

An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines

Citation for published version (APA):

Ghilas, V., Demir, E., & Woensel, van, T. (2015). *An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines*. (BETA publicatie : working papers; Vol. 483). Technische Universiteit Eindhoven.

Document status and date:

Published: 01/01/2015

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows and Scheduled Lines

Veaceslav Ghilas
Emrah Demir
Tom Van Woensel

Beta Working Paper series 483

BETA publicatie	WP 483 (working paper)
ISBN	
ISSN	
NUR	804
Eindhoven	June 2015

An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows and Scheduled Lines

Veaceslav Ghilas*, Emrah Demir, Tom Van Woensel

*School of Industrial Engineering,
Operations, Planning, Accounting and Control (OPAC),
Eindhoven University of Technology,
Eindhoven, 5600 MB, The Netherlands*

Abstract

The Pickup and Delivery Problem with Time Windows and Scheduled Lines (PDPTW-SL) concerns scheduling a set of vehicles to serve freight requests such that a part of the journey can be carried out on a scheduled public transportation line using bus, train, tram, metro, etc. Due to the complexity of the problem, which is NP-hard, we propose an Adaptive Large Neighborhood Search (ALNS) heuristic algorithm to solve the PDPTW-SL. Complex aspects such as fixed lines' schedules, synchronization and time-windows constraints are efficiently considered in the proposed algorithm. Results of extensive computational experiments show that the ALNS is highly effective in finding good-quality solutions on the generated PDPTW-SL instances with up to 100 freight requests that reasonably represent real life situations.

Keywords: Freight transportation, Pickup and delivery problem, Heuristic algorithm, Scheduled lines

1. Introduction

A successful integration of passenger and freight transportation creates a seamless movement of people and freight. This integration achieves socially desirable transport options economically viable in urban areas as it reduces the impact of congestion and air pollution (Lindholm and Behrends [21]). Actual integration is already being observed in long-haul freight transportation (e.g., passenger aircraft and ferries). Norwegian Hurtigruten carries freight and people efficiently and seamlessly in the region of Northern Europe (Levin et al. [18], Hurtigruten [16]). However, short-haul passenger and freight (i.e., small packages) transportation is rarely integrated, although these services largely use the same infrastructure.

This paper investigates opportunities and the feasibility of using available public transportation vehicles, which operate according to predetermined routes and schedules, for transporting freight.

Email address: V.Ghilas@tue.nl, E.Demir@tue.nl, T.v.Woensel@tue.nl (Veaceslav Ghilas*, Emrah Demir, Tom Van Woensel)

* Corresponding author, Tel. +31 40 247 4984

A successful synchronization of pickup and delivery (PD) vehicles with scheduled lines (SLs) is directly related to coordination and consolidation of transport. *Coordination* assures the precision and timing of each leg's movement. On the other hand, *consolidation* implies that the amount of freight transferred to/from SLs fits with capacities of PD and SL vehicles. Considering both coordination and consolidation of such integrated system, we introduce the *Pickup and Delivery Problem with Time Windows and Scheduled Lines* (PDPTW-SL).

There exists a rich literature on PDP-related problems. The interested readers are referred to Berbeglia et al. [3] for a literature survey on the solution methodologies for PDPs. A special case of PDPs where passengers need to be transported from their origin to their destination by considering service level constraints are called dial-a-ride problems (DARP). An overview on the mathematical models and the solution methodologies for single as well as multiple-vehicle DARPs can be found in Cordeau and Laporte [6].

In another extension of the PDP, it is possible to consider transfer opportunity between PD vehicles at predefined transfer nodes (PDP-T). Shang and Cuff [30] proposed an insertion heuristic algorithm to solve instances with up to 167 requests for the PDP-T. Cortes et al. [7] proposed an exact decomposition method, namely Branch-and-Cut (B&C), and solved PDP-T instances with up to six requests. Moreover, Masson et al. [22, 23] proposed an ALNS to solve PDP-T and DARP-T, respectively. In both cases, the authors solved instances with up to 100 requests. The PDP with cross-docking opportunities has been studied by Petersen and Røpke [25], who proposed a large neighborhood search heuristic to solve instances with up to 982 requests.

To the best our knowledge, Nash [24] is the first author who conceptually introduced the idea of transporting freight using public transportation. One of the first attempts to combine scheduled line services with DARP was done by Liaw et al. [20]. The authors formulated the problem where transfers to public scheduled transportation are allowed for passengers and wheelchair persons. The authors proposed two types of heuristics (i.e., online and offline) and solved instances with up to 120 requests. Later, Aldaihani and Dessouky [1] considered an integrated DARP with public transport and proposed a two-stage heuristic algorithm to solve instances with up to 155 requests. Hall et al. [15] introduced a mixed-integer program to solve an integrated dial-a-ride problem (IDARP). The authors considered transfers to fixed lines without modeling schedules of the public transportation. The authors solved small-sized instances with up to 4 requests. Trentini and Malhene [33] presented a review of solutions for combining freight and passenger transportation used in practice. In a related study, Trentini et al. [34] investigated a two-echelon VRP with transshipment in the context of passenger and freight integrated system. The authors proposed an ALNS to solve instances with 50 customers.

As a real-life application of the integrated transport system, a project, called *City Cargo Amsterdam* [5], was held as a pilot experiment in the Netherlands in 2007. Two cargo trams were used to transport freight in the city centre of Amsterdam. The goods were transferred from cargo trucks onto trams to be delivered to the inner part of the city. In 2009 the project was quit due to lack of funds.

In this paper, we consider two transport scenarios: (i) an integrated transportation system and, (ii) the pickup and delivery problem where transfers to scheduled lines are not allowed. These two scenarios are intended to provide plausible and sufficient reasons to implement such system

by comparing them in terms of the total operating costs of the PD vehicles. The PDPTW-SL is NP-hard since it is an extension of the classical PDPTW. It is shown that only a limited number of requests can be solved to optimality within a reasonable time limit. For this reason, we have developed a metaheuristic to obtain good-quality solutions. More specifically, we propose an ALNS heuristic algorithm which is based on the destroy and re-create principle.

The contribution of the paper is three-fold: (i) to adapt the classical ALNS heuristic algorithm to solve PDPTW-SL, and (ii) to efficiently handle synchronization constraints within the proposed ALNS framework and finally, (iii) to analyze the cost savings due to the use of scheduled lines in freight transportation. The remainder of this paper is organized as follows. In Section 2, we present a mathematical formulation of the PDPTW-SL. Section 3 describes the proposed ALNS heuristic algorithm for the PDPTW-SL. Section 4 presents the results of extensive computational experiments. Conclusions are stated in Section 5.

2. The pickup and delivery problem with time windows and scheduled lines

The PDPTW-SL is an extension of the standard PDPTW that additionally considers the flexibility of using available scheduled lines, such as bus, train, metro, etc. Due to the fact that public transportation systems have a certain coverage (i.e., rural or urban), some delivery trips of the PD vehicles may overlap with SL services. Thus, using public transportation as a part of freight transportation may lead to cost and environmental benefits for the whole transportation system. For instance, due to less driving time of the PD vehicles, logistics service providers may experience substantial operating cost savings. As a consequence, less traveling time of the vehicles also leads to fewer carbon dioxide-equivalent (CO_2e) emissions at the global level for the whole society Demir et al. [9, 11]. Furthermore, using scheduled lines for carrying freight gives extra cost benefits for public transport service providers as the utilization of SL services increases. An example application which is based on the current metro system of Amsterdam can be illustrated as in Figure 1.

The system depicted in Figure 1 contains four scheduled lines, and five transfer nodes (end-of-lines stations). Note that T_1^{51} , T_1^{53} , T_1^{54} as well as T_2^{50} and T_2^{54} can be merged into two transfer nodes as these denote the same locations. Each metro line has one single route with one destination in each direction. To use this metro system as a part of the transportation plan, package(s) can be delivered to one of its end-of-line stations and transported to other side of the line considering available capacity and schedules of the metro line.

2.1. A formal description of the PDPTW-SL

In this section we give a formal description of the PDPTW-SL. All information (e.g., requests, demands, travel times, time windows) is considered known beforehand, and a possible transport plan for the whole planning horizon (e.g., one day) should be generated before the execution of the transport activities. A solution to the problem is a routing plan and schedules for both requests and PD vehicles. We now define the PDPTW-SL on a digraph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where \mathcal{N} represents the set

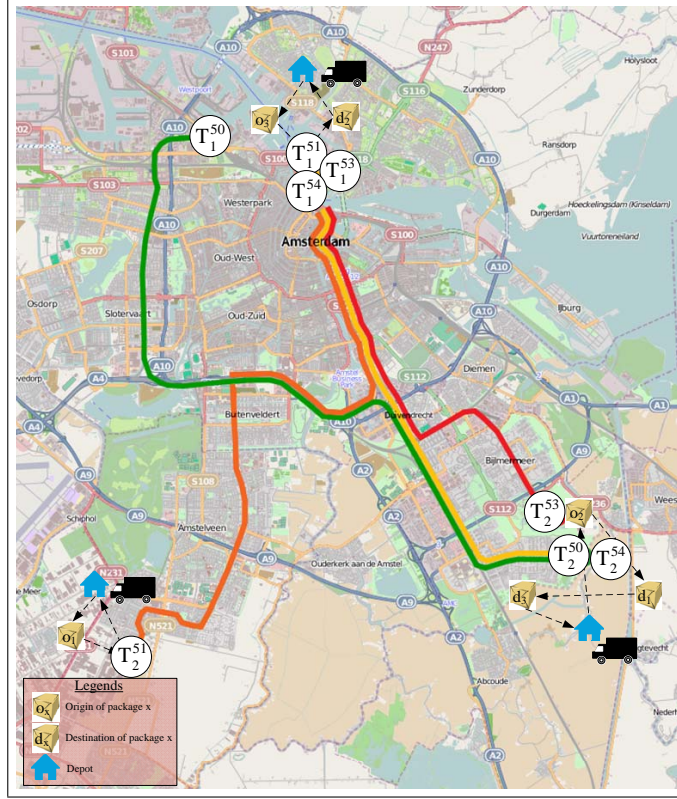


Figure 1 An illustration of the metro system in Amsterdam (2014)

of nodes (i.e., depots, pickup, delivery and replicated transfer nodes³) and \mathcal{A} represents the set of arcs. We note that i, j, r, v, w , and t are used as indices in this paper.

- *Request*: Each request $r \in \mathcal{P}$ is associated with an origin node r and a destination node $r + n$, where n is the number of requests to be satisfied. Each request is associated with two desired time windows: one for the origin ($[l_r, u_r]$), and one for the destination point ($[l_{r+n}, u_{r+n}]$). Furthermore, demand d_r is known for each request.
- *PD Vehicle*: A set of PD vehicles is given by \mathcal{V} . In addition, each vehicle $v \in \mathcal{V}$ is associated with the carrying capacity Q_v , the depot (origin and destination) o_v , and time window $[l_{o_v}, u_{o_v}]$.
- *Travel and service time*: Travel and service times are known beforehand and remain unchanged during the planning horizon. Each arc $(i, j) \in \mathcal{A}$ is associated with travel time Υ_{ij} . The service time at node $i \in \mathcal{N}$ is represented as s_i .
- *Scheduled line*: A set of all physical transfer nodes is given as \mathcal{S} and the set of all physical scheduled lines is given as \mathcal{E} , which is defined by the directed arc between the start and the end of the line (i, j) , such that $i, j \in \mathcal{S}$. For example, between two transfer nodes i and j , there are two scheduled lines considered that are one arc for each direction, (i, j) and (j, i) . Each scheduled line has a set of indices \mathcal{K}^{ij} for the departure times from i (the start of fixed line), such that the departure time is given as $p_{ij}^w, \forall (i, j) \in \mathcal{E}, w \in \mathcal{K}^{ij}$ (e.g., $p_{T_1, T_2}^0 = 30$). We note that each scheduled line may have different frequencies than other lines, thus the size

³see more detailed explanation at page 5

of the \mathcal{K}^{ij} may differ. Furthermore, it is assumed that SL vehicles are designed to carry a limited number of packages, thus implying a finite carrying capacity $k_{ij}, \forall (i, j) \in \mathcal{E}$.

We state the following additional assumptions related to SL services.

- The package carrying capacity is not influenced by the passenger demand on SL services.
- Each transfer node is considered as a coordination and a consolidation point for requests (e.g., DHL-Packstation [12]). In other words, a storage space for to-be-shipped packages is available at the transfer nodes. The packages can be stored until the vehicle's departure time (i.e., scheduled line or PD). Furthermore, it is considered that multiple PD vehicles can be serviced simultaneously at a transfer node. In other words, we assume that the loading/unloading infrastructure is always available for multiple vehicles at a time.
- In case of multiple freight carriers using SL services, it is assumed that each carrier has a limited storage space at the transfer node and on the SL vehicle (e.g., contract-based agreement). Hence, it is assured that the considered capacity is not affected by the demand of other actors involved.
- It is assumed that cost η_{ij} per each unit of parcels shipped on the SL (i, j) includes transportation, handling (transshipment) and storage costs. Handling of the packages during transshipment is assumed to be under the responsibility of the SL service provider (e.g., the driver of the SL vehicle).
- Since the focus of the present problem is related to the operational-level aspects, we disregard all investments needed, such as the re-design of the SL vehicle and the physical storage space required at the transfer nodes. Obviously this assumption may affect the outcome of the system (i.e., might lead to less benefits).
- It is assumed that all data is known beforehand. In practice, this might not be the case because of the uncertainty in travel times and demands. Disregarding these aspects in the planning process might eventually lead to infeasible plans and expensive repair (recourse) actions to tackle such violations.

In order to model waiting times of the PD vehicles and multiple visits at the transfer nodes, each physical scheduled line (i.e., $(i, j) \in \mathcal{E}$, e.g., arcs $(1, 2)$ and $(2, 1)$ in Figure 2a) is replicated n times as in Hall et al. [15]. In addition, each replication is assigned to one request, and only that specific request can travel on the assigned scheduled line (see Figure 2b). This is done to reduce the number of decision variables. Therefore, the set of all replicated scheduled lines is given as \mathcal{F} (e.g., $(1a, 2a)$, $(1b, 2b)$, $(2a, 1a)$ and $(2b, 1b)$ in Figure 2b).

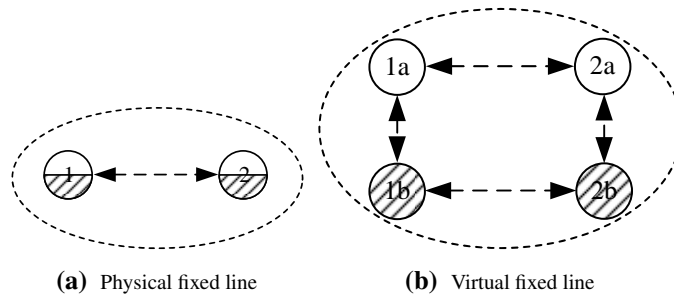


Figure 2 An illustration of a replicated fixed line

Furthermore, the replication process means that each physical transfer node in \mathcal{S} (e.g., nodes 1 and 2 in Figure 2a) is replicated n times (e.g., $\mathcal{T} \equiv \{1a, 1b, 2a, 2b\}$ in the considered example). For the sake of modeling let $\psi^t, \forall t \in \mathcal{T}$ be the physical transfer node represented by the replicated transfer node t (e.g., $\psi^{1a} = \psi^{1b} = 1$). Consequently, set \mathcal{T}^t can be defined as $\{i \in \mathcal{T} | \psi^i = \psi^t, i \neq t\}, \forall t \in \mathcal{T}$ (e.g., in Figure 2b, $\mathcal{T}^{2a} = \{2b\}$, as 2a and 2b represent physical transfer node 2).

The proposed model is not limited to only one fixed line (e.g., (1, 2) and (2, 1)). Different topologies may be considered, such as square, triangle, star or any other complex networks. Furthermore, the requests are allowed to be shipped from one scheduled line to the other. The additional parameters used in our model are given in Table 1.

