$ if body $b \in B$ traverses arc $(i,j) \in A^D$ without being attached to any vessel, 0 otherwise.

$\gamma^r = 1$ if request $r \in R$ is served by a vessel-body combination, 0 otherwise.

$\tau_i^{Av} \geq 0$ is the arrival time of vessel $v \in V$ at node $i \in N$.

$\tau_i^{Dv} \geq 0$ is the departure time of vessel $v \in V$ from node $i \in N$.

$\tau_i^{BAb} \geq 0$ is the arrival time of body $b \in B$ at node $i \in N$.

$\tau_i^{BDb} \geq 0$ is the departure time of body $b \in B$ from node $i \in N$.

$y_{ij}^v = 1$ if $i \in N$ precedes $j \in N$ (not necessarily immediately) during the trip of vessel v, 0 otherwise.

$\varphi_{ij}^b = 1$ if $i \in N$ precedes $j \in N$ (not necessarily immediately) during the trip of body b, 0 otherwise.

3.3. A mixed integer programming formulation for the VSBR

The objective function: Equation (1) minimizes the total cost of vessel routes plus the outsourcing cost for requests served by trucks.

$$\min \sum_{v \in V} \sum_{(i,j) \in A} c_{ij} z^v_{ij} + \sum_{r \in R} \pi(1-\gamma^r) \tag{1}$$

Routing of vessels and bodies: By Constraints (2)-(4) and (10)-(13), each vessel starts its journey from the node associated with its initial location and ends it at the dummy node without any sub-tours. Constraints (5)-(7) and (14)-(17) function analogously for bodies. Constraints (8) and (9) ensure a body either traverses an arc while attached to a single vessel or not attached to any vessel (if possible), thereby disabling the possibility that a body is simultaneously attached to two or more vessels.

$$\sum_{j:(o_v,j) \in A} z^v_{o_v j} = 1, \qquad \forall v \in V \tag{2}$$

$$\sum_{j:(j,*) \in A} z^v_{j*} = 1, \qquad \forall v \in V \tag{3}$$

$$\sum_{j:(j,i) \in A} z^v_{ji} = \sum_{j:(i,j) \in A} z^v_{ij}, \qquad \forall v \in V, i \in I \cup S \tag{4}$$

$$\sum_{j:(o_b,j) \in A} \beta^b_{o_b j} = 1, \qquad \forall b \in B \tag{5}$$

$$\sum_{j:(j,*) \in A} \beta^b_{j*} = 1, \qquad \forall b \in B \tag{6}$$

$$\sum_{j:(j,i) \in A} \beta^b_{ji} = \sum_{j:(i,j) \in A} \beta^b_{ij}, \qquad \forall b \in B, i \in I \cup S \tag{7}$$

$$\beta^b_{ij} = \delta^b_{ij} + \sum_{v \in V} \omega^{vb}_{ij}, \qquad \forall b \in B, (i,j) \in A^D \tag{8}$$

$$\beta^b_{ij} = \sum_{v \in V} \omega^{vb}_{ij}, \qquad \forall b \in B, (i,j) \in A \setminus A^D \tag{9}$$

Sub-tour elimination for vessel routes: For elimination of sub-tours, we utilize Constraints (10) - (13), which are shown to provide tight bounds in a comparative study by (Öncan et al., 2009). This group of sub-tour elimination constraints have also been adopted in recent and relevant PDPT and PDPTW studies (Rais et al., 2014; Zhang et al., 2020; Christiaens et al., 2020).

$$z^v_{ij} \le y^v_{ij}, \qquad \forall (i,j) \in A, v \in V \tag{10}$$

$$y^v_{ij} + y^v_{ji} = 1, \qquad \forall (i,j) \in A : (j,i) \in A, v \in V \tag{11}$$

$$y^v_{ij} = 1, \qquad \forall (i,j) \in A : (j,i) \notin A, v \in V \tag{12}$$

$$y^v_{ij} + y^v_{jl} + y^v_{li} \le 2, \qquad \forall (i,j),(j,l),(l,i) \in A, v \in V \tag{13}$$

Sub-tour elimination for body routes: Constraints (14) - (17) are analogous to Constraints (10) - (13).

$$\beta_{ij}^b \le \varphi_{ij}^b, \qquad \forall (i,j) \in A, b \in B \tag{14}$$

$$\varphi_{ij}^b + \varphi_{ji}^b = 1, \qquad \forall (i,j) \in A : (j,i) \in A, b \in B \tag{15}$$

$$\varphi_{ij}^b = 1, \qquad \forall (i,j) \in A : (j,i) \notin A, b \in B \tag{16}$$

$$\varphi_{ij}^b + \varphi_{jl}^b + \varphi_{li}^b \le 2, \qquad \forall (i,j),(j,l),(l,i) \in A, b \in B \tag{17}$$

Capacity constraints: Constraints (18) and (19) ensure that capacities are not exceeded for bodies and vessels, respectively.

$$\sum_{r\in R} q_r x_{ij}^{br} \le Q^B \beta_{ij}^b, \qquad \forall b \in B, (i,j) \in A \tag{18}$$

$$\sum_{b\in B} \omega_{ij}^{bv} \le Q^V z_{ij}^v, \qquad \forall v \in V, (i,j) \in A \tag{19}$$

Serving requests: If a request is served, Constraints (20) ensure that all service arcs associated with that request are traversed by a body holding the containers of the request. Constraints (21) and (23) ensure that the delivery service is conducted by the same body as the pickup service for each request. Constraints (22) ensure that the flow between the first and the last service node associated with each request is preserved for each body. Constraints (24) prevent the containers associated with a request from being split across multiple bodies.

$$\sum_{b\in B} x_{i(i+n)}^{br_i} = \gamma^{r_i}, \qquad \forall i \in P \cup D \tag{20}$$

$$x_{i(i+n)}^{br_i} = x_{(i+2n)(i+3n)}^{br_i}, \qquad \forall i \in P : r_i \in R^1, b \in B \tag{21}$$

$$\sum_{k\in N: k\ne i, (j,k)\in A} x_{jk}^{br_i} = \sum_{k\in N: k\ne i+3n, (k,j)\in A} x_{kj}^{br_i}, \qquad \forall i \in P : r_i \in R^1, b \in B, j \in N \setminus \{*, i, i+3n\} \tag{22}$$

$$x_{i(i+n)}^{b_{r_i} r_i} = \gamma^{r_i}, \qquad \forall i \in D : r_i \in R^2 \tag{23}$$

$$\sum_{j:(j,i)\in A} x_{ji}^{b_1 r} + \sum_{j:(i,j)\in A} x_{ij}^{b_2 r} \le 1, \qquad \forall i \in N, r \in R, b_1, b_2 \in B : b_1 \ne b_2 \tag{24}$$

Updating arrival/departure times: Let $T^{max} = \max_{r\in R} t_r^{D+}$. Constraints (25) update the arrival and departure times of a vessel at the end nodes of an arc visited by that vessel. Constraints (26) function similarly for bodies.

$$\tau_j^{Av} \ge \tau_i^{Dv} + e_{ij} z_{ij}^v - T^{max}(1 - z_{ij}^v), \qquad \forall (i,j) \in A, v \in V \tag{25}$$

$$\tau_j^{BAb} \ge \tau_i^{BDb} + e_{ij} \beta_{ij}^b - T^{max}(1 - \beta_{ij}^b), \qquad \forall (i,j) \in A, b \in B \tag{26}$$

Departures after arrivals: Constraints (27) and (28) ensure that the departure time from a node is not earlier than the arrival time at that node for vessels and bodies, respectively.

$$\tau_i^{Dv} \ge \tau_i^{Av}, \qquad \forall i \in N, v \in V \tag{27}$$

$$\tau_i^{BDb} \geq \tau_i^{BAb}, \qquad \forall i \in N, b \in B \tag{28}$$

Attaching and detaching times: If a body enters node i attached to a vessel but leaves the node detached from that vessel, Constraints (29) add the detaching time to the vessel's departure time label from node i. Similarly, if a body enters the node detached from any vessel but leaves attached to a vessel, Constraints (30) add the attaching time to the vessel's departure time label from that node.