Table 1

Parameters

Notation	Definition
δ	Number of depots
d_r	Demand of request r
Υ_{ij}	Traveling time from node i to node j
s_i	Service time at node i
k_{ij}	Freight carrying capacity on the fixed line (i, j)
Q_v	Carrying capacity of the vehicle v
$[l_i, u_i]$	Time window of node i
p_{ij}^v	Departure time from transfer node i on the scheduled line (i, j) , indexed by v
ψ^t	Physical transfer node that is represented by replicated transfer node t
τ	Number of replicated transfer nodes (i.e., $ \mathcal{T} $)
f_i^r	$\begin{cases} 1 & \text{if node } i \text{ is } r, \\ 0 & \text{if node } i \in \mathcal{N}_2 \setminus \{r; r+n\} \text{ (see Table 2),} \\ -1 & \text{if node } i \text{ is } r+n. \end{cases}$
θ_v	The routing cost per one time unit of vehicle v
η_{ij}	The cost of shipping one unit of package on the SL (i, j)

Using the parameters above, the notation of sets is given in Table 2.

Table 2

Sets

Notation	Definition
\mathcal{V}	Set of PD vehicles
\mathcal{O}	Set of depots, $\mathcal{O} \equiv [1, \dots, \delta]$ (i.e. $o_v \in \mathcal{O}, \forall v \in \mathcal{V}$)
\mathcal{P}	Set of requests or pickup nodes, $\mathcal{P} \equiv [\delta + 1, \dots, \delta + n]$
\mathcal{D}	Set of delivery nodes, $\mathcal{D} \equiv [\delta + n + 1, \dots, \delta + 2n]$
\mathcal{T}	Set of replicated transfer nodes, $\mathcal{T} \equiv [\delta + 2n + 1, \dots, \delta + 2n + n\tau]$ (see nodes 1a, 1b, 2a and 2b in Figure 2b)
\mathcal{T}^t	Set of replicated transfer nodes associated the same physical transfer node as t (e.g. in Figure 2b, $\mathcal{T}^{2a} = \{2b\}$, $\mathcal{T}^{1a} = \{1b\}$, etc.)
\mathcal{N}	Set of nodes in the graph \mathcal{G} ; $\mathcal{P} \cup \mathcal{D} \cup \mathcal{O} \cup \mathcal{T} \equiv \mathcal{N}$
\mathcal{N}_1	Set of nodes that are related to requests ($\mathcal{P} \cup \mathcal{D}$)
\mathcal{N}_2	Set of nodes that represent requests and replicated transfer nodes ($\mathcal{P} \cup \mathcal{D} \cup \mathcal{T}$)
\mathcal{E}	Set of physical SLs which is defined as (i, j) , with associated \mathcal{K}^{ij} and k_{ij}
\mathcal{K}^{ij}	Set of indices for the departure times from the physical transfer node i on SL $(i, j) \in \mathcal{E}$
\mathcal{F}	Set of replicated SLs which is defined as (i, j) with associated $\mathcal{K}^{\psi^i \psi^j}$
\mathcal{F}^r	Set of replicated SLs associated with request r (e.g. in Figure 2, $\mathcal{F}^a = \{(1a, 2a), (2a, 1a)\}$)
\mathcal{F}^t	Set of replicated SLs connected to the replicated transfer node t (e.g. in Figure 2, $\mathcal{F}^{1a} = \{(1a, 2a), (2a, 1a)\}$)
\mathcal{F}^{ij}	Set of replicated SLs associated with a physical SL $(i, j) \in \mathcal{E}$ (e.g. in Figure 2, $\mathcal{F}^{1,2} = \{(1a, 2a), (1b, 2b)\}$, $\mathcal{F}^{2,1} = \{(2a, 1a), (2b, 1b)\}$)
\mathcal{A}	Set of arcs in \mathcal{G} defined by $\mathcal{N} \times \mathcal{N}$, (note that $\mathcal{A} \setminus \mathcal{A}^1 \equiv \mathcal{F} \cup \{(i, j) i \in \mathcal{O}, j \in \mathcal{N}_2\} \cup \{(i, j) i \in \mathcal{N}_2, j \in \mathcal{O}\}$)
\mathcal{A}^1	$= \mathcal{N}_2 \times \mathcal{N}_2 \setminus \mathcal{F}$

The decision variables used to handle routing and scheduling of the PD vehicles, along with the flow and the timing of the requests are given in Table 3.

Table 3

Decision variables

Notation	Definition
x_{ij}^v	A binary variable equal to 1 if arc (i, j) is used by PD vehicle v , 0 otherwise, $\forall (i, j) \in \mathcal{A}, v \in \mathcal{V}$
α_v	A continuous variable which shows the time at which vehicle v returns to its depot, $\forall v \in \mathcal{V}$
β_i	A continuous variable which shows the departure time of a vehicle from node i , $\forall i \in \mathcal{N}$
y_{ij}^r	A binary variable equal to 1 if arc (i, j) is used by request r , 0 otherwise, $\forall i, j \in \mathcal{N}_2, r \in \mathcal{P}$
γ_i^r	A continuous variable which shows the departure time of request r from node i , $\forall i \in \mathcal{N}_2, r \in \mathcal{P}$
q_{ij}^{rw}	A binary variable equal to 1 if replicated fixed line (i, j) is used by request r that departs from i at time p_{ij}^w , 0 otherwise, $\forall r \in \mathcal{P}, (i, j) \in \mathcal{F}^r, w \in \mathcal{K}^{\psi^i \psi^j}$

2.2. Mathematical formulation of the PDPTW-SL

The PDPTW-SL can be formalized as the following mixed-integer program.

$$\min \sum_{(i,j) \in \mathcal{A}} \sum_{v \in \mathcal{V}} \theta_v \Upsilon_{ij} x_{ij}^v + \sum_{r \in \mathcal{P}} \sum_{(i,j) \in \mathcal{F}^r} \sum_{w \in \mathcal{K}^{\psi^i \psi^j}} \eta_{ij} d_r q_{ij}^{rw} \quad (1)$$

Term (1) minimizes the total travel cost of the PD vehicles and the cost of using SLs for transferred requests.

subject to

Routing and flow constraints

$$\sum_{i \in \mathcal{N}} \sum_{v \in \mathcal{V}} x_{ij}^v = 1 \quad \forall j \in \mathcal{N}_1 \quad (2)$$

$$\sum_{i \in \mathcal{N}_2} x_{o_v, i}^v \leq 1 \quad \forall v \in \mathcal{V} \quad (3)$$

$$\sum_{i \in \mathcal{N}} \sum_{v \in \mathcal{V}} x_{it}^v \leq 1 \quad \forall t \in \mathcal{T} \quad (4)$$

$$\sum_{j \in \mathcal{N}} x_{ij}^v - \sum_{j \in \mathcal{N}} x_{ji}^v = 0 \quad \forall i \in \mathcal{N}, v \in \mathcal{V} \quad (5)$$

$$\sum_{j \in \mathcal{N}_2} y_{ij}^r - \sum_{j \in \mathcal{N}_2} y_{ji}^r = f_i^r \quad \forall r \in \mathcal{P}, i \in \mathcal{N}_2 \quad (6)$$

$$\sum_{i \in \mathcal{N}} \sum_{v \in \mathcal{V}} x_{it}^v \leq \sum_{r \in \mathcal{P}} \sum_{(i,j) \in \mathcal{F}^r} y_{ij}^r \quad \forall t \in \mathcal{T} \quad (7)$$

Constraints (2) assure that all pickup and delivery nodes are visited exactly once. Constraints (3) ensure that each vehicle leaves its depot at most once and Constraints (4) assure that each replicated transfer node is visited at most once. Flow conservation for PD vehicles is considered in Constraints (5). Constraints (6) assure flow conservation for the paths of each request. Constraints (7) ensure that if a request uses a scheduled line, a PD vehicle should pick it up/drop it off at a transfer node related to that specific SL.

Scheduling constraints

$$y_{ij}^r = 1 \implies \gamma_j^r \geq \gamma_i^r + \Upsilon_{ij} + s_j \quad \forall r \in \mathcal{P}, i, j \in \mathcal{N}_2 \quad (8)$$

$$\sum_{v \in \mathcal{V}} x_{ij}^v = 1 \implies \beta_j \geq \beta_i + \Upsilon_{ij} + s_j \quad \forall i \in \mathcal{N}, j \in \mathcal{N}_2 \quad (9)$$

$$x_{i,o_v}^v = 1 \implies \alpha_v \geq \beta_i + \Upsilon_{i,o_v} + s_{o_v} \quad \forall i \in \mathcal{N}_2, v \in \mathcal{V} \quad (10)$$

$$\beta_{r+n} \geq \beta_r + \Upsilon_{r,r+n} + s_{r+n} \quad \forall r \in \mathcal{P} \quad (11)$$

$$l_i \leq \beta_i - s_i \leq u_i \quad \forall i \in \mathcal{N}_1 \quad (12)$$

$$l_{g_v} \leq \alpha_v \leq u_{o_v} \quad \forall v \in \mathcal{V} \quad (13)$$

$$\sum_{w \in \mathcal{K}^{\psi^i \psi^j}} q_{ij}^{rw} = y_{ij}^r \quad \forall r \in \mathcal{P}, (i, j) \in \mathcal{F}^r \quad (14)$$

$$q_{ij}^{rw} = 1 \quad \text{and} \quad y_{ij}^r = 1 \implies \gamma_i^r = p_{ij}^w \quad \forall r \in \mathcal{P}, (i, j) \in \mathcal{F}^r, w \in \mathcal{K}^{\psi^i \psi^j} \quad (15)$$

$$\sum_{r \in \mathcal{P}} \sum_{(a,b) \in \mathcal{F}^{ij}} d_r q_{ab}^{rw} \leq k_{ij} \quad \forall (i, j) \in \mathcal{E}, w \in \mathcal{K}^{ij} \quad (16)$$

$$\sum_{r \in \mathcal{P}} d_r y_{ij}^r \leq \sum_{v \in \mathcal{V}} Q_v x_{ij}^v \quad \forall (i, j) \in \mathcal{A}^1 \quad (17)$$

Timing for each request is considered in Constraints (8). Similarly for PD vehicles, scheduling is updated in Constraints (9) and (10). Constraints (11) assure precedence relationship of each request. Constraints (12) and (13) force the time windows to be respected. Constraints (14) – (15) assure that if a request uses a scheduled line, it departs at a scheduled departure time. Constraints (16) ensure that package carrying capacity on the scheduled line is not exceeded. Constraints (17) consider the capacity of each PD vehicle.

Synchronization constraints

$$\sum_{j \in \mathcal{N}_1} y_{ij}^r = 1 \implies \gamma_i^r = \beta_i \quad \forall r \in \mathcal{P}, i \in \mathcal{T} \quad (18)$$

$$\sum_{j \in \mathcal{N}_2} y_{ij}^r = 1 \implies \gamma_i^r = \beta_i \quad \forall r \in \mathcal{P}, i \in \mathcal{N}_1 \quad (19)$$

$$\sum_{i \in \mathcal{N}_2} y_{i,r+n}^r = 1 \implies \gamma_{r+n}^r = \beta_{r+n} \quad \forall r \in \mathcal{P} \quad (20)$$

$$y_{ij}^r = 1 \implies \gamma_t^r = \beta_t \quad \forall r \in \mathcal{P}, t \in \mathcal{T}, j \in \mathcal{T}^t \quad (21)$$

The set of constraints (18) – (21) ensure the synchronization between requests' and vehicles' schedules. Constraints (18) force departure times of requests and vehicles from a replicated transfer node to be equal if there is a request flow from that node towards a pickup/delivery node. Constraints (19) force departure times of requests and vehicles from a pickup/delivery node to be equal if there is a request flow from that node. Constraints (20) force arrival time at the destination node of a

request be equal to departure time of a vehicle from that node. Constraints (21) assure time synchronization between vehicles and requests at each transfer node, with regards to flow between replications of the same original transfer node.

Decision variable domains

$$x_{ij}^v \in \{0, 1\} \quad \forall (i, j) \in \mathcal{A}, v \in \mathcal{V} \quad (22)$$

$$y_{ij}^r \in \{0, 1\} \quad \forall i, j \in \mathcal{N}_2, r \in \mathcal{P} \quad (23)$$

$$\alpha_v \in \mathbb{R}^+ \quad \forall v \in \mathcal{V} \quad (24)$$

$$\gamma_i^r \in \mathbb{R}^+ \quad \forall i \in \mathcal{N}_2, r \in \mathcal{P} \quad (25)$$

$$\beta_i \in \mathbb{R}^+ \quad \forall i \in \mathcal{N} \quad (26)$$

$$q_{ij}^{rw} \in \{0, 1\} \quad \forall r \in \mathcal{P}, (i, j) \in \mathcal{F}^r, w \in \mathcal{K}^{\psi^i \psi^j} \quad (27)$$

Note that Constraints (8) – (10), (15) and (18) – (21) are formulated as implications, and standard linearization techniques can be used to express them using one or two linear inequalities.

3. An adaptive large neighborhood search heuristic algorithm for the PDPTW-SL

The proposed metaheuristic is an extension of the Large Neighborhood Search (LNS) heuristic, which is first proposed by Shaw [31]. LNS is based on the idea of gradually improving an initial solution by using both destroy and repair neighborhood operators. In other words, LNS consists of a series of removal and insertion moves. If a new solution is better than the current best solution, the algorithm replaces the current solution and uses this new current solution as an input in the next iteration.

ALNS heuristic framework was first introduced in Pisinger and Røpke [26], Røpke and Pisinger [29], Pisinger and Røpke [27] to solve several vehicle routing problems. Instead of using one large neighborhood as in LNS, the ALNS applies several removal and insertion operators to a given solution. The neighborhood of a given set of feasible routes is investigated by removing some requests and reinserting them back to the solution. The removal and insertion operators are dynamically selected according to their past and current performance. To this end, each operator is assigned a *probability* of being selected. The new solution is accepted if it satisfies some criteria defined by the metaheuristic framework (e.g., simulated annealing, tabu search). In the following sections, the main features of the proposed ALNS algorithm are provided in detail.

3.1. Initialization stage

A greedy insertion heuristic is used to obtain a feasible initial solution to the PDPTW-SL. All requests are initially stored in a list \mathcal{L} and inserted one by one in a random order in their best available positions (for more detail on greedy insertion (GI) – see Section 3.4). The feasibility with regard to the capacity of PD and SL vehicles and time windows is always maintained.

3.2. Adaptive score adjustment procedure

The selection of the removal and insertion operators is controlled by a roulette-wheel mechanism. Initially, all removal and insertion operators are equally weighted (i.e., for 10 considered removal and 10 insertion operators, *probabilities* of each are set to 0.1). As in the implementation of Røpke and Pisinger [29], during the course of the algorithm, more specifically every N_w iterations, each operator is given a score π_i (i.e., to assess its performance). The *probability* is updated as $P_d^{t+1} = P_d^t (1 - r_p) + r_p \pi_i / \omega_i$, where r_p is the roulette wheel parameter, π_i is the score of operator i and ω_i is the number of times it was used during the last N_w iterations. The score π_i of each operator measures how well the operator has performed at every iteration. If a new best solution is found, the score of the removal and insertion operators is increased by σ_1 . If the solution is better than the current solution, the score is increased by σ_2 . If the solution is worse than the current solution but accepted, the score is increased by σ_3 .

3.3. Removal stage

Ten removal operators are used in our ALNS heuristic algorithm. All operators are adapted from or inspired by Shaw [31], Røpke and Pisinger [29] and Demir et al. [8]. The removal stage mainly consists of removing ϕ requests from the current solution and adding them to so-called *removal list* \mathcal{L} . A pseudo-code of the removal procedure is presented in Algorithm 1. The algorithm is initialized with a feasible solution X as input and returns a partially destroyed solution. The parameter ϕ defines the number of iterations of the search. In Algorithm 1, a chosen operator is used to remove a set of requests (i.e., pickup, delivery and transfer nodes if used) from the solution. Removed pickup and delivery nodes are then inserted into list \mathcal{L} .

Algorithm 1: The overall structure of the removal operators

input : A feasible solution X and maximal number of iterations ϕ

output: A partially destroyed solution X_p

- 1 Initialize removal list ($\mathcal{L} \leftarrow \emptyset$)
 - 2 **for** ϕ iterations **do**
 - 3 Apply removal operator to remove a request r (includes two nodes; pickup and delivery)
 - 4 $\mathcal{L} \leftarrow \mathcal{L} \cup r$
-

The removal operators used in our implementation are introduced below.

- **Random Removal (RR)**: This operator randomly removes ϕ requests from the solution, and runs for $\phi \in [\underline{\phi}, \bar{\phi}]$ iterations, where $\underline{\phi}$ and $\bar{\phi}$ are the lower and the upper limits on the number of requests to be removed and ϕ is a random integer number within the specified range. The idea of randomly selecting nodes helps diversifying the search space. The worst-case time complexity of the RR operator is found to be $O(|\mathcal{P}|)$.
- **Route Removal (ROR)**: This operator removes a full route from the solution. It randomly selects a route from the set of routes in the solution. The remove operator then repeatedly selects a node j from this route until all nodes are removed. The corresponding node of j , i.e., its pickup or delivery node, is removed irrespective of which route it is positioned in. The ROR operator can be implemented in $O(\mathcal{P})$ worst-case time.

- **Late-Arrival Removal (LAR):** For each request r , this operator calculates the deviation of service start time from time l_r and l_{r+n} , and then removes the request with the largest deviation (i.e., $r^* = \operatorname{argmax}_{r \in \mathcal{P}} \{[\beta_r - s_r - l_r] + [\beta_{r+n} - s_{r+n} - l_{r+n}]\}$, where β_x is the departure time from node x). The idea is to prevent long waits or delayed service start times. The algorithm starts with an empty removal list, and similarly to RR runs for $\phi \in [\underline{\phi}, \bar{\phi}]$ iterations. The worst-case time complexity of the LAR operator is $O(|\mathcal{P}|^2)$.
- **Worst-Distance Removal (WDR):** This operator iteratively removes high-cost customers, where the cost is defined as the sum of distances from the preceding and following nodes on the tour of both the pickup and the delivery nodes of a request, i.e., it removes node $r^* = \operatorname{argmax}_{r \in \mathcal{P}} \{\Upsilon_{i-1,r} + \Upsilon_{r,i+1} + \Upsilon_{j-1,r+n} + \Upsilon_{r+n,j+1}\}$, where $i - 1$ and $j - 1$, are predecessors and $i + 1$ and $j + 1$ successors of the pickup and delivery nodes, respectively. The worst-case time complexity of the WDR operator is $O(|\mathcal{P}|^2)$.
- **Shaw Removal (SR):** The objective of the SR operator is to remove a set of customers that are related in a predefined way and therefore are easy to interchange. The algorithm starts by randomly selecting a request r_1 and adds it to the removal list. Let $l_{r_1,r_2} = -1$ if two nodes, one related to r_1 (i.e., r_1 or $r_1 + n$) and another to r_2 (i.e., r_2 or $r_2 + n$) are in the same route, and 1 otherwise. The operator selects the request $r^* = \operatorname{argmin}_{r_2 \in \mathcal{P}} \{\Pi_1[\Upsilon_{r_1,r_2} + \Upsilon_{r_1+n,r_2+n}] + \Pi_2[|\beta_{r_1} - \beta_{r_2}| + |\beta_{r_1+n} - \beta_{r_2+n}|] + \Pi_3 l_{r_1,r_2} + \Pi_4 |d_{r_1} - d_{r_2}|\}$, where Π_1 – Π_4 are weights that are normalized to find the best candidate from the considered solution. The operator is applied $\phi \in [\underline{\phi}, \bar{\phi}]$ times by selecting a request not yet in the removal list which is most similar to the one last added to the list. The worst-case time complexity of the SR operator is $O(|\mathcal{P}|^2)$.
- **Proximity-Based Removal (PR):** The operator removes a set of requests that are related in terms of distance. This operator is a special case of the Shaw removal operator with $\Pi_1 = 1$, and $\Pi_2 = \Pi_3 = \Pi_4 = 0$. The worst-case time complexity of the PR operator is $O(|\mathcal{P}|^2)$.
- **Demand-Based Removal (DR):** This operator is a special case of the Shaw removal with $\Pi_4 = 1$, and $\Pi_1 = \Pi_2 = \Pi_3 = 0$. The worst-case time complexity of the DR operator is $O(|\mathcal{P}|^2)$.
- **Time-Based Removal (TR):** The operator is a special case of the Shaw removal with $\Pi_2 = 1$, and $\Pi_1 = \Pi_3 = \Pi_4 = 0$. The worst-case time complexity of the TR operator is $O(|\mathcal{P}|^2)$.
- **Historical knowledge Removal (HR):** This operator keeps a record of the position cost of every request r , defined as the sum of the distances between its preceding and following nodes of its origin and destination nodes, and calculated as $c_r = \Upsilon_{i-1,r} + \Upsilon_{r,i+1} + \Upsilon_{j-1,r+n} + \Upsilon_{r+n,j+1}$ at every iteration of the algorithm. Note that $i - 1$ and $j - 1$ are the preceding nodes of the origin and, respectively, destination nodes of r and $i + 1$ and $j + 1$ are their successors in the corresponding PD-vehicle routes. At any point in the algorithm, the best position cost c_r^* of request r is updated to be the minimum of all c_x values calculated until that point. The HR operator then picks the request r^* with maximum deviation from its best position cost, i.e., $r^* = \operatorname{argmax}_{r \in \mathcal{P}} \{c_r - c_r^*\}$. Request r^* is then added to the removal list. The operator iterates $\phi \in [\underline{\phi}, \bar{\phi}]$. The worst-case time complexity of the HR operator is $O(|\mathcal{P}|^2)$.
- **Worst Removal (WR):** This operator removes $\phi \in [\underline{\phi}, \bar{\phi}]$ requests with the highest cost. The cost in this case is computed as follows: given a solution, the cost of a request r is the difference in the objective function between the current solution (with r) and the same solution without serving r . Note that the difference between WR and WDR is that WR uses the total cost (including transfer cost) of the requests as objective, whereas WDR focuses only on the

distance. In addition, WR also takes into account the length of the newly introduced arcs $(i - 1, i + 1)$ and $(j - 1, j + 1)$ when removing nodes i and j from a route.

3.4. Insertion stage

Five insertion operators are used in the proposed ALNS heuristic algorithm. The aim of these operators is to repair a partially destroyed solution by reinserting the requests from the removal list \mathcal{L} back into the existing routes, if possible. The general schematic overview of an insertion procedure is shown in Algorithm 2. It is noteworthy that in Line 5 of the insertion algorithm, we do not consider every possible insertion. In order to avoid redundant computations in $repairTransfers(X, r, T^r)$ and $feasibleSchedule(X)$ due to time windows, a preliminary time window check is applied as described in Braekers et al. [4]. Due to the fact that PD-vehicle capacity constraints can be fully verified only after applying $repairTransfers(X, r, T^r)$ procedure, capacity violations for the routes with no transferable requests can be easily detected. Note that $feasibleSchedule(X)$ and $feasibleCapacity(X, c(X), \eta_{ij})$ (see Line 11 in Algorithm 2) methods are described in Section 3.5.

Algorithm 2: The generic structure of an $insertion\ i^*$ (X_{new}^* , \mathcal{L} , $X_{current}$) procedure

input : A partially destroyed current solution X_{new}^* , a list of removed requests \mathcal{L} , and current solution $X_{current}$
output: A feasible solution obtained after the insertion procedure

```

1 for (each request  $r$  in  $\mathcal{L}$ ) do
2    $X_{new} \leftarrow NULL$ 
3    $c(X_{new}^*) \leftarrow +\infty$ 
4   for (each route  $Q$  and route  $M$ ) do
5     for (each position  $q$  within a route  $Q$  and each position  $m$  within a route  $M$ ) do
6       Insert  $r$  and  $r + n$  in positions  $q$  and  $m$ , respectively, in solution  $X_{new}^*$ 
7       if ( $Q \neq M$ ) then
8          $(X_t, t_r^*, t_{r+n}^*) \leftarrow repairTransfers(X_{new}^*, r, T^r)$ 
9       else
10         $X_t \leftarrow X_{new}^*$ 
11        if ( $feasibleCapacity(X_t, c(X_t), \eta_{ij})$  and  $feasibleSchedule(X_t)$ ) then
12          if (acceptance criteria of  $i^*$  is satisfied) then
13             $X_{new} \leftarrow X_t$ 
14    if ( $X_{new} \neq NULL$ ) then
15       $X_{new}^* \leftarrow X_{new}$ 
16    else
17      return  $X_{current}$ 
18 return  $X_{new}^*$ 

```

The feasibility with regard to the capacity, SL schedules and time windows is always maintained. Note that checking the feasibility is not trivial, since pickup and delivery nodes of the same request can be located in different PD routes, and hence, a tour must include at least one scheduled line. Such partial solutions need to be repaired by adding transfer nodes. Therefore, $repairTransfers(X, r, T^r)$ procedure is run to insert corresponding replicated transfer nodes in a greedy fashion. As proposed for the MIP formulation, each original transfer node is replicated n (number of requests) times, hence each request gets assigned a list of replicated transfer nodes. The overview of the repair mechanism is shown in Algorithm 3.

Algorithm 3: The generic structure of the *repairTransfers*(X, r, T^r) procedure

input : A partial solution X , a request r and T^r replicated transfer nodes related to r
output: A feasible solution X_t and a origin and a destination transfer nodes (t_r^* , t_{r+n}^*) of r in the current solution X_t

- 1 $Q \leftarrow$ the route related to r
- 2 $M \leftarrow$ the route related to $r + n$
- 3 $t_r^* \leftarrow$ transfer node $\in T^r$ with the cheapest insertion in route Q , sequenced after r
- 4 $T^r = T^r \setminus t_r^*$
- 5 $t_{r+n}^* \leftarrow$ transfer node $\in T^r$ with the cheapest insertion in route M , sequenced earlier than $r + n$
- 6 $X_t \leftarrow$ updated X given minimal cost insertions of t_r^* and t_{r+n}^*
- 7 return (X_t, t_r^*, t_{r+n}^*)

A transferable request r is assigned two related replicated transfer nodes: t_r^* (i.e., origin transfer node), and t_{r+n}^* (i.e., destination transfer node).

We now briefly define the five insertion operators used in the ALNS algorithm. We note that the requests are randomly chosen from the removal list \mathcal{L} .

- **Greedy Insertion (GI)**: This operator repeatedly inserts a request (both pickup and delivery nodes) in the best possible position of the routes. Δ_{I,J_j}^r is the objective function value when the pickup node of r is inserted in route I (position i) and its corresponding delivery node is inserted in route J (position j). Thus, $\Delta_{I,J_j}^{r*} = \operatorname{argmin}\{\Delta_{I,J_j}^r\}$. The worst-case time complexity of this operator is $O(|\mathcal{P}|^2)$.
- **Second-best Insertion (SI)**: This new operator chooses the second best insertion for randomly selected unassigned request. The main idea of using this operator is to diversify the search. The worst-case time complexity of the SI is $O(|\mathcal{P}|^2)$.
- **Greedy Insertion with Noise function (GIN)**: This operator is an extension of the greedy algorithm but uses a degree of freedom in selecting the best place for a node. This degree of freedom is achieved by modifying the cost for request r : $\text{New Cost} = \text{Actual Cost} + \bar{d}\mu\epsilon$, where \bar{d} is the maximum distance between nodes, μ is a noise parameter used for diversification and is set equal to 0.1, and ϵ is a random number between $[-1, 1]$. New Cost is calculated for each request in \mathcal{L} . The worst-case time complexity of the GIN operator is $O(|\mathcal{P}|^2)$.
- **Second-best Insertion with Noise function (SIN)**: This operator is an extension of the SI and uses the same noise function as the GIN operator. The worst-case time complexity of the SIN operator is $O(|\mathcal{P}|^2)$.
- **Best out of λ Feasible Insertions (λ FI)**: This new operator is similar to GI but chooses the best insertion out of the first λ feasible insertions. The parameter λ is a randomly generated integer number between 1 and ψ . For each unassigned request we generate a set of possible insertions associated with corresponding distance-based costs (disregarding distance to the corresponding transfer nodes), while considering precedence constraints. These positions are not necessary feasible in terms of time windows. Additionally, in order to filter out some more of the infeasible insertions, the time windows of the subsequent nodes visited within the considered routes are verified. The generated set is sorted in increasing order based on insertion costs. Hence the operator investigates the cheapest insertions overall routes first. The worst-case time complexity of the λ FI operator is $O(|\mathcal{P}|^2)$.

Moreover, additional five variants of these operators are also used, where the requests are sorted in \mathcal{L} in terms of their time flexibility (i.e., the least flexible first). The flexibility of request r can be computed as $|u_{r+n} - l_r|$.

3.5. Feasibility of the routes and schedules

Scheduling (time windows) constraints bring complexity to the PDPTW-SL since the requests may be picked up by a PD vehicle and delivered by another one. This implies that PD vehicles and SLs must be synchronized. This problem can be handled as shown in Algorithm 4.

Algorithm 4: The generic structure of the *feasibleSchedule(X)* procedure

```

input : A solution  $X$ 
output: Boolean value: TRUE if feasible, FALSE if infeasible

1  $list \leftarrow \emptyset$ 
2  $\beta_{t_r^*} \leftarrow 0, \beta_{t_{r+n}^*} \leftarrow 0, \forall r \in \mathcal{P}$ 
3 for (each route  $I$  in  $X$ ) do
4   for (each node  $i$  in  $I$ ) do
5      $\beta_i \leftarrow \max\{\beta_{p_i} + \Upsilon_{p_i,i} + s_i; l_i + s_i\}$ 
6     if ( $\beta_i > u_i + s_i$ ) then
7        $\text{return } FALSE$ 
8     if ( $i = t_{r+n}^*, \forall r \in \mathcal{P}$ ) then
9        $list \leftarrow list \cup i$ 
10       $i \leftarrow \text{end route}$ 

11 while ( $list \neq \emptyset$ ) do
12   for ( $t_{r+n}^*$  in  $list$ ) do
13     if ( $\beta_{t_r^*} > 0$ ) then
14        $path_r \leftarrow \text{generatePath}(t_r^*, t_{r+n}^*)$ 
15        $\beta_{t_{r+n}^*} \leftarrow \text{determineTime}(\beta_{t_r^*}, path_r)$ 
16       for (each successor node  $i$  of  $t_{r+n}^*$ ) do
17          $\beta_i \leftarrow \max\{\beta_{p_i} + \Upsilon_{p_i,i} + s_i; l_i + s_i\}$ 
18         if ( $\beta_i > u_i + s_i$ ) then
19            $\text{return } FALSE$ 
20         if ( $i = t_{r_1+n}^*, \forall r_1 \in \mathcal{P}$ ) then
21            $list \leftarrow list \cup i$ 
22            $i \leftarrow \text{end route}$ 
23        $list \leftarrow list \setminus t_{r+n}^*$ 

24   if ( $\beta_{t_r^*} = 0, \forall \beta_{t_{r+n}^*} \in list$ ) then
25      $\text{return } FALSE$ 

26  $\text{return } TRUE$ 

```

All β values for the transfer nodes are reset to 0 in Line 2. The first block of the algorithm (Lines 3 – 10) sets the time for all nodes at the earliest possible value (depending on the start of the time window and the departure time from the preceding node p_i). The algorithm switches to the next route whenever a destination transfer node is reached. This node is added to a *list* that contains all destination transfer nodes which do not have an updated synchronized time. The second part of

the algorithm (Lines 11 – 25) runs until the *list* of destination transfer nodes is emptied or a cycle (see Figure 4) is found.

- For each $t_{r+n}^* \in list$ with the corresponding origin transfer node (i.e., t_r^*) that has already received a timing, generate a shortest path (Dijkstra [13]) from t_r^* to t_{r+n}^* (Line 14). Given the shortest path and the departure time from t_r^* , the algorithm computes the time at t_{r+n}^* by considering earliest scheduled departure time later than $\beta_{t_r^*}$ (i.e., $\lceil \beta_{t_r^*} \rceil$) for every intermediate line used (Line 15). Considered t_{r+n}^* with a synchronized timing is removed from the *list*. Finally, all succeeding nodes of t_{r+n}^* get assigned a value until a destination transfer node is reached and added to the *list*.
- Whenever all $\beta_{t_r^*}$ is zero $\forall t_{r+n}^* \in list$, the algorithm finds a cycle (see Figure 4) and stops. Hence, no feasible schedule is possible for a given solution X .

For a better understanding, we refer to the example shown in Figure 3. Three requests, namely a , b , and c , are to be delivered to destination nodes $a+n$, $b+n$ and $c+n$, respectively. All the requests are transferable, thus each request is shipped on a scheduled line. All these transfer nodes are replications of the original transfer nodes and each replication is assigned to only one request. In the present example, T_a represents the origin transfer node of a , and T_{a+n} is its destination transfer node. For the sake of simplicity, each arc has one time unit and each node does not require any service time. The numbers on top of the nodes indicate the departure time from that specific node. In this example, three PD vehicles are needed for the transportation of these three requests.