$$\tau_i^{Dv} \geq \tau_i^{Av} + T^C + T^{max} \sum_{j:(j,i)\in A} w_{ji}^{vb} - T^{max} \sum_{j:(i,j)\in A} w_{ij}^{vb} - T^{max}, \quad \forall b \in B, v \in V, i \in N : i \neq o_b, * \tag{29}$$

$$\tau_i^{Dv} \geq \tau_i^{Av} + T^C - T^{max} \sum_{j:(j,i)\in A} w_{ji}^{vb} + T^{max} \sum_{j:(i,j)\in A} w_{ij}^{vb} - T^{max}, \quad \forall b \in B, v \in V, i \in N : i \neq o_b, * \tag{30}$$

Vessel arrives before body: Constraints (31)-(34) establish the relation between the arrival and departure times of attached vessel-body pairs entering or leaving a node.

$$\tau_i^{BAb} \geq \tau_i^{Av} + T^{max} \sum_{j:(j,i)\in A} w_{ji}^{vb} - T^{max}, \qquad \forall i \in N, v \in V, b \in B \tag{31}$$

$$\tau_i^{BDb} \geq \tau_i^{Dv} + T^{max} \sum_{j:(j,i)\in A} w_{ji}^{vb} - T^{max}, \qquad \forall i \in N, v \in V, b \in B \tag{32}$$

$$\tau_i^{Av} \geq \tau_i^{BAb} + T^{max} \sum_{j:(i,j)\in A} w_{ij}^{vb} - T^{max}, \qquad \forall i \in N, v \in V, b \in B \tag{33}$$

$$\tau_i^{Dv} \geq \tau_i^{BDb} + T^{max} \sum_{j:(i,j)\in A} w_{ij}^{vb} - T^{max}, \qquad \forall i \in N, v \in V, b \in B \tag{34}$$

Time windows: The time windows at service nodes are respected via Constraints (35)-(38).

$$\tau_i^{BDb} \geq t_{r_i}^{P-} \gamma^{r_i}, \qquad \forall i \in P, b \in B \tag{35}$$

$$\tau_i^{BDb} \geq t_{r_i}^{D-} \gamma^{r_i}, \qquad \forall i \in D, b \in B \tag{36}$$

$$\tau_i^{BAb} \leq t_{r_i}^{P+} \gamma^{r_i}, \qquad \forall i \in P', b \in B \tag{37}$$

$$\tau_i^{BAb} \leq t_{r_i}^{D+} \gamma^{r_i}, \qquad \forall i \in D', b \in B \tag{38}$$

Transfers at non-client nodes: Constraints (39) ensure that a body is only detached at one of its own transfer nodes. Constraints (40) and (41) require body b to be detached if one of its transfer nodes is visited.

$$\sum_{j:(j,s)\in A} w_{js}^{vb} = \sum_{j:(s,j)\in A} w_{sj}^{vb}, \qquad \forall b \in B, v \in V, s \notin S_b \tag{39}$$

$$\sum_{j:(j,s_k^1)\in A} \beta_{js_k^1}^{b} = \delta_{s_k^1 s_k^2}^{b}, \qquad \forall b \in B, s_k^1 \in S_b^1 \tag{40}$$

$$\sum_{j:(s_k^2,j)\in A} \beta_{s_k^2 j}^{b} = \delta_{s_k^1 s_k^2}^{b}, \qquad \forall b \in B, s_k^2 \in S_b^2 \tag{41}$$

The binary and non-negativity restrictions on the decision variables are expressed via Constraints (42) and (43).

$$z_{ij}^{v}, y_{ij}^{v}, \beta_{ij}^{b}, \omega_{ij}^{bv}, x_{ij}^{br}, \delta_{ij}^{b}, \gamma^{r}, \varphi_{ij}^{b} \in \{0,1\}, \qquad \forall v \in V, b \in B, r \in R, (i,j) \in A \qquad (42)$$

$$\tau_{i}^{Av}, \tau_{i}^{Dv}, \tau_{i}^{BAb}, \tau_{i}^{BDb} \geq 0, \qquad \forall v \in V, b \in B, i \in N \qquad (43)$$

Assumption: In order to keep the model straightforward to follow, we make the following practical assumption. At the beginning of the scheduling horizon, we assume that the attaching or detaching of bodies to/from vessels at their starting point is performed a priori and thus the routing can begin immediately at time zero. This assumption has the following impact: if body b is attached to vessel v_1 but will be taken by vessel v_2 from its start location then the model will not count the time to detach b from v_1, but it will count the time to attach b to v_2.

Valid inequalities: Constraints (44) and (45) ensure that the containers associated with a request are not inside a body when arriving at the pickup service node or leaving the delivery service, respectively.

$$\sum_{b \in B} \sum_{j:(j,i) \in A} x_{ji}^{br_i} = 0, \qquad \forall i \in P \qquad (44)$$

$$\sum_{b \in B} \sum_{j:(i,j) \in A} x_{ij}^{br_i} = 0, \qquad \forall i \in D' \qquad (45)$$

During preliminary experiments using an off-the-shelf solver, Constraints (44) and (45) were helpful in decreasing the solving time of the MIP model. Moreover in these experiments, we observed that the problem very quickly becomes intractable for the solver even for some small instances. In order to handle larger instances in a reasonable amount of time, we introduce a heuristic algorithm in the following section.

4. A heuristic for the VSBR

Preliminary experiments revealed that the proposed integer programming formulation was unable to generate high-quality solutions within reasonable runtimes. Therefore, in order to provide time-constrained industry with a decision support tool they can use in practice, this chapter will introduce a tailored heuristic for the VSBR. Given the many problem components involved and the necessity of efficiently exploring the solution space, we will focus on the following two decision levels: (i) deciding when and at which locations to transfer bodies and (ii) PDPTW optimization.

The first decision level requires one to efficiently explore the vast number of possible body transfers given that each body can be transferred multiple times in many different places throughout the scheduling horizon. This decision level can be very disruptive since removing or inserting body transfers may impact multiple vessels' routes as well as the assignment of containers to bodies. However, once decided, body transfers can be fixed. What this means is that vessels have mandatory stops in their routes. The next decision level can then improve these routes by employing an efficient PDPTW algorithm which takes into account these fixed transfers.

Given the need to search with respect to two decision levels, the Iterated Local Search (ILS) method (Lourenço et al., 2003) seems an intuitive approach. Given an initial solution, ILS generates neighboring solutions by applying a perturbation method followed by a local search. Besides featuring an iterative search which is subdivided into levels, ILS has also demonstrated good results when applied to routing and scheduling problems (Lourenço et al., 2019). For instance, the SB-VRP was successfully addressed by Toffolo et al. (2018), who developed a hybrid algorithm combining ILS and the Late Acceptance Hill-Climbing (LAHC) metaheuristic (Burke and Bykov, 2017).

We therefore decided to adapt ILS in order to address the VSBR, henceforth referred to as ILS-SB, in which the perturbation method consists of a transfer phase where body transfers are inserted or removed. The local search phase of ILS-SB then fixes these transfers in the vessels' routes before solving a PDPTW where vessels with fixed bodies must be routed to serve requests. Each newly generated solution is potentially accepted based on a list inspired by LAHC. This acceptance criterion benefits from the fact that there is only one parameter which requires calibration: the length of the list. Although the high-level framework of our heuristic shares some similarities with Toffolo et al. (2018), we only employ the late acceptance list embedded into ILS and not the entire LAHC metaheuristic framework. All of the other components of our algorithm, which will be described in the following sections, are completely different and tailored to the VSBR.

The basic framework of ILS-SB is provided by way of Algorithm 1. First, the algorithm receives as input an initial solution s and the maximum length for the late acceptance list, denoted by l (line 1). The late acceptance list (*AcceptL*) is then initialized with the initial solution value (line 3), meaning that solutions generated during the first $l-1$ iterations will be accepted. At each iteration, ILS-SB generates a neighbor solution s' by performing a transfer and routing phase (lines 6-7). Solution s' replaces the current best solution if it improves upon s^* (lines 8-9). Similarly, solution s' replaces incumbent solution s if *AcceptL* is not full or if $f(s')$ improves upon the considered entry in the late acceptance list (lines 10-11). Next, *AcceptL* is updated with the cost of the incumbent solution (line 13). The best solution is returned (line 14) when the algorithm reaches either of its two stopping criteria: the maximum number of iterations without improvement ($\#maxIt$) or the time limit.