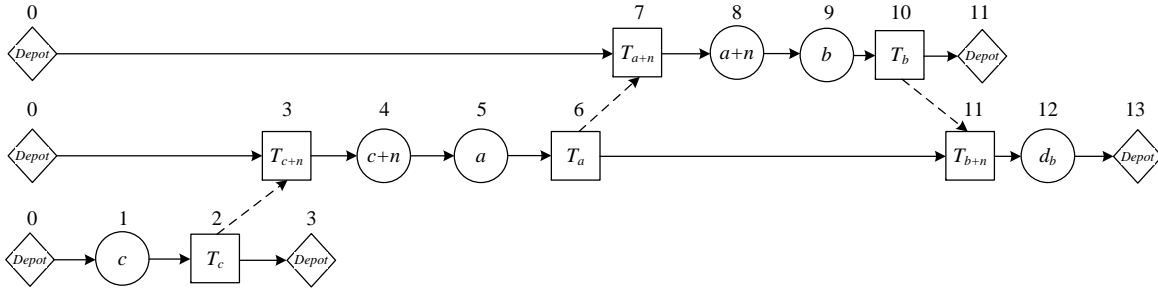


Figure 3 An interdependency relation of the routes

By referring to Figure 3, the feasibility of the schedule is checked as follows: the process starts in the first route. The algorithm reaches the transfer node T_{a+n} that is a destination transfer node of request a . The algorithm moves to the second route and reaches T_{c+n} , which is a destination transfer node of request c . Finally, it moves to the last route and updates the timing for the whole route as no destination transfer node is encountered. The *list* contains two destination transfer nodes, i.e., T_{a+n} and T_{c+n} . A shortest path is generated from T_c to T_{c+n} , and the departure time $\beta_{T_{c+n}}$ is updated (i.e., 3). Afterwards, the subsequent nodes of T_{c+n} for a given synchronized time $\beta_{T_{c+n}}$ get assigned time values until T_{b+n} is reached. Similarly, the timing of T_{a+n} is updated and followed by T_{b+n} .

Note that a cycle implies that the precedence constraints are violated for at least two transferable requests, since such requests need to be picked up and dropped off twice (see Figure 4). In addition, cycles may be composed of multiple routes and requests.

The procedure $determineTime(\beta_{t_r^*}, path_r)$ (see Algorithm 4, Line 15) computes departure time from

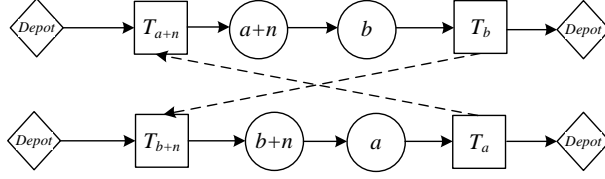


Figure 4 An example of a two-request cycle

t_{r+n}^* for a given departure time from t_r^* .

The capacity constraints of both PD and SL vehicles are verified in Algorithm 5. First, the algorithm checks SL capacity by first generating scheduled line paths along with every scheduled departure time from the considered transfer nodes (Line 3). The capacity variables are updated for each SL and each possible scheduled departure. The cost of the solution is updated in Line 6. Whenever the capacity is violated (Line 7), the algorithm stops and returns *FALSE* value. The PD-vehicle capacity constraints can be checked in a standard way with some preprocessing (Line 9). In particular, the destination transfer node of a transferable request r (i.e., t_{r+n}^*) becomes an origin node on a different route, hence it will have a positive demand (i.e., d_r) whereas the origin transfer node of r (i.e., t_r^*) becomes a destination node and it is assigned a negative demand (i.e., $-d_r$).

Algorithm 5: The generic structure of *feasibleCapacity*($X, c(X), \eta_{ij}$) procedure

input : A solution X , routing cost of X , and cost of using SL η_{ij} per unit shipped on (i, j)

output: A boolean value θ

- 1 Initialize m_{ij}^w array for the capacity used on the scheduled line (i, j) at time p_{ij}^w and $\overline{\mathcal{P}}^t \leftarrow$ set of transferable requests in solution X
 - 2 **for** (r in $\overline{\mathcal{P}}^t$) **do**
 - 3 set $(\overline{\mathcal{K}}^r, \gamma) \leftarrow$ scheduled lines (i, j) used by r and corresponding scheduled departure times γ_{ij}^r
 - 4 **for** $((i, j) \in \overline{\mathcal{K}}^r)$ **do**
 - 5 set $m_{ij}^w \leftarrow m_{ij}^w + d_r$
 - 6 set $c(X) \leftarrow c(X) + d_r \eta_{ij}$
 - 7 **if** $(m_{ij}^w > k_{ij})$ **then**
 - 8 return *FALSE*
 - 9 **if** (*PDvehicleCapacityViolated*(X)) **then**
 - 10 return *FALSE*
 - 11 return *TRUE*
-

3.6. Acceptance and stopping criteria

In the ALNS, we have implemented simulated annealing as a master search framework for the PDPTW-SL. The overall framework of the ALNS algorithm with simulated annealing is provided in Algorithm 6.

In the algorithm, X_{best} indicates the best solution found during the search, $X_{current}$ is the current solution obtained at the beginning of an iteration, and X_{new} is a temporary solution found at the end of iteration that can be discarded or become the current solution. The cost of solution X is denoted

Algorithm 6: The general framework of the ALNS with simulated annealing

input : A set of removal operators D , a set of insertion operators I , initialization constant P_{init} , cooling rate κ
output: X_{best}

- 1 Generate an initial solution by using the Greedy insertion algorithm
- 2 Initialize probability P_d^t for each destroy operator $d \in D$ and probability P_i^t for each insertion operator $i \in I$
- 3 Let T be the temperature and j be the counter initialized as $j \leftarrow 1$
- 4 Let $X_{current} \leftarrow X_{best} \leftarrow X_{init}$
- 5 **repeat**
- 6 Select a removal operator $d^* \in D$ with probability P_d^t
- 7 Let X_{new}^* be the solution obtained by applying operator d^* to $X_{current}$
- 8 Select an insertion operator $i^* \in I$ with probability P_i^t
- 9 Let X_{new} be the new solution obtained by applying operator i^* to X_{new}^*
- 10 **if** $c(X_{new}) < c(X_{current})$ **then**
- 11 $X_{current} \leftarrow X_{new}$
- 12 **if** $c(X_{current}) < c(X_{best})$ **then**
- 13 $X_{best} \leftarrow X_{current}$
- 14 **else**
- 15 Let $v \leftarrow e^{-(c(X_{new})-c(X_{current}))/T}$
- 16 Generate a random number $\epsilon \in [0, 1]$
- 17 **if** $\epsilon < v$ **then**
- 18 $X_{current} \leftarrow X_{new}$
- 19 $T \leftarrow \kappa T$
- 20 Update probabilities using the adaptive weight adjustment procedure
- 21 $j \leftarrow j + 1$
- 22 **until** the maximum number of iterations is reached

by $c(X)$. A solution X_{new} is always accepted if $c(X_{new}) < c(X_{current})$, and accepted with probability $e^{-(c(X_{new})-c(X_{current}))/T}$ if $c(X_{new}) > c(X_{current})$, where T is the *temperature*. The initial temperature is set at P_{init} , where P_{init} is an initialization constant. The current temperature is gradually decreased during the course of the algorithm as κT , where $0 < \kappa < 1$ is a fixed parameter. The algorithm returns the best found solution after a fixed number of iterations (i.e., 10,000 iterations).

4. Computational results

This section presents the results of extensive computational experiments performed to assess the performance of our ALNS heuristic algorithm. We first describe the generation of the instances and of the parameters. We then present the computational results obtained by the proposed heuristic algorithm.

4.1. Data and experimental setting

Three sets of instances, namely R , C , and RC , with three scheduled lines in a triangular topology and a frequency of one departure of every 30 time units are considered. Each instance contains 100 requests (i.e., 100 pick-up and 100 delivery nodes) over 200×200 time units on an Euclidean space. Instances follow a naming convention of G_n_sl , where G is the geographic distribution of the customers, n is the number of requests that needs to be served, and sl is the number of SLs.

Instances are classified with respect to the geographical locations of the nodes. For example, *C* involves clustered nodes around transfer nodes, *R* considers uniform-randomly distributed request nodes, and finally *RC* involve randomly clustered nodes. More specifically, *C* and *RC* have the nodes positioned within at most 30, respectively 80 time units to one of its three available transfer nodes. In all cases, two depots with 30 heterogeneous PD vehicles each are considered.

The planning horizon is set to 10 working hours (i.e., 600 time units). The widths of the time windows are randomly generated between 26 and 91 time units. Service times are considered to be up to three time units. Each demand is assigned between one and three units. The capacity of PD vehicle is generated between six and 20 units. The carrying capacity on the considered SLs is assumed to be 15 demand units. Additional instances were also generated such that subsets of requests, PD vehicles, SLs out of *R*, *C*, and *RC* datasets are used. The three main datasets (i.e., *R*, *C*, and *RC*) can be found on the web page of SmartLogisticsLab [32].

4.2. Parameter tuning

The proposed algorithm is implemented in NetBeans IDE 7.1.1 using Java. All experiments were conducted on an Intel Core i5 with 2.6 GHz speed and 4 GB RAM. A more detailed tuning was done on the five most sensitive parameters as shown in Table 4. These were identified during some preliminary tests. Three instances of small sizes (i.e., 8, 16 and 25 requests) were solved 10 times each for the given combination of parameters. The rest of the parameters were chosen based on some other experiments and our intuition.

Table 4
Results of parameter tuning

Iterations	Destroy rate %	σ_1	σ_2	σ_3	Average value €	Iterations	Destroy rate %	σ_1	σ_2	σ_3	Average value €
10,000	0.15	1	0	1	954.09	15,000	0.15	1	0	1	953.72
10,000	0.15	1	0	5	953.08	15,000	0.15	1	0	5	953.61
10,000	0.15	1	3	1	954.18	15,000	0.15	1	3	1	950.98
10,000	0.15	1	3	5	951.86	15,000	0.15	1	3	5	954.02
10,000	0.15	5	0	1	958.49	15,000	0.15	5	0	1	954.03
10,000	0.15	5	0	5	958.52	15,000	0.15	5	0	5	952.59
10,000	0.15	5	3	1	954.66	15,000	0.15	5	3	1	952.94
10,000	0.15	5	3	5	953.00	15,000	0.15	5	3	5	952.57
10,000	0.25	1	0	1	950.04	15,000	0.25	1	0	1	949.80
10,000	0.25	1	0	5	953.80	15,000	0.25	1	0	5	953.82
10,000	0.25	1	3	1	954.87	15,000	0.25	1	3	1	951.89
10,000	0.25	1	3	5	948.25	15,000	0.25	1	3	5	950.64
10,000	0.25	5	0	1	953.16	15,000	0.25	5	0	1	951.42
10,000	0.25	5	0	5	954.14	15,000	0.25	5	0	5	954.47
10,000	0.25	5	3	1	954.10	15,000	0.25	5	3	1	952.34
10,000	0.25	5	3	5	951.46	15,000	0.25	5	3	5	953.02
10,000	0.35	1	0	1	950.63	15,000	0.35	1	0	1	949.67
10,000	0.35	1	0	5	951.08	15,000	0.35	1	0	5	950.36
10,000	0.35	1	3	1	950.53	15,000	0.35	1	3	1	954.97
10,000	0.35	1	3	5	951.65	15,000	0.35	1	3	5	954.01
10,000	0.35	5	0	1	951.77	15,000	0.35	5	0	1	952.77
10,000	0.35	5	0	5	949.89	15,000	0.35	5	0	5	950.59
10,000	0.35	5	3	1	949.72	15,000	0.35	5	3	1	951.11
10,000	0.35	5	3	5	951.88	15,000	0.35	5	3	5	952.34

In total, our algorithm contains 16 user-controlled parameters which are listed in Table 5.

Table 5
Parameters used in the ALNS heuristic

Category	Description	Typical values
A	Total number of iterations (N_i)	10,000
	Number of iterations for roulette wheel (N_w)	200
	Roulette wheel parameter (r_p)	0.1
	New global solution (σ_1)	1
	Better solution (σ_2)	3
	Worse solution (σ_3)	5
B	Startup temperature parameter (P_{init})	200
	Cooling rate (κ)	0.9995
	Lower limit of removable requests ($\underline{\phi}$)	2.5% of $ \mathcal{P} $
	Upper limit of removable requests ($\bar{\phi}$)	25% of $ \mathcal{P} $
	First Shaw parameter (Π_1)	0.5
	Second Shaw parameter (Π_2)	0.2
	Third Shaw parameter (Π_3)	0.1
	Fourth Shaw parameter (Π_4)	0.2
	Noise parameter (μ)	0.1
	Number of feasible insertions (ψ)	30

It is noted that a driving cost of the PD vehicles is assumed to be 0.5 € per minute. It seems reasonable considering all operational costs, such as fuel consumption, driver wage, insurance, and tax. The cost of each demand unit of package request shipped on a fixed line is set to 1 €, which includes handling, storage and transportation costs. The parameters used in the ALNS algorithm are categorized into two categories as described below.

- Group A defines the selection procedure with the roulette wheel mechanism. We note that our setting of the parameters σ_1 , σ_2 and σ_3 is contrary to the expected setting $\sigma_1 \geq \sigma_2 \geq \sigma_3$, normally used to reward an operator for good performance. In our implementation and similar to Demir et al. [8, 10], we have chosen (based on parameter tuning) an unconventional setting of these parameters whereby the discovery of a worse solution is rewarded more than the discovery of a better solution. This is to help diversify the search in the algorithm.
- Group B of parameters is used to calibrate the simulated annealing acceptance mechanism and the removal and insertion operators.

To show the number of times each operator called within the ALNS, we give relevant information in Table 6 and Table 7. These tables show, for each operator, the frequency of use during the course of the algorithm. The total time spent to run each operator is also shown in the parentheses. We note that the results are obtained using only one instance of each different size in terms of the number of requests.

The results shown in Table 6 indicate that operators RR, SR, PR and HR are used almost equally often. In many cases, the most widely used operators are LAR, TR and WR.

As seen in Table 7, most used insertion operators are the ones that have ordered \mathcal{L} . Since least flexible requests are inserted first, these operators have higher chances to insert all requests within the existing routes. More specifically, oGI, oSI and their variants with noise functions are widely

Table 6

Number of iterations and the CPU times required by the removal operators

Instance	RR	ROR	LAR	WDR	SR	PR	DR	TR	HR	WR
R_25_1	986(0.0)	800(0.0)	1,358(0.1)	766(0.0)	979(0.0)	938(0.0)	900(0.0)	1,121(0.0)	904(0.0)	1,248(0.0)
R_50_1	946(0.0)	947(0.0)	900(0.0)	951(0.0)	1,051(0.0)	963(0.0)	826(0.0)	1,241(0.0)	946(0.0)	1,229(0.0)
R_75_1	995(0.0)	893(0.0)	1,104(0.0)	943(0.1)	958(0.1)	933(0.1)	769(0.0)	1,196(0.1)	936(0.1)	1,273(0.0)
R_100_1	1,024(0.0)	640(0.0)	1,323(0.1)	1,294(0.1)	907(0.1)	926(0.1)	743(0.1)	1,205(0.1)	1,096(0.1)	842(0.1)

Table 7

Number of iterations and the CPU times required by the insertion operators

Instance	GI	SI	GIN	SIN	λ FI	oGI	oSI	oGIN	oSIN	o λ FI
R_25_1	860(0.2)	855(0.2)	537(0.1)	510(0.2)	451(0.0)	1,843(0.9)	1,945(1.2)	1,024(0.6)	1,045(0.6)	930(0.3)
R_50_1	655(0.7)	631(0.5)	312(0.1)	414(0.4)	453(0.1)	2,359(11.2)	2,332(10.8)	971(4.5)	941(4.5)	932(1.1)
R_75_1	636(1.4)	532(0.9)	404(0.6)	263(0.5)	286(0.0)	2,697(44.9)	2,643(42.5)	1,040(13.5)	832(13.0)	667(1.5)
R_100_1	531(4.2)	485(2.1)	212(1.3)	252(1.1)	273(0.2)	3,315(285.0)	3,095(263.8)	875(56.7)	510(42.5)	452(4.7)

selected. It is noted that randomly-ordered insertion operators do not perform well, as many times the algorithm cannot insert all unassigned requests back to the solution due to their order of insertion.