Algorithm 1: ILS-SB framework.

```
1  Input: initial solution $s$, list size $l$
2  $s^* \leftarrow s$
3  $AcceptL[0] \leftarrow f(s)$
4  $i \leftarrow 0$
5  while $\#maxIt$ or time limit is not reached do
6      $s_t \leftarrow$ Transfer phase($s$) // Section 4.4
7      $s' \leftarrow$ Routing phase($s_t$) // Section 4.5
8      if $f(s') < f(s^*)$ then
9          $s^* \leftarrow s'$;
10     if $length(AcceptL) < l$ or $f(s') < AcceptL[i \bmod l]$ then
11         $s \leftarrow s'$
12     i++
13     $AcceptL[i \bmod l] \leftarrow f(s)$
14 return $s^*$
```

4.1. Acceptance criterion

The VSBR's objective function aims to fulfill as many requests as possible with vessels while minimizing fuel consumption. Thus, the VSBR minimizes the travel cost plus the outsourcing costs incurred when requests are served by trucks. When using the objective function in Equation (1), referred to as OF^1, it is common to find different solutions with equal objective values. To distinguish between two solutions when this situation occurs, a secondary objective function, referred to as OF^2 and calculated by Equation (46), is used every time the first objective results in a tie. This secondary objective function minimizes the total *operational cost* plus outsourcing costs. The operational cost of a vessel is given by the time the last request was served minus the starting time of the vessel multiplied by the cost coefficient per unit time. Therefore, OF^2 takes into account everything already present in OF^1 in addition to request servicing, detaching/attaching and waiting times. As a result, a solution with identical travel cost but lower operational cost is preferable.

$$\min \sum_{v \in V} (\tau_*^{Av} - \tau_{o_v}^{Av})\epsilon + \sum_{r \in R} \pi(1 - \gamma^r) \tag{46}$$

4.2. Initial solution

At the start of the scheduling horizon, all vessels are located at a container terminal and have either zero or more bodies attached. Similarly, bodies have an initial location, may be initially connected to a vessel and may have zero or more containers already loaded. Since ILS-SB has methods dedicated to handling body transfers, we opt to quickly generate an initial solution which does not include such transfers. Thus, we propose an constructive method that generates a solution in which bodies remain attached to their initially assigned vessel and no transfers take place. This means that bodies which began detached will not be used, while a given vessel which serves a pickup service will also perform the corresponding delivery.

For a feasible solution, all capacity and temporal constraints must be satisfied and transportation requests which cannot be served by vessels are placed into the set of unserved requests U. A solution for the VSBR is represented by a set of routes for all vessels $v \in V$, where each route is defined by a sequence of services, the location of each of those, the body used to fulfill the service and any other attached bodies.

An initial solution is generated by inserting requests one by one into body routes. Requests are iterated over in a random order and inserted into the best position considering all body routes. The request insertion method designed to construct the initial solution is detailed in Section 4.3. To avoid poor quality initial solutions, ι candidates are produced from which the best is selected for ILS-SB.

4.3. Best insertion method for requests

The best insertion method attempts to insert every unserved request at the best position of a given solution. Given the sequence of visited nodes in a body/vessel route, an insertion position is given by a node n where the request is inserted into the route immediately after n. The list of unserved requests U is iterated over in one of the following three ways, which are selected with equal probability: (i) random order, (ii) request r with the earliest t_r^{P-} first or (iii) request r with the earliest t_r^{D-} first.

For each request r_i ($i \in R_1 \cup R_3$), the pickup service r_i^p is first inserted into the best position considering every position in the routes of all available bodies. Once inserted, the corresponding delivery service r_i^d is inserted into the best position of body route $b_{r_i^p}$. If $i \in R_2$, meaning that the pickup service was already loaded into a specific body b_j, only the positions in b_j will be considered for the insertion of r_i^d. When no feasible insertion position can be found for a pickup or delivery service, the complete request r_i is added to the set of unserved requests U.

Best insertion methods for PDPTWs are often costly in terms of processing time given the need to evaluate the solution after the insertion of a request into each position. To accelerate the method, we follow the efficient feasibility testing method for request insertion proposed by Savelsbergh (1992). Before the insertion of a request, both forward time slack and forward load slack are calculated in linear time. Forward time slack consists of the maximum amount of time a service in the route can be postponed by without violating any time windows constraints associated with succeeding services. Similarly, forward load slack corresponds to the maximum number of containers a body can load at a certain stop without violating the capacity constraints associated with any of its succeeding nodes. By maintaining these two forms of slack, it is possible to check in constant time whether or not a given insertion is feasible. When feasible positions for both the pickup and delivery service are obtained, the new solution value considering the request insertion is calculated in constant time with a delta evaluation which updates the travel cost based on the removed and added edges. The complete solution is therefore only evaluated once, after the insertion of the request into its best feasible position.

4.4. Transfer phase

The transfer phase inserts attaching and detaching operations into the solution. This phase introduces diversity with respect to the vessels' attached bodies and enables different vessel-body combinations to be explored. The rationale behind attaching/detaching a body is that a single vessel is neither obligated to visit both the pickup and delivery location of requests, nor does it need to wait for the entire container service time to elapse before departing. Therefore, by using the vessel time more efficiently, one may reduce travel cost and/or outsourcing costs.

We designed four transfer insertion and two transfer removal neighborhoods for the transfer phase. Each time the transfer phase is called, one of the six neighborhoods is uniformly selected. Transfer insertion neighborhoods are selected with $\nu\%$ probability, while transfer removal neighborhoods are selected with the remaining $100 - \nu\%$ probability. Transfers are performed either at a customer's location, where the detached body has one or more services to attend to, or at transfer points where the body is simply detached before later being reattached to either the same or a different vessel.

4.4.1. Body-transfer insertion

The four different transfer insertion neighborhoods follow the same basic steps outlined in Algorithm 2. Each neighborhood receives as input a solution, a body to be transferred, a node to perform the transfer and a position in the body's route to insert the transfer (lines 1-2). First, since locations after $prev_t$ will no longer be visited by b, the requests associated with those locations are removed from the body's route and inserted into the unserved request set U (line 3). If b was transferred after $prev_t$, the body transfer is also removed. When removing requests and/or transfers from the body's route, they are also removed from the route of the vessel towing body b. Next, the detaching operation is inserted at $node_t$ after $prev_t$ in the body's route (line 4).

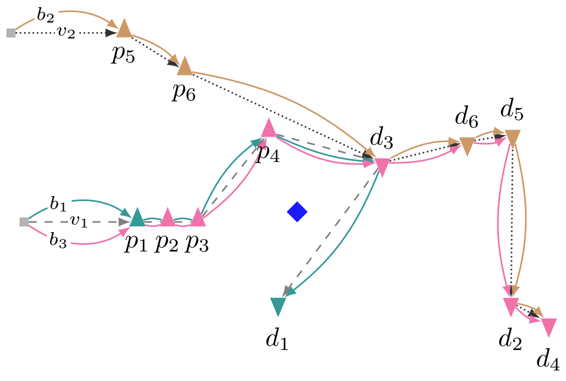
Algorithm 2: Body-transfer insertion.

```
1  Input: solution $s$, body $b$ to be transferred
2  Input: transfer node $node_t$, last node $prev_t$ into $b$'s route before transfer
3  $U \leftarrow$ Remove from $b$'s route every served request after $prev_t$ (s)
4  Detachment of $b$ at $node_t$ after $prev_t$ (s)
5  if Detached body is empty then
6      Best insertion of all requests in $U$ (s)
7  else
8      foreach vessel route $v_r \in V$ do
9          foreach vessel stop $v_{stop} \in v_r$ do
10             $s' \leftarrow$ Attaching of body $b$ after vessel stop $v_{stop}$ (s)
11             Best insertion of all requests in $U$ (s')
12             if $f(s') < f(s)$ then
13                 $s \leftarrow s'$
14 return $s$
```