Tables 8 – 9 indicate the number of times an operator has found the best and a better solution compared to the current one, respectively. It is noted that the number in parenthesis indicates the number of times a current solution is improved, but not to become a best known.

Table 8

Number of global best solutions found and number of improving solutions achieved by the removal operators

Instance	RR	ROR	LAR	WDR	SR	PR	DR	TR	HR	WR
R_25_1	1(61)	0(47)	1(29)	0(4)	0(77)	0(43)	2(48)	4(97)	1(60)	2(152)
R_50_1	24(179)	2(111)	6(205)	3(68)	4(188)	5(137)	1(72)	5(257)	1(56)	1(154)
R_75_1	14(195)	6(94)	3(208)	1(154)	7(235)	6(178)	4(42)	6(290)	9(189)	4(168)
R_100_1	17(226)	9(73)	9(292)	4(107)	7(228)	7(196)	2(32)	4(306)	8(163)	7(256)

Table 9

Number of global best solutions found and number of improving solutions achieved by the insertion operators

Instance	GI	SI	GIN	SIN	λ FI	oGI	oSI	oGIN	oSIN	o λ FI
R_25_1	1(7)	1(7)	0(3)	0(2)	0(0)	3(253)	4(210)	2(73)	0(63)	0(0)
R_50_1	18(36)	0(10)	0(3)	0(1)	0(0)	19(755)	15(575)	0(25)	0(22)	0(0)
R_75_1	8(21)	0(5)	0(0)	0(2)	0(0)	28(1069)	23(628)	1(15)	0(13)	0(0)
R_100_1	12(27)	0(6)	0(1)	0(0)	0(0)	32(1211)	30(630)	0(2)	0(2)	0(0)

The results indicate that all removal operators, to some extent, contribute to achieving improved solutions. On the other hand, some insertion operators (i.e., λ FI and o λ FI) do not improve a current solution at any time. However, as it will be shown below, these operators are needed to diversify the search and achieve better overall performance of the algorithm. Furthermore, as expected, greedy operators (i.e., best and second-best) help obtaining improved solutions.

In order to identify the usefulness of the new insertion operators proposed in this article, we tested four configurations of the ALNS. These are shown in Table 10, along with the average objective function values over ten runs of the algorithm. We used the same instances as in Table 4.

According to the obtained results, using SI, SIN and λ FI operators leads to the best performance of the algorithm. These operators are mainly destined to diversify the search. As seen in Table 10,

Table 10
A tuning of insertion operators

Configuration	Average value €
Without SI and SIN	949.61
Without λ FI and α λ FI	952.15
Without SI, SIN, λ FI and α λ FI	956.13
With all operators	948.25

it seems imperative to use such operators along with the unconventional scoring setting.

4.3. Results of the ALNS heuristic on the PDPTWs

In this section, we provide in Tables 11 – 13 computational results on the PDPTW benchmark instances (i.e., Li and Lim [19]), which come in three sets: R , C and RC classified with respect to the positioning of the customers (i.e., random, clustered and randomly-clustered). The reason for choosing these instances is that Røpke and Cordeau [28] and Baldacci et al. [2] provided their results by using the minimization of operating costs as our algorithm does. Tables 11 – 13 compare published results to the ones obtained by our ALNS heuristic algorithm. The comparison is made in terms of the average solution values obtained through 10 runs of the algorithm. This table presents, for each instance, the value of the best known or optimal solution compiled from [28, 2, 29] under column “Best known value”. Note that the values in bold emphasize that the proposed algorithm found the best known solution. The symbol “*” indicates that the values are not necessary optimal and are obtained by Røpke and Pisinger [29]. Moreover, we note that all figures presented in these tables use a single decimal digit. For the ALNS algorithm, we then present the best solution value obtained in column “ALNS best found”. In addition, we indicate “ALNS average value” after 10 runs with the corresponding average GAP (%) (i.e., let $\nu(ALNS)$ be the solution value produced by our algorithm, then, the $GAP\ (%) = 100(\nu(ALNS) - \nu(\text{Best})) / \nu(ALNS)$, where $\nu(\text{Best})$ is the best known solution value for each instance). Finally, we show the corresponding average CPU times required to run the algorithm.

Table 11
Results of the ALNS heuristic on benchmark PDPTW-C instances

Li and Lim [19] instances	# of requests	Best known value	CPU seconds	ALNS best found	ALNS average value	GAP %	CPU seconds
LC1.2.1	106	2,704.6	3.3	2,704.6	2,704.6	0.00	47
LC1.2.2	105	2,764.6	21.5	2,764.6	2,764.8	0.01	51
LC1.2.3	103	2,772.2	114.9	2,772.2	2,779.3	0.26	121
LC1.2.4	105	2,661.4*	209	2,661.4	2,684	0.84	123
LC1.2.5	107	2,702	4.8	2,702	2,702	0.00	40
LC1.2.6	107	2,701	7.4	2,701	2,701	0.00	42
LC1.2.7	107	2,701	7.7	2,701	2,701	0.00	38
LC1.2.8	105	2,689.8	16	2,689.8	2,689.9	0.00	43
LC1.2.9	105	2,724.2	55.3	2,724.2	2,758.1	1.23	63
LC1.2.10	104	2,741.6	137.1	2,743.9	2,763.8	0.80	67
Average			58			0.31	66

Table 12
Results of the ALNS heuristic on benchmark PDPTW-RC instances

Li and Lim [19] instances	# of requests	Best known value	CPU seconds	ALNS best found	ALNS average value	GAP %	CPU seconds
LRC1.2.1	106	3,606.1	3.1	3,606.1	3,608.9	0.08	45
LRC1.2.2	103	3,292.4	322.3	3,292.4	3,304.3	0.36	67
LRC1.2.3	105	3,079.5*	183	3,108.5	3,121.2	1.34	107
LRC1.2.4	106	2,525.8*	284	2,552.1	2,583.2	2.22	190
LRC1.2.5	107	3,715.8	42.1	3,766.2	3,788.3	1.91	55
LRC1.2.6	105	3,360.9	7	3,382.4	3,401	1.18	39
LRC1.2.7	106	3,317.7	408.2	3,344.3	3,377.1	1.76	52
LRC1.2.8	104	3,086.5	1562.7	3,129.5	3,143.7	1.82	63
LRC1.2.9	104	3,053.8	1757.2	3,093.6	3,141.5	2.79	54
LRC1.2.10	105	2,837.5*	156	2,857.2	2,881.3	1.52	51
Average			473			1.49	72

Table 13
Results of the ALNS heuristic on benchmark PDPTW-R instances

Li and Lim [19] instances	# of requests	Best known value	CPU seconds	ALNS best found	ALNS average value	GAP %	CPU seconds
LR1.2.1	105	4,819.1	1.6	4,819.1	4,819.1	0.00	42
LR1.2.2	105	4,093.1	20.6	4,093.1	4,101.4	0.20	96
LR1.2.3	104	3,486.8	3690.8	3,486.8	3,503.6	0.48	162
LR1.2.4	105	2,830.7*	228	2,839.1	2,873.1	1.48	201
LR1.2.5	106	4,221.6	2.6	4,221.6	4,239.2	0.42	42
LR1.2.6	107	3,763	180.9	3,763	3,769.9	0.18	70
LR1.2.7	103	3,112.9*	173	3,112.9	3,124.9	0.38	107
LR1.2.8	103	2,645.4*	226	2,652.4	2,664.7	0.72	159
LR1.2.9	105	3,953.5	15.4	3,953.5	3,961	0.19	52
LR1.2.10	104	3,386.3	1376.7	3,390.4	3,411.2	0.73	62
Average			592			0.48	99

As shown in Table 11 – 13, the ALNS heuristic performs very well on the PDPTW instances considered in our tests. For the majority of instances, our heuristic algorithm was able to obtain the best known solutions published in the literature in at least one out of the 10 runs. For the rest of the instances, the percentage deviations are found to be not greater than 2.79%. The average CPU time required for the algorithm on the instances is found to be around 78 seconds.

4.4. Results of the generated instances

This section presents the results obtained by the proposed heuristic on the four generated sets of PDPTW-SL instances. These sets are generated from the three main datasets described in Section 4.1, by considering subsets of request, transfer node and PD vehicle sets. For the first group (with up to 12 requests and one SL), each instance was solved 10 times with the proposed heuristic and once with the PDPTW-SL MIP model by using CPLEX 12.3 (IBM ILOG [17]) with its default settings and the valid inequalities proposed by Ghilas et al. [14]. A common time-limit of ten hours was imposed to CPLEX on the solution time for all instances. The following three groups were solved 10 times using the proposed ALNS in the context of PDPTW-SL and PDPTW. The detailed results of these experiments are presented in Table 14 – 15.

In most of the cases, Table 14 indicates that the ALNS algorithm generated the same solution values as those of CPLEX, but in a substantially smaller amount of time. For the instances solved to optimality, the average CPU time required by CPLEX is approximately 8,251 seconds where the

Table 14
Computational results for the instances with up to 12 requests

Instance	CPLEX				ALNS		
	Upper bound	Lower bound	GAP %	CPU seconds	Value	GAP %	CPU seconds
C_6.1	369.36	369.36	0.00	5,267	369.36	0.00	1
C_7.1	390.30	390.30	0.00	3,762	390.30	0.00	1
C_8.1	446.34	384.93	13.76	36,000	446.34	-	2
C_9.1	493.66	327.43	33.67	36,000	472.68	-	2
C_10.1	-	335.89	-	36,000	510.01	-	2
C_11.1	-	347.68	-	36,000	522.84	-	2
C_12.1	-	366.81	-	36,000	541.60	-	2
RC_6.1	572.75	572.75	0.00	185	572.75	0.00	1
RC_7.1	575.95	575.95	0.00	368	575.95	0.00	1
RC_8.1	585.32	585.32	0.00	961	585.32	0.00	1
RC_9.1	593.69	593.69	0.00	32,204	593.69	0.00	2
RC_10.1	599.94	599.94	0.00	24,925	599.94	0.00	2
RC_11.1	624.47	624.47	0.00	18,128	624.47	0.00	2
RC_12.1	-	608.89	-	36,000	662.03	-	2
R_6.1	416.16	416.16	0.00	2	416.16	0.00	1
R_7.1	473.05	473.05	0.00	5	473.05	0.00	1
R_8.1	558.17	558.17	0.00	16	558.17	0.00	1
R_9.1	632.41	632.41	0.00	8,782	632.41	0.00	1
R_10.1	636.05	636.05	0.00	3,421	636.05	0.00	2
R_11.1	748.28	748.28	0.00	17,493	748.28	0.00	2
R_12.1	-	771.01	-	36,000	934.73	-	2

same statistic for the ALNS to produce the reported solutions is approximately 2 seconds. In some cases where CPLEX could not find any solution or could obtain a sub-optimal one, the proposed ALNS was able to find solutions that have a tighter GAP relative to the best lower bound found within the imposed time limit.

Tables 15 – 17 provide the results obtained for larger instances. The column *Instance* indicates the instance identification. The *Best known cost* column indicates the best objective value found after 10 runs of the algorithm. In addition, columns *Average cost* and *Average GAP* show the average objective values over 10 runs and respectively the average GAP from the best solution found. The *Cost savings* column indicates the cost savings of the best PDPTW-SL solution compared to the best corresponding PDPTW solution. The *Best driving time* column shows the total driving time of the best solution found and the *Driving time savings* indicate the savings with regard to the total driving time. *CPU* indicates the average computational time for solving the instances. *Vehicles used* and *Units on SLs* provide the number of PD vehicles used and the number of shipments (demand units) on the available SLs in the best solution found.

The proposed algorithm is relatively fast. For example, instances of up to 100 requests are solved in less than 40 minutes. As it can be noticed from the Tables 15 – 17, the ALNS algorithm for the PDPTW-SL is substantially slower than the same algorithm in case of the PDPTW. The main reason is the extra complexity that is induced by having the flexibility of using available scheduled lines, thus making multiple PD-vehicle routes depend on each other. In particular synchronization constraints (i.e., time windows and capacity) require extra computation time (e.g., 20 – 30% of the CPU time).

It is noted that the efficiency of the proposed system may be highly dependent on both the spatial pattern of the requests and the configuration of the scheduled lines. Unless the design of the scheduled line services (routes and schedules) is integrated with vehicle routing, it is likely that the

Table 15
Results of *C* instances

Instance	Best known value	Average value	Average GAP %	Cost savings %	Best driving time	Driving time savings %	CPU seconds	# of vehicles used	Units on SLs
C_25_0	997.83	997.83	0.00		1,995.66		1	5	
C_25_1	943.92	944.25	0.04	5.40	1,873.84	6.10	10	7	14
C_25_2	927.40	935.25	0.84	7.06	1,828.79	8.36	11	8	26
C_25_3	818.46	824.20	0.70	17.98	1,586.93	20.48	17	8	50
C_50_0	1,529.37	1,541.56	0.79		3,058.73		4	8	
C_50_1	1,415.60	1,425.67	0.71	7.44	2,793.19	8.68	124	10	14
C_50_2	1,342.23	1,356.01	1.02	12.24	2,634.47	13.87	121	10	50
C_50_3	1,214.16	1,229.30	1.23	20.61	2,354.31	23.03	144	10	74
C_75_0	2,040.54	2,069.10	1.38		4,081.09		12	10	
C_75_1	1,807.33	1,829.14	1.19	11.43	3,558.66	12.80	411	12	56
C_75_2	1,686.85	1,702.30	0.91	17.33	3,283.70	19.54	528	12	90
C_75_3	1,621.18	1,641.60	1.24	20.55	3,112.36	23.74	650	13	130
C_100_0	2,349.46	2,378.79	1.23		4,698.91		34	12	
C_100_1	2,112.73	2,126.42	0.64	10.08	4,167.47	11.31	2378	13	58
C_100_2	2,040.36	2,069.71	1.42	13.16	3,964.72	15.62	2224	14	116
C_100_3	1,915.40	1,939.87	1.26	18.47	3,662.80	22.05	2357	14	168

Table 16
Results of *RC* instances

Instance	Best known value	Average value	Average GAP %	Cost savings %	Best driving time	Driving time savings %	CPU seconds	# of vehicles used	Units on SLs
RC_25_0	1,633.56	1,633.56	0.00		3,267.11		2	8	
RC_25_1	1,393.59	1,394.94	0.10	14.69	2,779.19	14.93	5	7	8
RC_25_2	1,338.54	1,338.54	0.00	18.06	2,665.08	18.43	5	7	12
RC_25_3	1,328.42	1,328.42	0.00	18.68	2,642.84	19.11	5	7	14
RC_50_0	2,445.38	2,456.61	0.46		4,890.76		3	11	
RC_50_1	2,348.14	2,365.36	0.73	3.98	4,684.27	4.22	28	11	4
RC_50_2	2,348.14	2,366.23	0.76	3.98	4,684.27	4.22	33	11	12
RC_50_3	2,337.20	2,356.19	0.81	4.42	4,656.39	4.79	36	10	18
RC_75_0	2,863.05	2,868.34	0.18		5,726.11		9	12	
RC_75_1	2,814.39	2,825.63	0.40	1.70	5,616.78	1.91	143	12	12
RC_75_2	2,814.39	2,836.39	0.78	1.70	5,616.78	1.91	155	12	12
RC_75_3	2,814.39	2,839.05	0.87	1.70	5,616.78	1.91	181	12	12
RC_100_0	3,114.35	3,119.10	0.15		6,228.70		25	12	
RC_100_1	3,088.07	3,093.66	0.18	0.84	6,164.13	1.04	794	12	12
RC_100_2	3,088.07	3,100.22	0.39	0.84	6,164.13	1.04	677	12	12
RC_100_3	3,088.07	3,134.30	1.48	0.84	6,164.13	1.04	985	12	12

gains from an integrated system operation would be very small. Hence, designing such a system involves tactical decisions related to the pattern of the scheduled lines (positioning of the transfer nodes relative to the demand nodes clusters), the storage areas at the transfer nodes, and the re-design of the SL vehicles (e.g., freight compartment), that are not taken into consideration in this paper as the focus was on operational costs of the proposed system.