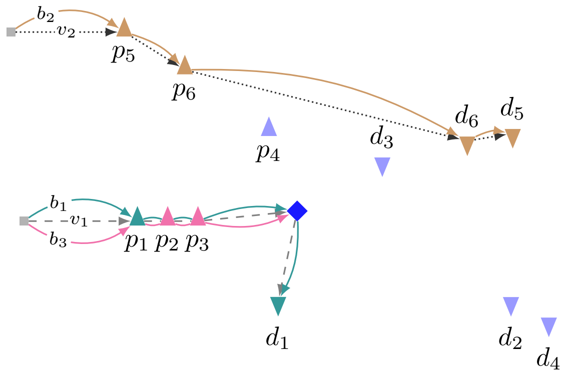
If the detached body is empty, meaning there is no container loaded in the body, the unserved requests are inserted into the solution using the best insertion method and the body is left unattached (lines 5-6). Otherwise, the body must be attached to a vessel in order to be able to continue on its route and deliver the containers that have already been loaded. Note that if a pickup service was served by b before $node_t$, the corresponding delivery service must also be served by b after the transfer. The attaching of a body is conducted in a "best insertion" manner. In other words, every position in each vessel route is checked before selecting the best insertion position. The vessel which will visit $node_t$ to attach body b must respect feasibility constraints such as vessel capacity, the time windows of customers visited after $node_t$ and cross synchronization which avoids cycle dependencies between transfers (Masson et al., 2013b). As with the best insertion of requests into body routes, the best insertion of body transfers into vessel routes calculates the forward time slack and forward load slack to accelerate the feasibility check of each insertion. After each feasible attachment, the unserved requests are inserted into the solution (lines 8-13). Finally, the solution yielding the lowest cost is saved and returned (line 14).

Figure 4 details the steps to insert a body transfer. The network representation follows the same format as that used in Figure 2, where the routes of bodies and vessels are depicted by colored and dashed edges, respectively. Figure 4(a) shows a solution for the VSBR where bodies b_1 and b_3 start attached to vessel v_1 and body b_2 starts attached to vessel v_2. A transfer occurs at node d_3, where body b_3 is detached by vessel v_1 and attached to vessel v_2.

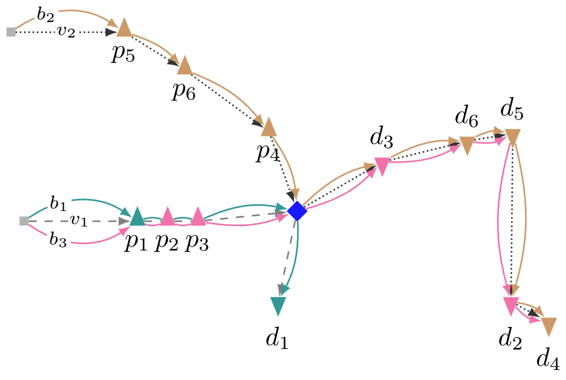
Consider a body-transfer insertion of body b_3 at the transfer point ($prev_t$ is p_3). Figure 4(b) shows a destroyed solution after detaching b_3 at the transfer point. Services p_4, d_3, d_2 and d_4 are all successors of p_3 in the route of body b_3 and are therefore removed from the solution and included in the set of unserved requests U. Customers removed from a body's route are also removed from the route of the vessel which tows this body. Note that these removed services alter the route of both vessels, with b_2 left at the transfer point with two services already loaded (p_2 and p_3). Figure 4(c) shows one possible way of repairing the solution, whereby body b_3 is picked up by vessel v_2. Services d_2 and d_3 must be served by body b_3 since their respective pickup services were previously loaded into this body. By contrast, both the pickup and delivery services of request 4 are unserved and can be served by any other body. In Figure 4(c), b_3 is picked



(a)



(b)



(c)

Figure 4: Body transfer insertion.

up by vessel v_2 and is able to serve requests d_2 and d_3. Finally, vessel v_2 serves request 4 by loading it into body b_2.

Each transfer insertion neighborhood may vary in terms of how it selects the node to perform the transfer and the body to be transferred. Neighborhoods and their particularities are outlined in what follows.

Score-based transfer insertion at a customer location (TI1). Given the large number of possibilities, performing transfers at randomly selected customers may lead to many poor-quality solutions. This first neighborhood attempts to minimize the chance of this occurring and identify promising customers to perform transfers. The neighborhood begins by selecting a customer c_r which is associated with either a pickup or delivery service of a request r not in U, in other words, a served customer. It is assumed that beneficial selections for customers are those with the following characteristics:

1. *Low travel cost*: customers which are located close to another vessel's route. Small detours for vessels which attach a body should not significantly affect the objective value.

2. *Large number of containers*: customers where a significant number of containers ρ_r are being served. The total service time of a customer is proportional to the number of containers to be (un)loaded. By detaching a body at such a customer, vessels will no longer need to wait for the full service time. For example, instead of waiting at a customer location a vessel may use this time to serve other requests, which can potentially reduce the travel cost of other vessels or reduce outsourcing costs.

Taking into account these two desirable characteristics, we introduce a function which assigns a score to each served customer c_r:

$$score_{c_r} = \alpha * \frac{1}{travel_{c_r}} + (1-\alpha) * \rho_r \tag{47}$$

In Equation (47), parameter α controls the extent to which travel cost and number of requested containers contribute to the selection criterion. Parameter $travel_{c_r}$ corresponds to the total travel cost between customer c_r and all the other customers which are visited by other bodies. A low $travel_{c_r}$ reflects a customer which is close to many others, while a high value implies that the customer in question is far away from others. The total number of containers (un)loaded at a customer by a body is defined by ρ_r, which is directly proportional to the total service time.

A probabilistic selection is employed in the present study to diversify the solution and avoid an overly greedy convergence of the method. Scores are sorted in descending order and a rank γ is assigned to each, which corresponds to their position in the array ($\gamma = 1$ for the first element). A rank-based $bias(\gamma)$ is then assigned to each option, which is calculated by an exponential bias function. The probability $p(\gamma)$ of selecting an option is given by Equation (48):

$$p(\gamma) = (\sum_{k=1}^{R} bias(k))^{-1} * bias(\gamma) \tag{48}$$

The node where the selected customer is located becomes $node_t$: the node to perform the detachment. Body b_r is the body to be detached and $prev_t$ is the last node visited by b_r before serving customer c_r. Note that c_r will still be served by b_r, but only after it is detached. After determining the value of these three variables, the transfer insertion method follows the steps outlined in Algorithm 2.

Transfer insertion at transfer points (TI2). Bodies may also be detached/attached at special transfer points where no additional services are performed. Transfer points are usually located in central positions and intersections which connect different clusters of customers. This neighborhood randomly selects a transfer point, which corresponds to $node_t$. Each body is then a candidate to be detached at $node_t$. A score is calculated for each body b_i and corresponds to the average distance between $node_t$ and every customer served by b_i. The exponential bias function used in neighborhood TI1 is also used here to select body b. The final decision to make is where in the body's route to insert the transfer point. The closest customer to $node_t$ visited by b is selected as $prev_t$.

Transfer at first node (TI3). Transferring at the first node enables a vessel to detach a body as its first action within the scheduling horizon, even if the body has no service to perform at this node. This assumes a service was conducted during the preceding scheduling horizon. Thus, this move serves as an opportunity for vessels to detach undesired bodies and free capacity to attach other bodies.

This neighborhood starts by selecting a vessel, with priority given to vessels with many bodies attached. A body b is then selected, with priority given to empty bodies. Note that detaching an empty body does not lead to immediate reattachment. The remaining variables, $node_t$ and $prev_t$, are given by the location where the vessel started and the dummy node at the beginning of the vessel's route, respectively.

Pick up originally detached body (TI4). The scheduling horizon may begin with detached bodies which may or may not be empty. It is advantageous for a vessel to attach an empty body when it needs more capacity, whereas attaching a body which started loaded with customer containers is required to fulfill all requests. First a detached body is selected. For this, the bias function gives priority to selecting bodies with loaded containers. The attachment location ($node_t$) is given by the node where the body is located at the start of the scheduling horizon. Since detaching is not required, $prev_t$ is null and lines 3-6 in Algorithm 2 are ignored.