Overall, the results indicate the potential operating costs due to available scheduled lines. In particular, the savings range from 0 to 20% with regard to operating costs and from 0 to 23% in terms of driving time. Note that in this study the amount of CO_2e emissions is directly proportional to

Table 17
Results of R instances

Instance	Best known value	Average value	Average GAP %	Cost savings %	Best driving time	Driving time savings %	CPU seconds	# of vehicles used	Units on SLs
R_25_0	1,774.75	1,774.75	0.00		3,549.50		1	8	
R_25_1	1,747.61	1,750.59	0.17	1.53	3,491.22	1.64	2	8	4
R_25_2	1,560.51	1,576.77	1.03	12.07	3,109.01	12.41	3	8	12
R_25_3	1,560.51	1,570.65	0.65	12.07	3,109.01	12.41	3	8	12
R_50_0	2,614.49	2,623.19	0.33		5,228.98		3	12	
R_50_1	2,614.49	2,624.28	0.37	0.00	5,228.98	0.00	15	12	0
R_50_2	2,553.97	2,570.34	0.64	2.31	5,075.95	2.93	20	13	32
R_50_3	2,531.17	2,559.32	1.10	3.19	5,036.33	3.68	22	13	26
R_75_0	3,337.85	3,337.85	0.00		6,675.70		7	14	
R_75_1	3,337.85	3,355.41	0.52	0.00	6,675.70	0.00	69	14	0
R_75_2	3,321.14	3,361.77	1.21	0.50	6,630.28	0.68	98	15	12
R_75_3	3,321.14	3,349.07	0.83	0.50	6,630.28	0.68	110	15	12
R_100_0	3,643.07	3,646.89	0.10		7,286.14		26	15	
R_100_1	3,643.07	3,665.31	0.61	0.00	7,286.14	0.00	538	15	0
R_100_2	3,628.87	3,646.65	0.49	0.39	7,245.74	0.55	690	16	12
R_100_3	3,628.87	3,637.37	0.23	0.39	7,245.74	0.55	725	16	12

total driving time as we disregard the extra emissions produced by the SLs due to extra carried weight (i.e., packages).

The most of the savings can be achieved by shipping requests on the available SLs. However, the number of PD vehicles used slightly increases in PDPTW-SL compared to the solutions for PDPTW, especially in C instances. It is explained by the fact that the number of vehicles used depends on the time windows, capacities, and demands. Moreover, we note that savings decrease along with the increase in the number of requests for R and RC instances. This can be explained by the increasing density of the requests over the considered area (200×200 time units). Hence, driving time from one demand node to another becomes shorter. For C instances the savings remain significant for larger instances as well. Hence, we can conclude that the more demand points are clustered around transfer nodes, the better performance of the system is. An obvious point that is supported by the results is that the more SLs are available, more savings can be achieved compared to the classical PDPTW.

4.5. The effect of heterogeneous routing costs on the algorithm performance

In this section, we present computational experiments on the instances with heterogeneous vehicle routing costs that are based on the vehicle capacity. The minimum-capacity vehicle is assumed to cost 0.5 € per operating time unit. Larger vehicles are assigned a cost that increases linearly along with the carrying capacity. Each instance is run ten times and the results are given in Table 18. The columns are self-explanatory, similar to previously presented tables.

According to the obtained results, the proposed ALNS seems to perform stable when considering heterogeneous costs, leading to solutions with an average GAP of 0.88% compared to the best known solutions. In addition, the results indicate that objective function values tend to increase due to larger routing costs (heterogeneous costs) for the considered vehicles, compared to homogeneous case (i.e., 0.5 € for all vehicles). In this context, making use of available SLs leads to

Table 18
An analysis of heterogeneous vehicle routing costs

Instance	Best known value	Average value	Average GAP %	Cost savings %	CPU seconds	# of vehicles used	Units on SLs
25_C_0	1,106.39	1,113.83	0.67		1	5	-
25_C_1	998.32	1,010.29	1.18	9.77	6	6	10
25_RC_0	1,704.62	1,714.14	0.56		1	7	-
25_RC_1	1,449.07	1,456.16	0.49	14.99	4	7	4
25_R_0	1,916.39	1,942.96	1.37		1	8	-
25_R_1	1,861.04	1,878.43	0.93	2.89	2	8	3
50_C_0	1,623.09	1,635.45	0.76		3	8	-
50_C_1	1,459.64	1,489.81	2.03	10.07	127	9	17
50_RC_0	2,603.40	2,639.39	1.36		4	11	-
50_RC_1	2,476.35	2,510.05	1.34	4.88	24	11	6
50_R_0	2,796.81	2,815.62	0.67		4	12	-
50_R_1	2,796.81	2,802.74	0.21	0.00	15	12	0
75_C_0	2,278.57	2,309.17	1.33		8	11	-
75_C_1	1,955.53	1,972.39	0.85	14.18	698	11	25
75_RC_0	3,164.14	3,183.45	0.61		7	12	-
75_RC_1	3,144.66	3,155.30	0.34	0.62	122	12	6
75_R_0	3,583.39	3,603.99	0.57		5	14	-
75_R_1	3,572.31	3,586.76	0.40	0.31	138	15	2
100_C_0	3,985.45	4,017.42	0.80		22	12	-
100_C_1	3,557.39	3,576.59	0.54	10.74	2178	13	36
100_RC_0	3,420.74	3,461.23	1.17		9	13	-
100_RC_1	3,344.28	3,362.95	0.56	2.24	644	12	6
100_R_0	3,758.27	3,792.44	0.90		13	15	-
100_R_1	3,758.27	3,825.33	1.75	0.00	668	15	0

average cost savings of 5.89% compared to the corresponding best-known PDPTW solutions.

4.6. An application study

In this section we investigate the performance of the PDPTW-SL environment on a realistic scheduled lined system. In particular, we solve one instance of 100 randomly generated requests on a 60 x 60 time-units area, three depots and 20 PD vehicles. The scheduled lines graph is shown in Figure 5 and it is inspired from the current metro system in Amsterdam (see Figure 1). The distances are considered Euclidean and time windows are randomly generated.

Table 19
Results on a realistic real-life scheduled line system

Instance	PDPTW-SL					PDPTW			
	Driving time	# of shipments on scheduled lines	Cost €	CPU seconds	# of vehicles	Driving time	Cost €	CPU seconds	# of vehicles
Amsterdam 100 requests	1,767.22	36	919.61	1,258.52	10	1,945.69	972.85	34.62	7

The results shown in Table 19 are obtained after five runs of the algorithm and indicate the best solutions found for both PDPTW-SL and PDPTW. In particular, the PDPTW-SL integrated transportation system led to 5% savings in terms of operating costs and 9% in fewer total driving time.

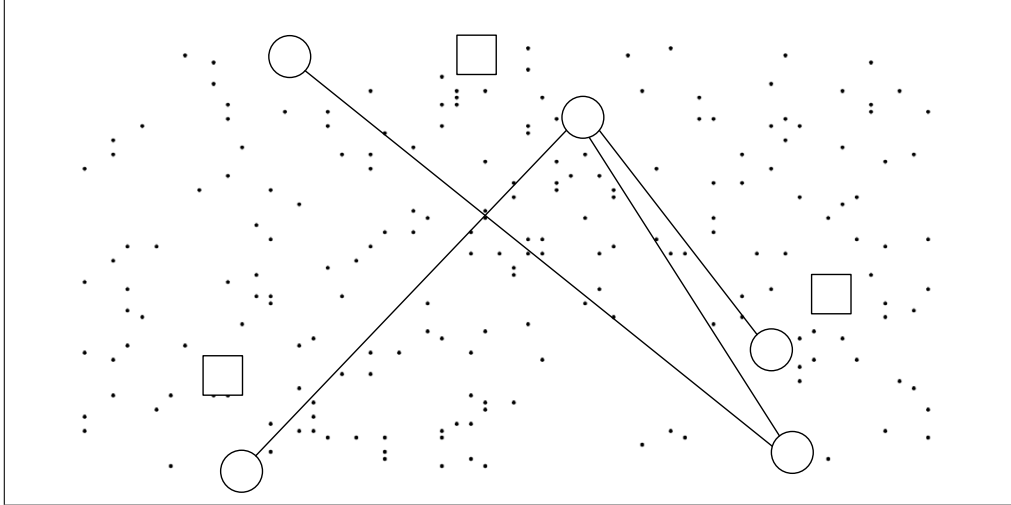


Figure 5 An illustrative representation of Amsterdam's metro system

Even though PDPTW-SL system can lead to operating costs and CO_2e emissions benefits, the number of vehicles used is increased as compared to PDPTW.

5. Conclusions

We have described a metaheuristic algorithm to solve the PDPTW-SL. To fully evaluate the effectiveness of the algorithm, we have generated different sets of instances and compiled a library of PDPTW-SL instances. Compared to the existing solutions on a set of PDPTW instances, the proposed algorithm performed well in terms of both, solution quality (with a maximum GAP of 2.79%) and CPU time (78 seconds on average). Furthermore, we have also shown that small PDPTW-SL instances can be solved optimally by the proposed formulation. The solutions obtained from solving larger instances, up to 100 requests, were compared to their corresponding PDPTW solutions and it is concluded that the flexibility of using scheduled line services leads to significant cost savings and fewer CO_2e emissions. However, note that the performance of the PDPTW-SL system may be highly dependent on the relative positioning of the scheduled lines to the request nodes. In addition, investment costs needed for implementing such system may affect its outcome.

The reliability of such a system may decrease due to extra causes of delays and cargo damages (i.e., transfers to/from SLs, delays in SL schedules). Therefore, shippers may not be willing to use PDPTW-SL system. In order to tackle such issues, more advanced planning tools are needed, which consider stochastic aspects of the problem. The current research state of the considered problem is yet young and further industry collaborations are to be done in order to learn more about practical issues that may or may not prevent the implementation and the execution of such a system.

Overall the numerical experiments show that the proposed algorithm is highly effective in finding good-quality solutions for relatively large instances in a reasonable amount of time (up to 40 minutes). Regarding extensions of the investigated problem, additional aspects may be considered such as driver-related constraints, stochastic aspects (demands, travel times) or passenger requests,

and the other related constraints (e.g., maximum ride-time). Investigation of exact decomposition algorithms (e.g., Branch & Price) could also be an interesting research direction. In addition, since the results show that the number of PD vehicles used may increase when SLs are introduced, it would be interesting to introduce fixed costs for the PD vehicles as well to make more clear the trade-off between using fewer PD vehicles or using SLs with more PD vehicles.

Acknowledgements

The authors gratefully acknowledge funds provided by Dinalog, the Dutch Institute for Advanced Logistics, under the grant titled “Cargo Hitching”, # 2011 4 086R. Thanks are due to the Editor and two anonymous reviewers for their useful comments and for raising interesting points for discussion.

Bibliography

- [1] Aldaihani, M. and Dessouky, M. (2003). Hybrid scheduling methods for paratransit operations. *Computers & Industrial Engineering*, 45(1):75–96.
- [2] Baldacci, R., Bartolini, E., and Mingozzi, A. (2011). An exact algorithm for the pickup and delivery problem with time windows. *Operations Research*, 59(2):414–426.
- [3] Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., and Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 15(1):1–31.
- [4] Braekers, K., Caris, A., and Janssens, G. (2014). Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. *Transportation Research Part B*, 67:166–186.
- [5] Cargo Tram (2012). Official webpage (accessed on March 24, 2014). Available at: www.eltis.org/index.php?id=13&study_id=1547.
- [6] Cordeau, J.-F. and Laporte, G. (2007). The dial-a-ride problem: models and algorithms. *Annals OR*, 153(1):29–46.
- [7] Cortes, E.-C., Matamala, M., and Contrado, 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.
- [8] Demir, E., Bektaş, T., and Laporte, G. (2012). An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, 223(2):346–359.
- [9] Demir, E., Bektaş, T., and Laporte, G. (2014). A review of recent research on green road freight transportation. *European Journal of Operational Research*, 237(3):775 – 793.
- [10] Demir, E., Bektaş, T., and Laporte, G. (2013). The bi-objective Pollution-Routing Problem. *European Journal of Operational Research*, 232(3):464–478.

- [11] Demir, E., Huang, Y., Scholts, S., and Van Woensel, T. (2015). A selected review on the negative externalities of the freight transportation: Modeling and pricing. *Transportation Research Part E: Logistics and Transportation Review*, 77:95–114.
- [12] DHL-Packstation (2013). DHL official webpage (accessed on November 29, 2013). Available at: www.dhl.de/en/paket/pakete-empfangen/packstation.html.
- [13] Dijkstra, E.-W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269 – 271.
- [14] Ghilas, V., Demir, E., and Van Woensel, T. (2013). Integrating passenger and freight transportation: model formulation and insights. Technical report, Industrial Engineering & Innovation Sciences, Eindhoven University of Technology. BETA No. 441, 23 pp.
- [15] Hall, C., Andersson, H., Lundgren, J., and Varbrand, P. (2009). The integrated dial-a-ride problem. *Public Transportation*, 1:39–54.
- [16] Hurtigruten (2013). Hurtigruten official webpage (accessed on November 7, 2014). Available at: www.hurtigruten-web.com/index_en.html.
- [17] IBM ILOG (2013). Copyright international business machines corporation 1987.
- [18] Levin, Y., Nediak, M., and Topaloglu, H. (2012). Cargo capacity management with allotments and spot market demand. *Operations Research*, 60(2):351–365.
- [19] Li, H. and Lim, A. (2003). A metaheuristic for the pickup and delivery problem with time windows. *International Journal on Artificial Intelligence Tools*, 12(2):173–186.
- [20] Liaw, C.-F., White, C., and Bander, J. (1996). A decision support system for the bimodal dial-a-ride problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 26(5):552–565.
- [21] Lindholm, M. and Behrends, S. (2012). Challenges in urban freight transport planning – a review in the Baltic Sea Region. *Journal of Transport Geography*, 22:129–136.
- [22] Masson, R., Lehuède, F., and Peton, O. (2012). An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science*, 47:1 – 12.
- [23] Masson, R., Lehuède, F., and Peton, O. (2014). The dial-a-ride problem with transfers. *Computers & Operations Research*, 41:12 – 23.
- [24] Nash, C., editor (1982). *Economics of public transport*. Longman, London, UK.
- [25] Petersen, H. L. and Røpke, S. (2011). The pickup and delivery problem with cross-docking opportunity. In *Computational Logistics*, volume 6971 of *Lecture Notes in Computer Science*, pages 101–113. Springer Berlin Heidelberg.
- [26] Pisinger, D. and Røpke, S. (2005). A general heuristic for vehicle routing problems. Technical report, DIKU - Department of Computer Science, University of Copenhagen. Available at: http://www.diku.dk/hjemmesider/ansatte/sropke/Papers/GeneralVRP_TechRep.pdf (accessed on November 11, 2013).

- [27] Pisinger, D. and Røpke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435.
- [28] Røpke, S. and Cordeau, J.-F. (2009). Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3):267–286.
- [29] Røpke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472.
- [30] Shang, J. and Cuff, C. (1996). Multicriteria pickup and delivery problem with transfer opportunity. *Computers & Industrial Engineering*, 30(4):631 – 645.
- [31] Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming*, pages 417–431. Springer, New York.
- [32] SmartLogisticsLab (2013). Official webpage (accessed on January 21, 2015). Available at: www.smartlogisticslab.nl.
- [33] Trentini, A. and Malhene, N. (2010). Toward a shared urban transport system ensuring passengers & goods cohabitation. *Trimestrale del Laboratorio Territorio Mobilita e Ambiente - TeMALab*, 4:37–44.
- [34] Trentini, A., Masson, R., Lehuede, F., Malhene, N., Peton, O., and Tlahig, H. (2012). A shared “passenger & goods” city logistics system. *4th International Conference on Information Systems, Logistics and Supply Chain, Quebec, Canada*.