4.4.2. Body-transfer removal

Algorithm 3 provides an overview of how to remove body transfers. A body transfer comprises of the delivery t_d and pickup t_p of a body b. Given a solution and a body transfer as input, each customer served by body b after pickup node t_p is removed from b's route and inserted into the unserved set (line 2). These nodes are also removed from the route of the vessel attached to body b. After removing all customers and transfers after t_p, the pickup node is safely removed (line 3).

Delivery node t_d is the last node to be removed, leaving the body attached to the vessel that begun the transfer (line 4). Finally, the removed requests are reinserted using the best insertion method and the solution is returned (lines 5-6). Transfers may comprise of only a delivery or only a pickup node. For the first scenario, the delivery node is removed and the body continues on its route attached to the vessel. In this case, lines 2, 3 and 5 are not executed. When only removing a pickup, line 4 is skipped and the remainder of the algorithm functions as normal.

Removing a transfer may lead to infeasible solutions which violate vessel capacities. For instance, a vessel towing the maximum number of bodies may detach a body b_1 as its first action, leaving a capacity slack of one body. Later in the route, this same vessel may attach to a body b_2, again reaching its capacity limit. In case b_1's detachment would not be conducted, the previously feasible attachment of b_2 becomes infeasible as the vessel's capacity is violated. To remedy this, a removal in cascade is employed whereby

Algorithm 3: Body-transfer removal.

```
1 Input: solution $s$, body transfer to remove ($t_p$ and $t_d$), transferred body $b$
2 $U \leftarrow$ Remove every served request after node $t_p$ from $b$'s route (s)
3 Remove transfer pickup node $t_p$ (s)
4 Remove transfer delivery node $t_d$ (s)
5 Best insertion of all requests in $U$ (s)
6 return $s$
```

transfers which cause infeasibility are also removed. The two removal neighborhoods differ in terms of how exactly they select the body transfer to remove.

Random transfer removal (TR1). This neighborhood selects, with equal probability, a single body transfer to be removed from the solution. If after the removal the solution is infeasible due to vessel capacity, additional transfers are removed until a feasible solution is obtained.

Worst transfer removal (TR2). This neighborhood considers one body transfer at a time. The removal which yields the best solution value is maintained and the solution is returned with at least one fewer body transfer.

4.5. Routing phase

Once body transfers are fixed, vessel-body combinations may be considered as a single vehicle which have fixed visits scheduled in their routes. Note that the capacity of a vessel-body combination may change during the execution of a route given that body transfers may take place. The VSBR with fixed body transfers is therefore reduced to a PDPTW which must be addressed by the routing phase. This method optimizes vessel routes by rescheduling requests while respecting predefined body transfers. From the many existing routing algorithms, we selected one which is fast and simple to implement with proven performance on PDPTWs. SISRs (Christiaens and Vanden Berghe, 2020) was developed to be a general method for VRPs and experiments have shown that it generates competitive solutions for a wide range of variants, including the PDPTW.

SISRs comprises of a single ruin operator which removes customers based on a novel property called spatial slack. SISRs removes a sufficient number of customers in different yet geographically proximate routes, creating spatial and capacity slack. When reinserting customers back into the solution, multiple routes close to those customers offer a range of efficient options. The recreate operator comprises of a best insertion method (described in Section 4.3) which uses rank-based probabilities to avoid always inserting customers into the best position, thereby avoiding premature convergence. Each time the routing phase is invoked, SISRs is executed for a fixed number of iterations and returns an equal quality or improved solution.

5. Computational experiments

This section will investigate whether body transfers have a positive impact on solution quality despite the increase it brings to the problem's complexity and search space. Computational experiments using newly generated instances are performed with ILS-SB to study if it can efficiently employ body transfers and obtain high-quality solutions in reasonable runtimes. Additionally, an evaluation on how each proposed transfer neighborhood impacts solution quality will be performed. All experiments were conducted on a computer with an IntelXeon E5-2660 processor at 2.6 GHz and 164 GB of RAM running Ubuntu 18.04 LTS. ILS-SB was implemented in C++ and compiled using gcc 7.4.0 and options -O3.

All parameters required by ILS-SB are detailed in Table 3 and were calibrated using *irace* (López-Ibáñez et al., 2016) with the range of values provided in column *Range*. SISRs was implemented with the original parameters calibrated by Christiaens and Vanden Berghe (2020) for its best behavior across a range of instance sizes. The maximum number of iterations and time limit were set manually considering the trade-off between solution quality and processing time.

Table 3: ILS-SB parameters and values.

Parameter	Value	Range	Parameter	Value
ι	10	{5, 10,...,45, 50}	Time limit(s)	600
l	75	{10, 25, 50,..., 175, 200}	SISRs iterations	1000
ν	70	{10, 20, ..., 80, 90}		
α	0.70	{0.0,...,1.0}		
$\#maxIt$	80	{10, 20,...,140, 150}		

5.1. Instance sets

Given that the VSBR is a new problem there are no academic instances available and so, in order to perform experiments and encourage future research, instances are generated based on historical data from a shipping company. The data provided contains information concerning vessels and bodies, such as their speed, capacity, detaching/attaching time and container service time. Furthermore, a set of requests was provided along with the corresponding customer locations on a waterway network. The data provided to us is limited given the confidentiality terms agreed upon. For example, the company's executed schedules and routes were not disclosed. Thus a direct comparison with their results is unfortunately not possible. However, the instances we generated employ as much of the real data provided to us as possible in order to approximate the real scenario and provide valuable insights concerning the problem's characteristics. All benchmark instances are anonymized and have been made publicly available in the supplementary material of this paper.

We generated a total of 70 instances, divided into two instance sets. In the first set, *AttB*, each body is attached to a single vessel and each vessel has at least one body attached to it. In the second instance set, *DetB*, bodies may start detached and vessels may start empty (not towing any body). Each instance set has seven groups of five instances containing 10, 50, 100, 150, 200, 250 and 300 requests. For each instance we assumed a scheduling horizon of 20 days and ϵ equals to one while the other attributes were generated as follows.

Routing network. The employed routing network consists of 14 nodes with each node representing a container terminal where the pickup and/or delivery service of one or more requests must be served. Each of the nodes is connected to at least one other node via edges. Distances between nodes are calculated by employing a routing Application Programming Interface [2] which returns the shortest waterway trajectory

[2]`https://www.routino.org/`

between each pair of terminals. To facilitate the reading of instances, a distance matrix is provided for every instance, while the real location of nodes have been anonymized.

Vessels and bodies. The total number and capacity of available vessels and bodies corresponds to the real-world scenario. Since there is no depot or source/sink node, vessels may start at any location. For each vessel, a location in the network is selected randomly with uniform probability. The scheduling horizon starts given a "partial snapshot" of a solution, thus vessels may be en route between two nodes. To simulate this scenario, instances have a *start time* parameter for each vessel which corresponds to the earliest time it will be available at the first node of its route. Given two uniformly selected nodes n_1 and n_2, the start time of each vessel is generated between 0 and 50% of the travel time from n_1 to n_2. In the MIP model it is sufficient to fix $\tau^{Av}_{o_v}$ to vessel v's start time to consider this particular information. For the *DetB* instance set, bodies have a one-in-three chance to start detached from vessels. Bodies' start locations are selected with uniform probability by first considering only locations or vessels that do not have a body present or attached.

Requests. A single request is identified by a pickup and delivery service — each with their corresponding customer and time windows — and the number of containers ρ to be transported. Using the historical data we calculated the average and standard deviation of the number of containers associated with requests, the length of services time windows and amount of time overlap between pickup and delivery services of the same request. On average, requests involve seven containers with time windows of 4 and 7 days for pickup and delivery services, respectively. Pickup and delivery time windows overlap three days on average. All these attributes were generated based on a normal distribution considering the calculated average and standard deviation.

When creating an instance, first the number of containers ρ associated with each request is generated. Then, the locations of pickup and delivery services are generated uniformly, selecting a different location for each service. Starting with the pickup service, the size of its time window (TW^P_{size}) is generated followed by the opening day ($Open^P_{day}$), which is selected with uniform probability between -5 and 20-TW^P_{size}. If $Open^P_{day} < 0$, the request belongs to R^2 which means no pickup service is needed and the containers are inserted into a body (uniformly selected) with sufficient capacity.

After generating the pickup, the delivery service is generated starting with the size of its time window (TW^D_{size}). The opening day for the delivery service ($Open^D_{day}$) strongly depends on the opening day of the pickup service. If $Open^P_{day} < 0$, then $Open^D_{day}$ is selected with uniform probability between 0 and 20-TW^D_{size}. Otherwise, the considered range for $Open^D_{day}$ is 0-20. If $Open^D_{day} > 20 - TW^D_{size}$, the request belongs to R^3. In this case the delivery service will be left for the next scheduling horizon. For R^1 requests, the amount of time overlap is generated such that the delivery service may start as early as the pickup service or as late as when the pickup time window ends.

In the considered scenario, requests are associated with multiple containers and vessels can tow bodies capable of holding hundreds of such containers. Therefore, due to economies of scale, serving requests with vessels will always be considerably cheaper than serving them with trucks. This is because trucks typically only carry one or two containers and therefore either multiple trips or multiple trucks would be required to serve a single request. As a result, an unserved request incurs a significant outsourcing cost π which ensures that it is extremely unlikely to have an advantageous solution by leaving a request unserved. For the introduced benchmark, π was artificially set to 10,000 in order to simulate an expensive outsourcing cost.

Thus, a solution which serves more customers, even if it has a greater travel cost, is almost always going to be better than a solution which serves fewer customers. Note that the value of π may be adjusted depending on the problem context. Take, for example, a small-scale problem where bodies have less capacity and requests are associated with fewer containers. In this case π may require a lower value in order to capture the fact that outsourcing will make financial sense more frequently.

5.2. Detailed results

We first conduct experiments with the MIP formulation using Java 8 and CPLEX 12.6.3. Given that the model is unable to solve instances of realistic size within reasonable computational runtimes, the instances employed in these experiments have reduced size and were generated specifically for the MIP (with fewer vessels, bodies and requests). For the sake of diversity and to test the different constraints implemented in the model, we generated instances with R^1, R^2 and R^3 requests and instances where at least one request must be served by a truck. Table 4 presents the results of the experiments obtained by the MIP formulation and by ILS-SB. We introduce a time limit of five hours and a memory limit of 100 GB when solving the instances with the MIP.

The first columns of Table 4 provide details of the instance: the number of requests |R|, vessels |V| and bodies |B|. Regarding the MIP results, the table provides the number of nodes in the model network |N|, the upper bound (UB), lower bound (LB), processing time in seconds (T(s)) and number of unserved requests |U|. Given the non-deterministic nature of ILS-SB, each experiment is performed 10 times with a computational time limit of 10 minutes per run. The final columns of Table 4 document the results for ILS-SB and provide the value of the best solution obtained (S^*), the average processing time T(s), number of transfers (BT) and number of unserved requests |U|.

Table 4: MIP and ILS-SB comparison.

Instance			MIP					ILS-SB			
|R|	|V|	|B|	|N|	UB	LB	T(s)	|U|	S^*	T(s)	BT	|U|
3	2	2	14	10460	10460	30.41	1	10460	0.23	1.90	1
3	2	2	11	290	290	1.29	0	290	0.22	0.00	0
4	2	2	15	560	560	8.23	0	560	0.91	3.20	0
4	2	2	15	10460	10460	1625.73	1	10460	0.95	2.50	1
5	2	2	17	10490	10490	288.33	1	10490	3.50	1.40	1
5	2	2	17	10490	10490	557.88	1	10490	1.89	1.50	1
5	3	3	31	10587.99	350.11	18000.00	1	1448	7.74	2.90	0
6	3	3	35	30796.10	318.25	18000.00	3	1782	9.91	1.90	0

ILS-SB results (with exception of S^*) correspond to the average over 10 runs.

Table 4 reveals that although the MIP formulation is able to obtain the optimal solutions rather quickly for instances with two vessels and two bodies with up to five requests, CPLEX already experiences difficulty in solving problems with one more vessel and body. If the number of requests increases to 10 then even five hours of runtime is not enough to find a non-trivial solution (a solution that serves at least one request). Meanwhile, ILS-SB is able to obtain all proven optimal solutions and much better solutions for the final two instances in under 10 seconds.

To evaluate the statistical significance of the differences between methods, the Wilcoxon signed-rank test is performed when comparing results of two methods while the Pairwise T-test is performed in the neighborhood analysis, both with a confidence level of 95%. Table 5 details the results of ILS-SB for the *DetB* instance set and reports the value of the best solution found (S^*), the average solution values (S_{avg}), the average solution value considering the additional objective function (OF^2_{avg}), the average number of body transfers (BT) and unserved requests |U|, the average processing time in seconds T(s) and the average standard deviation (SD). In order to facilitate the presentation of results, different instances with the same number of requests were aggregated. Thus, each row corresponds to the average results across five instances.

Table 5: ILS-SB results for the *DetB* instance set.

Instance	S^*	S_{avg}	OF^2_{avg}	BT	|U|	T(s)	SD
I_10	2053.20	2066.20	46057.52	4.60	0.00	64.62	13.10
I_50	4690.80	4843.24	80532.28	9.64	0.00	605.05	112.17
I_100	6677.50	6980.60	84837.40	8.03	0.00	601.57	256.95
I_150	8765.40	9436.12	91706.00	6.36	0.00	603.41	596.51
I_200	12395.20	13325.08	103928.48	6.28	0.00	604.90	816.82
I_250	17455.80	18663.84	102816.80	5.56	0.00	607.19	917.07
I_300	28441.80	46144.16	120687.28	4.84	1.48	608.77	36374.70
Avg.	11497.10	14494.18	90080.82	6.47	0.21	527.69	5583.90

For this instance set, all instances with up to 250 requests had all of their requests served, while three I_300 instance resulted in a few unserved requests given that it becomes more difficult to serve more requests within the same length scheduling horizon. The number of unserved requests helps explain the larger standard deviation, as not serving customers leads to large outsourcing costs in the solution value. The maximum number of iterations without improvement was triggered only for instances with 10 requests, while the execution of all others was halted by the time limit. Doubling ILS-SB's time limit does not lead to significant difference concerning solution value. In addition, maintaining a short time limit enables the method to be adapted for a dynamic version of the problem, where, for example, it is uncertain when exactly requests become available.

Remaining with the results provided in Table 5, body transfers were inserted into every solution, with the results showing that a given body is transferred at most three times during the 20 days scheduling horizon. The average number of transfers in a solution considering all bodies is 6.47, meaning that on average 60% of the available bodies were transferred at least once. However, when only considering transfers between two different vessels, the average drops to 3.59, with the most significant differences occurring in the instances with 50 and 100 requests.

In order to evaluate the impact of body transfers on solution quality, a comparison is performed between the results of the complete ILS-SB and the corresponding PDPTW where body transfers are not considered. Thus, for the purposes of this experiment, only the routing phase of Algorithm 1 is used, with the resulting method referred to as ILS-R. To enable a fair comparison this experiment is conducted on the *AttB* instance set, where each body starts attached to a vessel. Since the routing phase will never schedule body transfers, no instances where bodies start detached are used, given that this would provide an unfair advantage to ILS-SB which can attach such bodies. Table 6 provides a summary of the results produced by ILS-R and documents the average solution value (S_{avg}), the average solution value considering the secondary objective function (OF^2_{avg}) and the average number of unserved requests |U|. The final row of the table provides the gap from the best average solution value and the best average solution value with respect to OF^2. These gaps are calculated for each instance using the best solution generated from both methods.

Table 6: Body transfers analysis.