Nr.	Year	Title	Author(s)
483	2015	An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows and Scheduled Lines	Veaceslav Ghilas, Emrah Demir, Tom Van Woensel
482	2015	Inventory Dynamics in the Financial Crisis: An Empirical Analysis of Firm Responsiveness and its Effect on Financial Performance	Kai Hoberg, Maximiliano Udenio, Jan C. Fransoo
481	2015	The extended gate problem: Intermodal hub location with multiple actors	Yann Bouchery, Jan Fransoo, Marco Slikker
480	2015	Inventory Management with Two Demand Streams: A Maintenance Application	Rob J.I. Basten, Jennifer K. Ryan
479	2015	Optimal Design of Uptime-Guarantee Contracts	Behzad Hezarkhani
478	2015	Collaborative Replenishment in the Presence of Intermediaries	Behzad Hezarkhani, Marco Slikker, Tom Van Woensel
477	2015	Reference Architecture for Mobility-Related Services A reference architecture based on GET Service and SIMPLI-CITY Project architectures	A. Husak, M. Politis, V. Shah, R. Eshuis, P. Grefen
476	2015	A Multi-Item Approach to Repairable Stocking and Expediting in a Fluctuating Demand Environment	Joachim Arts
475	2015	An Adaptive Large Neighborhood Search Heuristic for the Share-a-Ride Problem	Baoxiang Li, Dmitry Krushinsky, Tom Van Woensel, Hajo A. Reijers
474	2015	An approximate dynamic programming approach to urban freight distribution with batch arrivals	Wouter van Heeswijk, Martijn Mes, Marco Schutten
473	2015	Dynamic Multi-period Freight Consolidation	Arturo Pérez Rivera, Martijn Mes
472	2015	Maintenance policy selection for ships: finding the most important criteria and considerations	A.J.M. Goossens, R.J.I. Basten
471	2015	Using Twitter to Predict Sales: A Case Study	Remco Dijkman, Panagiotis Ipeirotis, Freek Aertsen, Roy van Helden
470	2015	The Effect of Exceptions in Business Processes	Remco Dijkman, Geoffrey van IJzendoorn, Oktay Türetken, Meint de Vries
469	2015	Business Model Prototyping for Intelligent Transport Systems. A Service-Dominant Approach	Konstantinos Traganos, Paul Grefen, Aafke den Hollander, Oktay Türetken, Rik Eshuis
468	2015	How suitable is the RePro technique for rethinking care processes?	Rob J.B. Vanwersch, Luise Pufahl, Irene Vanderfeesten, Jan Mendling, Hajo A. Reijers
467	2014	Where to exert abatement effort for sustainable operations considering supply chain interactions?	Tarkan Tan, Astrid Koomen
466	2014	An Exact Algorithm for the Vehicle Routing Problem with Time Windows and Shifts	Said Dabia, Stefan Ropke, Tom Van Woensel
465	2014	The RePro technique: a new, systematic technique for rethinking care processes	Rob J.B. Vanwersch, Luise Pufahl, Irene Vanderfeesten, Hajo A. Reijers
464	2014	Exploring maintenance policy selection using the Analytic Hierarchy Process: an application for naval ships	A.J.M. Goossens, R.J.I. Basten
463	2014	Allocating service parts in two-echelon networks at a utility company	D. van den Berg, M.C. van der Heijden, P.C. Schuur
462	2014	Freight consolidation in networks with transshipments	W.J.A. van Heeswijk, M.R.K. Mes, J.M.J. Schutten, W.H.M. Zijm
461	2014	A Software Architecture for a Transportation Control Tower	Anne Baumgrass, Remco Dijkman, Paul Grefen, Shaya Pourmirza, Hagen Völzer, Mathias Weske
460	2014	Small traditional retailers in emerging markets	Youssef Boulaksil, Jan C. Fransoo, Edgar E. Blanco, Sallem Koubida
459	2014	Defining line replaceable units	J.E. Parada Puig, R.J.I. Basten

Nr.	Year	Title	Author(s)
458	2014	Inventories and the Credit Crisis: A Chicken and Egg Situation	Maximiliano Udenio, Vishal Gaur, Jan C. Fransoo
457	2014	An Exact Approach for the Pollution-Routing Problem	Said Dabia, Emrah Demir, Tom Van Woensel
456	2014	Fleet readiness: stocking spare parts and high-tech assets	Rob J.I. Basten, Joachim J. Arts
455	2014	Competitive Solutions for Cooperating Logistics Providers	Behzad Hezarkhani, Marco Slikker, Tom Van Woensel
454	2014	Simulation Framework to Analyse Operating Room Release Mechanisms	Rimmert van der Kooij, Martijn Mes, Erwin Hans
453	2014	A Unified Race Algorithm for Offline Parameter Tuning	Tim van Dijk, Martijn Mes, Marco Schutten, Joaquim Gromicho
452	2014	Cost, carbon emissions and modal shift in intermodal network design decisions	Yann Bouchery, Jan Fransoo
451	2014	Transportation Cost and CO2 Emissions in Location Decision Models	Josue C. Vélazquez-Martínez, Jan C. Fransoo, Edgar E. Blanco, Jaime Mora-Vargas
450	2014	Tracebook: A Dynamic Checklist Support System	Shan Nan, Pieter Van Gorp, Hendrikus H.M. Korsten, Richard Vdovjak, Uzay Kaymak
449	2014	Intermodal hinterland network design with multiple actors	Yann Bouchery, Jan Fransoo
448	2014	The Share-a-Ride Problem: People and Parcels Sharing Taxis	Baoxiang Li, Dmitry Krushinsky, Hajo A. Reijers, Tom Van Woensel
447	2014	Stochastic inventory models for a single item at a single location	K.H. van Donselaar, R.A.C.M. Broekmeulen
446	2014	Optimal and heuristic repairable stocking and expediting in a fluctuating demand environment	Joachim Arts, Rob Basten, Geert-Jan van Houtum
445	2014	Connecting inventory control and repair shop control: a differentiated control structure for repairable spare parts	M.A. Driessen, W.D. Rustenburg, G.J. van Houtum, V.C.S. Wiers
444	2014	A survey on design and usage of Software Reference Architectures	Samuil Angelov, Jos Trienekens, Rob Kusters
443	2014	Extending and Adapting the Architecture Tradeoff Analysis Method for the Evaluation of Software Reference Architectures	Samuil Angelov, Jos J.M. Trienekens, Paul Grefen
442	2014	A multimodal network flow problem with product quality preservation, transshipment, and asset management	Maryam SteadieSeifi, Nico Dellaert, Tom Van Woensel
441	2013	Integrating passenger and freight transportation: Model formulation and insights	Veaceslav Ghilas, Emrah Demir, Tom Van Woensel
440	2013	The Price of Payment Delay	K. van der Vliet, M.J. Reindorp, J.C. Fransoo
439	2013	On Characterization of the Core of Lane Covering Games via Dual Solutions	Behzad Hezarkhani, Marco Slikker, Tom van Woensel
438	2013	Destocking, the Bullwhip Effect, and the Credit Crisis: Empirical Modeling of Supply Chain Dynamics	Maximiliano Udenio, Jan C. Fransoo, Robert Peels
437	2013	Methodological support for business process redesign in healthcare: a systematic literature review	Rob J.B. Vanwersch, Khurram Shahzad, Irene Vanderfeesten, Kris Vanhaecht, Paul Grefen, Liliane Pintelon, Jan Mendling, Geofridus G. van Merode, Hajo A. Reijers
436	2013	Dynamics and equilibria under incremental horizontal differentiation on the Salop circle	B. Vermeulen, J.A. La Poutré, A.G. de Kok
435	2013	Analyzing Conformance to Clinical Protocols Involving Advanced Synchronizations	Hui Yan, Pieter Van Gorp, Uzay Kaymak, Xudong Lu, Richard Vdovjak, Hendriks H.M. Korsten, Huilong Duan
434	2013	Models for Ambulance Planning on the Strategic and the Tactical Level	J. Theresia van Essen, Johann L. Hurink, Stefan Nickel, Melanie Reuter

Nr.	Year	Title	Author(s)
433	2013	Mode Allocation and Scheduling of Inland Container Transportation: A Case-Study in the Netherlands	Stefano Fazi, Tom Van Woensel, Jan C. Fransoo
432	2013	Socially responsible transportation and lot sizing: Insights from multiobjective optimization	Yann Bouchery, Asma Ghaffari, Zied Jemai, Jan Fransoo
431	2013	Inventory routing for dynamic waste collection	Martijn Mes, Marco Schutten, Arturo Pérez Rivera
430	2013	Simulation and Logistics Optimization of an Integrated Emergency Post	N.J. Borgman, M.R.K. Mes, I.M.H. Vliegen, E.W. Hans
429	2013	Last Time Buy and Repair Decisions for Spare Parts	S. Behfard, M.C. van der Heijden, A. Al Hanbali, W.H.M. Zijm
428	2013	A Review of Recent Research on Green Road Freight Transportation	Emrah Demir, Tolga Bektas, Gilbert Laporte
427	2013	Typology of Repair Shops for Maintenance Spare Parts	M.A. Driessen, V.C.S. Wiers, G.J. van Houtum, W.D. Rustenburg
426	2013	A value network development model and implications for innovation and production network management	B. Vermeulen, A.G. de Kok
425	2013	Single Vehicle Routing with Stochastic Demands: Approximate Dynamic Programming	C. Zhang, N.P. Dellaert, L. Zhao, T. Van Woensel, D. Sever
424	2013	Influence of Spillback Effect on Dynamic Shortest Path Problems with Travel-Time-Dependent Network Disruptions	Derya Sever, Nico Dellaert, Tom Van Woensel, Ton de Kok
423	2013	Dynamic Shortest Path Problem with Travel-Time-Dependent Stochastic Disruptions: Hybrid Approximate Dynamic Programming Algorithms with a Clustering Approach	Derya Sever, Lei Zhao, Nico Dellaert, Tom Van Woensel, Ton de Kok
422	2013	System-oriented inventory models for spare parts	R.J.I. Basten, G.J. van Houtum
421	2013	Lost Sales Inventory Models with Batch Ordering and Handling Costs	T. Van Woensel, N. Erkip, A. Curseu, J.C. Fransoo
420	2013	Response speed and the bullwhip	Maximiliano Udenio, Jan C. Fransoo, Eleni Vatamidou, Nico Dellaert
419	2013	Anticipatory Routing of Police Helicopters	Rick van Urk, Martijn R.K. Mes, Erwin W. Hans
418	2013	Supply Chain Finance: research challenges ahead	Kasper van der Vliet, Matthew J. Reindorp, Jan C. Fransoo
417	2013	Improving the Performance of Sorter Systems by Scheduling Inbound Containers	S.W.A. Haneyah, J.M.J. Schutten, K. Fikse
416	2013	Regional logistics land allocation policies: Stimulating spatial concentration of logistics firms	Frank P. van den Heuvel, Peter W. de Langen, Karel H. van Donselaar, Jan C. Fransoo
415	2013	The development of measures of process harmonization	Heidi L. Romero, Remco M. Dijkman, Paul W.P.J. Grefen, Arjan van Weele
414	2013	BASE/X. Business Agility through Cross-Organizational Service Engineering. The Business and Service Design Approach developed in the CoProFind Project	Paul Grefen, Egon Lüftenegger, Eric van der Linden, Caren Weisleder
413	2013	The Time-Dependent Vehicle Routing Problem with Soft Time Windows and Stochastic Travel Times	Duygu Tas, Nico Dellaert, Tom van Woensel, Ton de Kok
412	2013	Clearing the Sky - Understanding SLA Elements in Cloud Computing	Marco Comuzzi, Guus Jacobs, Paul Grefen
411	2013	Approximations for the waiting time distribution in an M/G/c priority queue	A. Al Hanbali, E.M. Alvarez, M.C. van der Heijden
410	2013	To co-locate or not? Location decisions and logistics concentration areas	Frank P. van den Heuvel, Karel H. van Donselaar, Rob A.C.M. Broekmeulen, Jan C. Fransoo, Peter W. de Langen

Nr.	Year	Title	Author(s)
409	2013	The Time-Dependent Pollution-Routing Problem	Anna Franceschetti, Dorothée Honhon, Tom van Woensel, Tolga Bektas, Gilbert Laporte
408	2013	Scheduling the scheduling task: A time management perspective on scheduling	J.A. Larco, V. Wiers, J. Fransoo
407	2013	Clustering Clinical Departments for Wards to Achieve a Prespecified Blocking Probability	J. Theresia van Essen, Mark van Houdenhoven, Johann L. Hurink
406	2013	MyPHRMachines: Personal Health Desktops in the Cloud	Pieter Van Gorp, Marco Comuzzi
405	2013	Maximising the Value of Supply Chain Finance	Kasper van der Vliet, Matthew J. Reindorp, Jan C. Fransoo
404	2013	Reaching 50 million nanostores: retail distribution in emerging megacities	Edgar E. Blanco, Jan C. Fransoo
403	2013	A Vehicle Routing Problem with Flexible Time Windows	Duygu Tas, Ola Jabali, Tom van Woensel
402	2013	The Service Dominant Business Model: A Service Focused Conceptualization	Egon Lüftenegger, Marco Comuzzi, Paul Grefen, Caren Weisleder
401	2013	Relationship between freight accessibility and logistics employment in US counties	Frank P. van den Heuvel, Liliana Rivera, Karel H. van Donselaar, Ad de Jong, Yossi Sheffi, Peter W. de Langen, Jan C. Fransoo
400	2012	A Condition-Based Maintenance Policy for Multi-Component Systems with a High Maintenance Setup Cost	Qiushi Zhu, Hao Peng, Geert-Jan van Houtum
399	2012	A flexible iterative improvement heuristic to support creation of feasible shift rosters in self-rostering	E. van der Veen, J.L. Hurink, J.M.J. Schutten, S.T. Uijland
398	2012	Scheduled Service Network Design with Synchronization and Transshipment Constraints for Intermodal Container Transportation Networks	K. Sharypova, T.G. Crainic, T. van Woensel, J.C. Fransoo
397	2012	Destocking, the bullwhip effect, and the credit crisis: empirical modeling of supply chain dynamics	Maximiliano Udenio, Jan C. Fransoo, Robert Peels
396	2012	Vehicle routing with restricted loading capacities	J. Gromicho, J.J. van Hoorn, A.L. Kok, J.M.J. Schutten
395	2012	Service differentiation through selective lateral transshipments	E.M. Alvarez, M.C. van der Heijden, I.M.H. Vliegen, W.H.M. Zijm
394	2012	A Generalized Simulation Model of an Integrated Emergency Post	Martijn Mes, Manon Bruens
393	2012	Business Process Technology and the Cloud: defining a Business Process Cloud Platform	Vassil Stoitsev, Paul Grefen
392	2012	Vehicle Routing with Soft Time Windows and Stochastic Travel Times: A Column Generation and Branch-and-Price Solution Approach	D. Tas, M. Gendreau, N. Dellaert, T. van Woensel, A.G. de Kok
391	2012	Improve OR-Schedule to Reduce Number of Required Beds	J. Theresia van Essen, Joël M. Bosch, Erwin W. Hans, Mark van Houdenhoven, Johann L. Hurink
390	2012	How does development lead time affect performance over the ramp-up lifecycle? Evidence from the consumer electronics industry	Andreas Pufall, Jan C. Fransoo, Ad de Jong, A.G. (Ton) de Kok
389	2012	The Impact of Product Complexity on Ramp-Up Performance	Andreas Pufall, Jan C. Fransoo, Ad de Jong, A.G. (Ton) de Kok
388	2012	Co-location synergies: specialized versus diverse logistics concentration areas	Frank P. van den Heuvel, Peter W. de Langen, Karel H. van Donselaar, Jan C. Fransoo
387	2012	Proximity matters: Synergies through co-location of logistics establishments	Frank P. van den Heuvel, Peter W. de Langen, Karel H. van Donselaar, Jan C. Fransoo