	ILS-R			ILS-SB		
Instances	S_{avg}	OF^2_{avg}	|U|	S_{avg}	OF^2_{avg}	|U|
I_10	1836.80	73433.40	0.00	1775.92	65345.12	0.00
I_50	4932.32	99826.88	0.00	4741.44	90681.80	0.00
I_100	6838.23	108816.53	0.00	6750.93	92951.73	0.00
I_150	9037.20	122971.80	0.00	8685.12	107205.00	0.00
I_200	12105.76	120886.40	0.00	12088.68	108114.40	0.00
I_250	19361.44	126635.40	0.00	18295.16	115457.60	0.00
I_300	60487.16	158526.60	2.84	54219.80	144411.00	2.36
gap	4.79%	13.47%		0.33%	0.63%	

For the considered instance set, average solutions including body transfers are on average 4% better than those which do not. An even larger improvement is observed when comparing the values of OF^2 (13%), highlighting the even greater impact of body transfers on total operational cost. For best solution values, ILS-SB obtains solutions which are on average 6.5% better than those obtained by ILS-R. Despite the moderate average gap, a few outliers are observed which are worth mentioning. ILS-SB obtained solutions values 14%, 20%, 17% and 27% better than ILS-R for instances I_10_3, I_50_2, I_250_5 and I_300_2, respectively. ILS-R obtained better solutions for only three instances, resulting in the low positive ILS-SB gaps. Nevertheless, when comparing the results obtained by ILS-SB and ILS-R, significant statistical differences are observed regarding best solution, average solution and average OF^2 solution. Together, these results confirm the positive impact of permitting body transfers. Finally, it is worth noting that both algorithms were executed with the same time limit. This means that even with a far larger search space ILS-SB is able, within the same length of time, to significantly improve solution quality.

5.2.1. Additional analysis

ILS-SB comprises of six neighborhoods dedicated to the insertion/removal of body transfers. In order to identify the contribution of each neighborhood to solution quality, a set of experiments is conducted on the *DetB* instance set where a different neighborhood is deactivated in each experiment and the obtained solutions are compared. Table 7 provides the results for each neighborhood by dividing the objective function into two parts. Row *Travel cost* corresponds to the average travel cost of vessel routes, given by the number of edges used by vessels. Meanwhile, the average number of unserved requests is given by $|\text{U}|_{avg}$. The last row corresponds to the gap between the average solution value of the given neighborhoods versus the best average solution value considering all neighborhoods (gap(S_avg)). Column ILS-SB corresponds to the complete method with all neighborhoods, while each remaining column corresponds to the method without the labeled neighborhood. For instance, column TI1 corresponds to the results when the first insertion neighborhood is absent.

Although average solution quality always deteriorates when deactivating neighborhoods, significant statistical differences are present only between each method and TI4. The other neighborhoods did not significantly improve the solution, however each one of them is tailored to a unique way of performing body transfers and can prove valuable in different scenarios. Despite larger travel cost, ILS-SB obtains the smallest gap considering average solution value (2.93%). This result derives itself from the low number of unserved

Table 7: Solution values when using different sets of neighborhoods.

	ILS-SB	TI1	TI2	TI3	TI4	TR1	TR2
Travel cost	12432.33	12438.00	**12342.86**	12482.09	13706.50	12592.34	12659.30
$\|U\|_{avg}$	**0.21**	0.49	0.35	0.43	10.25	0.81	0.58
gap(S_{avg})	**2.93**	4.86	3.26	4.78	73.35	8.67	4.89

TI1: transfer at a customer location. TI2: transfer at transfer points. TI3: transfer at first node.
TI4: pick up originally detached body. TR1: random removal. TR2: worst removal.

requests (0.21 on average). Given the nature of the VSBR and the large outsourcing costs associated with unserved requests, a solution serving more requests, even if it has longer routes, will often be better than a solution serving fewer.

The large gaps between solutions with different sets of neighborhoods are concentrated in instances with more than 250 requests and primarily derive from the number of unserved requests. To demonstrate the impact of outsourcing costs on solution quality, Figure 5 provides the average solution value associated with the five instances containing 300 requests. The total value is divided into travel and outsourcing costs. All methods have similar travel costs, which is approximately 32000, but the total solution value varies depending on the number of unserved requests. The best solution is obtained by ILS-SB, which on average has 1.48 unserved requests. On the other hand, TI4 and TR1 have on average 26 and 4 unserved requests which results in solutions that are 84% and 43% worse, respectively.

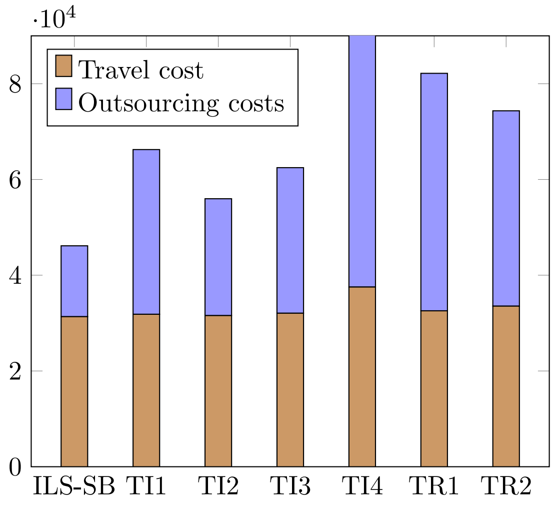


Figure 5: Solution values for the I_300 instances.

Note that despite the large number of unserved requests, TI4 still results in a greater average travel cost. TI4 is the only neighborhood capable of attaching bodies which began detached. When this particular neighborhood is deactivated, these bodies can no longer be attached. As a result, not only will requests which began loaded into these detached bodies remain unserved, but there is also less capacity available to serve requests since the initially detached bodies will not be used. For instances with up to 200 requests, TI4 obtains solutions with a low travel cost (average gap of 2.26). However, although the number of unserved requests for larger instances remains high, this does not lead to a decrease in travel cost. Due to the high number of requests to serve along with the reduced fleet of bodies (thus less total available capacity), more trips are necessary to serve requests which in turn leads to travel cost increases.

Figure 6 is a parallel-coordinates plot which illustrates the relationship between travel cost, the percentage of unserved requests |U|% and the total cost. For this experiment, a large number of solutions were generated by uniformly selecting the number of requests to remain unserved, while the other requests were randomly inserted into their first feasible position. Travel costs and total costs were normalized from 1-100 to better visualize the results. Note that the left-hand side of the figure has crossing edges, while the right-hand side demonstrates a near one-to-one direct correspondence. These results demonstrate that while a trade-off between travel cost and the percentage of unserved requests may occur, the proportion of unserved requests is directly connected to the total cost. This demonstrates the significant impact outsourcing costs have on the solution value and how more unserved requests will never result in lower total costs given the adopted value of π in the proposed instance set. Thus, it is always beneficial to serve more requests, despite the increase in travel cost.

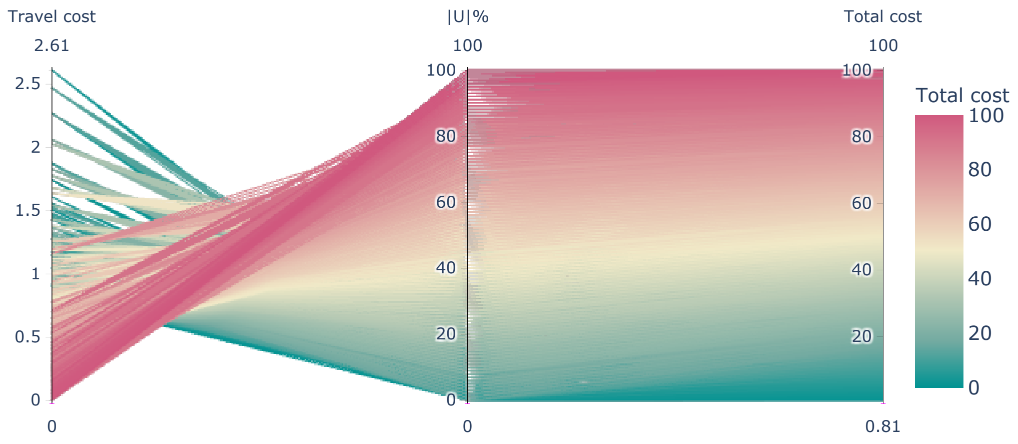


Figure 6: Relationship between travel cost, percentage of unserved requests and total cost.

Finally, an experiment is performed concerning the impact of employing OF^2. Breaking ties with respect to travel cost and directing the search towards solutions with less total operational cost improves overall solution quality by 5.9%. Although better solutions are obtained on average, statistical experiments show no significant difference between the results with and without the tie break regarding best and average solutions. However, significant differences are observed with respect to the average OF^2 solution values.

6. Conclusions

This paper introduced the vessel swap-body routing problem (VSBR), a generalization of the pickup and delivery problem with time windows. The problem is encountered in a shipping company where body transfers may be employed to reduce overall costs. These transfers significantly increase the solution space and difficulty of the problem given how they have a large impact on the number of routing and scheduling possibilities, confronting human operators with a significant challenge. The VSBR therefore represents a significant academic challenge with practical applications. We proposed a solution method for the problem and investigated whether body transfers can bring significant gains to solution quality when runtimes remain the same.

To facilitate the development of a solution method and to better tackle the multiple levels of decisions, we decomposed the problem into transfer decisions (when and at which locations to perform body transfers) and routing decisions. Once decomposed, these different decision levels were addressed using dedicated methods. We proposed a heuristic which consists of a state-of-the-art algorithm for the PDPTW and tailored neighborhoods for performing body transfers. Computational experiments on instances derived from real-world data indicated the positive impact of body transfers on both travel and outsourcing costs. The inclusion of body transfers lead to solutions with shorter vessel routes and more served customers, which directly impacted the overall cost of transportation. The significant gains obtained for a scheduling horizon of 20 days could translate into even larger gains for longer scheduling horizons.

This research indicated the potential benefit of transferring batches of requests between vehicles rather than handling the entirety of a request (pickup and delivery) with a single vehicle. These transfers give rise to an additional decision level which can be efficiently managed by decomposing the problem and employing dedicated methods for each decision level. Techniques to accelerate the solution method and better direct the search towards high-quality solutions should be incorporated as the search space significantly increases in size when handling these problems with multiple decisions levels. The foundations laid by this paper aim to encourage researchers to consider including the transfer of batches of requests in other VRP variants, as doing so has the potential to significantly improve solution quality in reasonable runtimes, despite the additional complexity.

Acknowledgements

This research was supported by Internal Funds KU Leuven (IMP/20/021) and by the Research Foundation Flanders (FWO) Strategic Basic Research project Data-driven logistics (S007318N). It was also supported by the National Counsel of Technological and Scientific Development (Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq) 408341/2018-1, Brazil. Editorial consultation provided by Luke Connolly (KU Leuven).

References

Battarra, M., Cordeau, J.-F., and Iori, M. (2014). Chapter 6: pickup-and-delivery problems for goods transportation. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 161–191. SIAM.

Burke, E. K. and Bykov, Y. (2017). The late acceptance hill-climbing heuristic. *European Journal of Operational Research*, 258(1):70–78.

Caramia, M. and Guerriero, F. (2010). A heuristic approach for the truck and trailer routing problem. *Journal of the Operational Research Society*, 61(7):1168–1180.

Chao, I.-M. (2002). A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, 29(1):33–51.

Christiaens, J., Çalik, H., Wauters, T., Chandrasekharan, R. C., and Vanden Berghe, G. (2020). The prisoner transportation problem. *European Journal of Operational Research*, 284(3):1058–1073.

Christiaens, J. and Vanden Berghe, G. (2020). Slack induction by string removals for vehicle routing problems. *Transportation Science*, 54(2):417–433.

Cordeau, J.-F., Laporte, G., and Ropke, S. (2008). Recent models and algorithms for one-to-one pickup and delivery problems. In *The vehicle routing problem: latest advances and new challenges*, pages 327–357. Springer.

Cortés, C. E., Matamala, M., and Contardo, C. (2010). The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *European Journal of Operational Research*, 200(3):711–724.

Cuda, R., Guastaroba, G., and Speranza, M. G. (2015). A survey on two-echelon routing problems. *Computers & Operations Research*, 55:185–199.

Danloup, N., Allaoui, H., and Goncalves, G. (2018). A comparison of two meta-heuristics for the pickup and delivery problem with transshipment. *Computers & Operations Research*, 100:155–171.

Derigs, U., Pullmann, M., and Vogel, U. (2013). Truck and trailer routing—problems, heuristics and computational experience. *Computers & Operations Research*, 40(2):536–546.

Drexl, M. (2012). Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints. *Transportation Science*, 46(3):297–316.

Drexl, M. (2013). Applications of the vehicle routing problem with trailers and transshipments. *European Journal of Operational Research*, 227(2):275–283.

Drexl, M. (2021). On the one-to-one pickup-and-delivery problem with time windows and trailers. *Central European Journal of Operations Research*, 29(3):1115–1162.

Lin, S.-W., Vincent, F. Y., and Chou, S.-Y. (2009). Solving the truck and trailer routing problem based on a simulated annealing heuristic. *Computers & Operations Research*, 36(5):1683–1692.

Lin, S.-W., Vincent, F. Y., and Lu, C.-C. (2011). A simulated annealing heuristic for the truck and trailer routing problem with time windows. *Expert Systems with Applications*, 38(12):15244–15252.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., and Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2003). Iterated local search. In *Handbook of metaheuristics*, pages 320–353. Springer.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2019). Iterated local search: Framework and applications. In *Handbook of metaheuristics*, pages 129–168. Springer.

Masson, R., Lehuédé, F., and Péton, O. (2013a). An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science*, 47(3):344–355.

Masson, R., Lehuédé, F., and Péton, O. (2013b). Efficient feasibility testing for request insertion in the pickup and delivery problem with transfers. *Operations Research Letters*, 41(3):211–215.

Masson, R., Lehuédé, F., and Péton, O. (2014). The dial-a-ride problem with transfers. *Computers & Operations Research*, 41:12–23.

Mitrović-Minić, S. and Laporte, G. (2006). The pickup and delivery problem with time windows and transshipment. *INFOR: Information Systems and Operational Research*, 44(3):217–227.

Öncan, T., Altınel, I. K., and Laporte, G. (2009). A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research*, 36(3):637–654.

Parragh, S. N. and Cordeau, J.-F. (2017). Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows. *Computers & Operations Research*, 83:28–44.

Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2008). A survey on pickup and delivery problems (part ii: Transportation between pickup and delivery locations). *Journal für Betriebswirtschaft*, 58(2):81–117.

Prodhon, C. and Prins, C. (2014). A survey of recent research on location-routing problems. *European Journal of Operational Research*, 238(1):1–17.

Qu, Y. and Bard, J. F. (2012). A grasp with adaptive large neighborhood search for pickup and delivery problems with transshipment. *Computers & Operations Research*, 39(10):2439–2456.

Rais, A., Alvelos, F., and Carvalho, M. S. (2014). New mixed integer-programming model for the pickup-and-delivery problem with transshipment. *European Journal of Operational Research*, 235(3):530–539.

Rothenbächer, A.-K., Drexl, M., and Irnich, S. (2018). Branch-and-price-and-cut for the truck-and-trailer routing problem with time windows. *Transportation Science*, 52(5):1174–1190.

Savelsbergh, M. W. (1992). The vehicle routing problem with time windows: Minimizing route duration. *ORSA journal on computing*, 4(2):146–154.

Scheuerer, S. (2006). A tabu search heuristic for the truck and trailer routing problem. *Computers & Operations Research*, 33(4):894–909.

Shiri, H., Rahmani, M., and Bafruei, M. (2020). Examining the impact of transfers in pickup and delivery systems. *Uncertain Supply Chain Management*, 8(1):207–224.

Takoudjou, R. T., Deschamps, J.-C., and Dupas, R. (2012). A mip formulation for the pickup and delivery problem with time window and transshipment. *IFAC Proceedings Volumes*, 45(6):333–338.

Toffolo, T. A., Christiaens, J., Van Malderen, S., Wauters, T., and Vanden Berghe, G. (2018). Stochastic local search with learning automaton for the swap-body vehicle routing problem. *Computers & Operations Research*, 89:68–81.

Villegas, J. G., Prins, C., Prodhon, C., Medaglia, A. L., and Velasco, N. (2013). A matheuristic for the truck and trailer routing problem. *European Journal of Operational Research*, 230(2):231–244.

Wolfinger, D. (2021). A large neighborhood search for the pickup and delivery problem with time windows, split loads and transshipments. *Computers & Operations Research*, 126:105110.

Zhang, Y., Atasoy, B., Souravlias, D., and Negenborn, R. R. (2020). Pickup and delivery problem with transshipment for inland waterway transport. In *International Conference on Computational Logistics*, pages 18–35. Springer.