Nr.	Year	Title	Author(s)
386	2012	Spatial concentration and location dynamics in logistics: the case of a Dutch province	Frank P. van den Heuvel, Peter W. de Langen, Karel H. van Donselaar, Jan C. Fransoo
385	2012	FNet: An Index for Advanced Business Process Querying	Zhiqiang Yan, Remco Dijkman, Paul Grefen
384	2012	Defining Various Pathway Terms	W.R. Dalinghaus, P.M.E. Van Gorp
383	2012	The Service Dominant Strategy Canvas: Defining and Visualizing a Service Dominant Strategy through the Traditional Strategic Lens	Egon Lüftenegger, Paul Grefen, Caren Weisleder
382	2012	A Stochastic Variable Size Bin Packing Problem with Time Constraints	Stefano Fazi, Tom van Woensel, Jan C. Fransoo
381	2012	Coordination and Analysis of Barge Container Hinterland Networks	K. Sharypova, T. van Woensel, J.C. Fransoo
380	2012	Proximity matters: Synergies through co-location of logistics establishments	Frank P. van den Heuvel, Peter W. de Langen, Karel H. van Donselaar, Jan C. Fransoo
379	2012	A literature review in process harmonization: a conceptual framework	Heidi Romero, Remco Dijkman, Paul Grefen, Arjan van Weele
378	2012	A Generic Material Flow Control Model for Two Different Industries	S.W.A. Haneyah, J.M.J. Schutten, P.C. Schuur, W.H.M. Zijm
377	2012	Dynamic demand fulfillment in spare parts networks with multiple customer classes	H.G.H. Tiemessen, M. Fleischmann, G.J. van Houtum, J.A.E.E. van Nunen, E. Pratsini
376	2012	Paper has been replaced by wp 417	K. Fikse, S.W.A. Haneyah, J.M.J. Schutten
375	2012	Strategies for dynamic appointment making by container terminals	Albert Douma, Martijn Mes
374	2012	MyPHRMachines: Lifelong Personal Health Records in the Cloud	Pieter van Gorp, Marco Comuzzi
373	2012	Service differentiation in spare parts supply through dedicated stocks	E.M. Alvarez, M.C. van der Heijden, W.H.M. Zijm
372	2012	Spare parts inventory pooling: how to share the benefits?	Frank Karsten, Rob Basten
371	2012	Condition based spare parts supply	X. Lin, R.J.I. Basten, A.A. Kranenburg, G.J. van Houtum
370	2012	Using Simulation to Assess the Opportunities of Dynamic Waste Collection	Martijn Mes
369	2012	Aggregate overhaul and supply chain planning for rotables	J. Arts, S.D. Flapper, K. Vernooij
368	2012	Operating Room Rescheduling	J.T. van Essen, J.L. Hurink, W. Hartholt, B.J. van den Akker
367	2011	Switching Transport Modes to Meet Voluntary Carbon Emission Targets	Kristel M.R. Hoen, Tarkan Tan, Jan C. Fransoo, Geert-Jan van Houtum
366	2011	On two-echelon inventory systems with Poisson demand and lost sales	Elisa Alvarez, Matthieu van der Heijden
365	2011	Minimizing the Waiting Time for Emergency Surgery	J.T. van Essen, E.W. Hans, J.L. Hurink, A. Oversberg
364	2012	Vehicle Routing Problem with Stochastic Travel Times Including Soft Time Windows and Service Costs	Duygu Tas, Nico Dellaert, Tom van Woensel, Ton de Kok
363	2011	A New Approximate Evaluation Method for Two-Echelon Inventory Systems with Emergency Shipments	Erhun Özkan, Geert-Jan van Houtum, Yasemin Serin
362	2011	Approximating Multi-Objective Time-Dependent Optimization Problems	Said Dabia, El-Ghazali Talbi, Tom Van Woensel, Ton de Kok
361	2011	Branch and Cut and Price for the Time Dependent Vehicle Routing Problem with Time Windows	Said Dabia, Stefan Röpke, Tom Van Woensel, Ton de Kok
360	2011	Analysis of an Assemble-to-Order System with Different Review Periods	A.G. Karaarslan, G.P. Kiesmüller, A.G. de Kok

Nr.	Year	Title	Author(s)
359	2011	Interval Availability Analysis of a Two-Echelon, Multi-Item System	Ahmad Al Hanbali, Matthieu van der Heijden
358	2011	Carbon-Optimal and Carbon-Neutral Supply Chains	Felipe Caro, Charles J. Corbett, Tarkan Tan, Rob Zuidwijk
357	2011	Generic Planning and Control of Automated Material Handling Systems: Practical Requirements Versus Existing Theory	Sameh Haneyah, Henk Zijm, Marco Schutten, Peter Schuur
356	2011	Last time buy decisions for products sold under warranty	Matthieu van der Heijden, Bermawi Iskandar
355	2011	Spatial concentration and location dynamics in logistics: the case of a Dutch province	Frank P. van den Heuvel, Peter W. de Langen, Karel H. van Donselaar, Jan C. Fransoo
354	2011	Identification of Employment Concentration Areas	Frank P. van den Heuvel, Peter W. de Langen, Karel H. van Donselaar, Jan C. Fransoo
353	2011	BPMN 2.0 Execution Semantics Formalized as Graph Rewrite Rules: extended version	Pieter van Gorp, Remco Dijkman
352	2011	Resource pooling and cost allocation among independent service providers	Frank Karsten, Marco Slikker, Geert-Jan van Houtum
351	2011	A Framework for Business Innovation Directions	E. Lüftenegger, S. Angelov, P. Grefen
350	2011	The Road to a Business Process Architecture: An Overview of Approaches and their Use	Remco Dijkman, Irene Vanderfeesten, Hajo A. Reijers
349	2011	Effect of carbon emission regulations on transport mode selection under stochastic demand	K.M.R. Hoen, T. Tan, J.C. Fransoo, G.J. van Houtum
348	2011	An improved MIP-based combinatorial approach for a multi-skill workforce scheduling problem	Murat Firat, Cor Hurkens
347	2011	An approximate approach for the joint problem of level of repair analysis and spare parts stocking	R.J.I. Basten, M.C. van der Heijden, J.M.J. Schutten
346	2011	Joint optimization of level of repair analysis and spare parts stocks	R.J.I. Basten, M.C. van der Heijden, J.M.J. Schutten
345	2011	Inventory control with manufacturing lead time flexibility	Ton G. de Kok
344	2011	Analysis of resource pooling games via a new extension of the Erlang loss function	Frank Karsten, Marco Slikker, Geert-Jan van Houtum
343	2011	Vehicle refueling with limited resources	Murat Firat, C.A.J. Hurkens, Gerhard J. Woeginger
342	2011	Optimal Inventory Policies with Non-stationary Supply Disruptions and Advance Supply Information	Bilge Atasoy, Refik Güllü, Tarkan Tan
341	2011	Redundancy Optimization for Critical Components in High-Availability Capital Goods	Kurtulus Baris Öner, Alan Scheller-Wolf, Geert-Jan van Houtum
340	2011	Making Decision Process Knowledge Explicit Using the Product Data Model	Razvan Petrusel, Irene Vanderfeesten, Cristina Claudia Dolean, Daniel Mican
339	2010	Analysis of a two-echelon inventory system with two supply modes	Joachim Arts, Gudrun Kiesmüller
338	2010	Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh	Murat Firat, Gerhard J. Woeginger
335	2010	Attaining stability in multi-skill workforce scheduling	Murat Firat, Cor Hurkens
334	2010	Flexible Heuristics Miner (FHM)	A.J.M.M. Weijters, J.T.S. Ribeiro
333	2010	An exact approach for relating recovering surgical patient workload to the master surgical schedule	P.T. Vanberkel, R.J. Boucherie, E.W. Hans, J.L. Hurink, W.A.M. van Lent, W.H. van Harten
332	2010	Efficiency evaluation for pooling resources in health care	Peter T. Vanberkel, Richard J. Boucherie, Erwin W. Hans, Johann L. Hurink, Nelly Litvak
331	2010	The Effect of Workload Constraints in Mathematical Programming Models for Production Planning	M.M. Jansen, A.G. de Kok, I.J.B.F. Adan

Nr.	Year	Title	Author(s)
330	2010	Using pipeline information in a multi-echelon spare parts inventory system	Christian Howard, Ingrid Reijnen, Johan Marklund, Tarkan Tan
329	2010	Reducing costs of repairable spare parts supply systems via dynamic scheduling	H.G.H. Tiemessen, G.J. van Houtum
328	2010	Identification of Employment Concentration and Specialization Areas: Theory and Application	Frank P. van den Heuvel, Peter W. de Langen, Karel H. van Donselaar, Jan C. Fransoo
327	2010	A combinatorial approach to multi-skill workforce scheduling	M. Firat, C. Hurkens
326	2010	Stability in multi-skill workforce scheduling	M. Firat, C. Hurkens, A. Laugier
325	2010	Maintenance spare parts planning and control: A framework for control and agenda for future research	M.A. Driessen, J.J. Arts, G.J. van Houtum, W.D. Rustenburg, B. Huisman
324	2010	Near-optimal heuristics to set base stock levels in a two-echelon distribution network	R.J.I. Basten, G.J. van Houtum
323	2010	Inventory reduction in spare part networks by selective throughput time reduction	M.C. van der Heijden, E.M. Alvarez, J.M.J. Schutten
322	2010	The selective use of emergency shipments for service-contract differentiation	E.M. Alvarez, M.C. van der Heijden, W.H.M. Zijm
321	2010	Heuristics for Multi-Item Two-Echelon Spare Parts Inventory Control Problem with Batch Ordering in the Central Warehouse	Engin Topan, Z. Pelin Bayindir, Tarkan Tan
320	2010	Preventing or escaping the suppression mechanism: intervention conditions	Bob Walrave, Kim E. van Oorschot, A. Georges L. Romme
319	2010	Hospital admission planning to optimize major resources utilization under uncertainty	Nico Dellaert, Jully Jeunet
318	2010	Minimal Protocol Adaptors for Interacting Services	R. Seguel, R. Eshuis, P. Grefen
317	2010	Teaching Retail Operations in Business and Engineering Schools	Tom Van Woensel, Marshall L. Fisher, Jan C. Fransoo
316	2010	Design for Availability: Creating Value for Manufacturers and Customers	Lydie P.M. Smets, Geert-Jan van Houtum, Fred Langerak
315	2010	Transforming Process Models: executable rewrite rules versus a formalized Java program	Pieter van Gorp, Rik Eshuis
314	2010	Working paper 314 is no longer available	----
313	2010	A Dynamic Programming Approach to Multi-Objective Time-Dependent Capacitated Single Vehicle Routing Problems with Time Windows	S. Dabia, T. van Woensel, A.G. de Kok
312	2010	Tales of a So(u)rcerer: Optimal Sourcing Decisions Under Alternative Capacitated Suppliers and General Cost Structures	Osman Alp, Tarkan Tan
311	2010	In-store replenishment procedures for perishable inventory in a retail environment with handling costs and storage constraints	R.A.C.M. Broekmeulen, C.H.M. Bakx
310	2010	The state of the art of innovation-driven business models in the financial services industry	E. Lüftenegger, S. Angelov, E. van der Linden, P. Grefen
309	2010	Design of Complex Architectures Using a Three Dimension Approach: the CrossWork Case	R. Seguel, P. Grefen, R. Eshuis
308	2010	Effect of carbon emission regulations on transport mode selection in supply chains	K.M.R. Hoen, T. Tan, J.C. Fransoo, G.J. van Houtum
307	2010	Interaction between intelligent agent strategies for real-time transportation planning	Martijn Mes, Matthieu van der Heijden, Peter Schuur
306	2010	Internal Slackening Scoring Methods	Marco Slikker, Peter Borm, René van den Brink
305	2010	Vehicle Routing with Traffic Congestion and Drivers' Driving and Working Rules	A.L. Kok, E.W. Hans, J.M.J. Schutten, W.H.M. Zijm
304	2010	Practical extensions to the level of repair analysis	R.J.I. Basten, M.C. van der Heijden, J.M.J. Schutten

Nr.	Year	Title	Author(s)
303	2010	Ocean Container Transport: An Underestimated and Critical Link in Global Supply Chain Performance	Jan C. Fransoo, Chung-Yee Lee
302	2010	Capacity reservation and utilization for a manufacturer with uncertain capacity and demand	Y. Boulaksil; J.C. Fransoo; T. Tan
300	2009	Spare parts inventory pooling games	F.J.P. Karsten; M. Slikker; G.J. van Houtum
299	2009	Capacity flexibility allocation in an outsourced supply chain with reservation	Y. Boulaksil, M. Grunow, J.C. Fransoo
298	2010	An optimal approach for the joint problem of level of repair analysis and spare parts stocking	R.J.I. Basten, M.C. van der Heijden, J.M.J. Schutten
297	2009	Responding to the Lehman Wave: Sales Forecasting and Supply Management during the Credit Crisis	Robert Peels, Maximiliano Udenio, Jan C. Fransoo, Marcel Wolfs, Tom Hendrikx
296	2009	An exact approach for relating recovering surgical patient workload to the master surgical schedule	Peter T. Vanberkel, Richard J. Boucherie, Erwin W. Hans, Johann L. Hurink, Wineke A.M. van Lent, Wim H. van Harten
295	2009	An iterative method for the simultaneous optimization of repair decisions and spare parts stocks	R.J.I. Basten, M.C. van der Heijden, J.M.J. Schutten
294	2009	Fujaba hits the Wall(-e)	Pieter van Gorp, Ruben Jubeh, Bernhard Grusie, Anne Keller
293	2009	Implementation of a Healthcare Process in Four Different Workflow Systems	R.S. Mans, W.M.P. van der Aalst, N.C. Russell, P.J.M. Bakker
292	2009	Business Process Model Repositories - Framework and Survey	Zhiqiang Yan, Remco Dijkman, Paul Grefen
291	2009	Efficient Optimization of the Dual-Index Policy Using Markov Chains	Joachim Arts, Marcel van Vuuren, Gudrun Kiesmuller
290	2009	Hierarchical Knowledge-Gradient for Sequential Sampling	Martijn R.K. Mes; Warren B. Powell; Peter I. Frazier
289	2009	Analyzing combined vehicle routing and break scheduling from a distributed decision making perspective	C.M. Meyer; A.L. Kok; H. Kopfer; J.M.J. Schutten
288	2010	Lead time anticipation in Supply Chain Operations Planning	Michiel Jansen; Ton G. de Kok; Jan C. Fransoo
287	2009	Inventory Models with Lateral Transshipments: A Review	Colin Paterson; Gudrun Kiesmuller; Ruud Teunter; Kevin Glazebrook
286	2009	Efficiency evaluation for pooling resources in health care	P.T. Vanberkel; R.J. Boucherie; E.W. Hans; J.L. Hurink; N. Litvak
285	2009	A Survey of Health Care Models that Encompass Multiple Departments	P.T. Vanberkel; R.J. Boucherie; E.W. Hans; J.L. Hurink; N. Litvak
284	2009	Supporting Process Control in Business Collaborations	S. Angelov; K. Vidyasankar; J. Vonk; P. Grefen
283	2009	Inventory Control with Partial Batch Ordering	O. Alp; W.T. Huh; T. Tan
282	2009	Translating Safe Petri Nets to Statecharts in a Structure-Preserving Way	R. Eshuis
281	2009	The link between product data model and process model	J.J.C.L. Vogelaar; H.A. Reijers
280	2009	Inventory planning for spare parts networks with delivery time requirements	I.C. Reijnen; T. Tan; G.J. van Houtum
279	2009	Co-Evolution of Demand and Supply under Competition	B. Vermeulen; A.G. de Kok
278	2010	Toward Meso-level Product-Market Network Indices for Strategic Product Selection and (Re)Design Guidelines over the Product Life-Cycle	B. Vermeulen, A.G. de Kok
277	2009	An Efficient Method to Construct Minimal Protocol Adaptors	R. Seguel, R. Eshuis, P. Grefen

Nr.	Year	Title	Author(s)
276	2009	Coordinating Supply Chains: a Bilevel Programming Approach	Ton G. de Kok, Gabriella Muratore
275	2009	Inventory redistribution for fashion products under demand parameter update	G.P. Kiesmuller, S. Minner
274	2009	Comparing Markov chains: Combining aggregation and precedence relations applied to sets of states	A. Basic, I.M.H. Vliegen, A. Scheller-Wolf
273	2009	Separate tools or tool kits: an exploratory study of engineers' preferences	I.M.H. Vliegen, P.A.M. Kleingeld, G.J. van Houtum
272	2009	An Exact Solution Procedure for Multi-Item Two-Echelon Spare Parts Inventory Control Problem with Batch Ordering	
271	2009	Distributed Decision Making in Combined Vehicle Routing and Break Scheduling	C.M. Meyer, H. Kopfer, A.L. Kok, M. Schutten
270	2009	Dynamic Programming Algorithm for the Vehicle Routing Problem with Time Windows and EC Social Legislation	A.L. Kok, C.M. Meyer, H. Kopfer, J.M.J. Schutten
269	2009	Similarity of Business Process Models: Metrics and Evaluation	Remco Dijkman, Marlon Dumas, Boudewijn van Dongen, Reina Kaarik, Jan Mendling
267	2009	Vehicle routing under time-dependent travel times: the impact of congestion avoidance	A.L. Kok, E.W. Hans, J.M.J. Schutten
266	2009	Restricted dynamic programming: a flexible framework for solving realistic VRPs	J. Gromicho; J.J. van Hoorn; A.L. Kok; J.M.J. Schutten;

Working Papers published before 2009 see: <http://beta.ieis.tue.nl>